



Universidad de
Oviedo



ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

MÁSTER EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

**ÁREA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, DE
COMPUTADORES Y SISTEMAS**

TRABAJO FIN DE MÁSTER N° TFM18010103

**Desarrollo y ensayo de un sistema de detección de fallos
en plantas de generación solar mediante inspección aérea automática**

**VALES-VILLAMARÍN SANJURJO, JORGE
TUTOR: PÉREZ GARCÍA, MIGUEL ÁNGEL**

FECHA: JULIO - 2018

Contenido

1. Introducción	18
2. Objetivos Generales	20
3. Contexto, alcance y planificación	21
3.1.- Contexto.....	21
3.2.- ALCANCE.....	21
3.3.- PLANIFICACIÓN	22
4. Estado del arte	27
4.1.- Concepto	27
4.2.- Origen e historia.....	27
4.3.- Actualidad	32
4.3.1.- Seguridad	32
4.3.2.- Entretenimiento.....	32
4.3.3.- Ciencia e investigación	33
4.4.- Proyectos futuros	34
4.4.1.- Aquila de Facebook	34
4.4.2.- Drones repartidores de Amazon.....	34
4.5.- Misiones de inspección con drones en instalaciones industriales.....	35
4.6.- Normativa	35
5. GENERALIDADES.....	36
5.1.- Elementos que constituyen un UAV	36
5.1.1.- Chasis o plataforma.....	36
5.1.2.- Carga	36
5.1.3.- Controladora	36
5.1.4.- Comunicación	36
5.2.- Clasificación de UAVs	36
6. UAV-Inspection	38
6.1.- Definición del sistema previo	38
6.1.1.- Partes del sistema.....	38
6.1.2.- Conexiones previas	44
6.2.- Sistemas	45
7. LED	46
7.1.- Problemática existente	46

7.2.- Conexión hardware	47
7.3.- Software	48
7.3.1.- Optimización del código	49
8. Envío automático de la misión	52
8.1.- Introducción	52
8.2.- HARDWARE.....	52
8.3.- SOFTWARE	56
8.3.1.- Introducción al “boot process” de Linux (“startup sequence”).....	56
8.3.2.- Programación	57
8.3.3.- Multiplexación de terminal	60
8.3.4.- Mavproxy.py	63
8.3.5.- MAVLink	63
8.3.6.- ¿Cómo se realiza la comunicación usando el protocolo MAVLink? Estructura de un paquete MAVLink.	63
8.3.7.- Instalación de Mavproxy y GNU screen	64
8.3.8.- RPI_to_PIX_xxx	66
8.4.- Prueba de funcionamiento	68
8.5.- Implementación de mecanismo de seguridad	71
8.5.1.- Prueba de funcionamiento	74
9. RTK	80
9.1.- Problemática inicial.....	80
9.2.- Principio teórico	80
9.3.- partes del sistema RTK	81
9.4.- Notación importante.....	82
9.5.- Conexiones físicas.....	82
9.6.- procedimiento de aplicación e integración en el sistema	83
9.6.1.- Encendido.....	83
9.6.2.- Conexión con “ReachView”	83
9.6.3.- Ajustando la conexión Wi-Fi	84
9.6.4.- Acceso al dispositivo Reach RS dentro de una red Wi-Fi	84
9.6.5.- Ajustes de la estación base	85
9.6.6.- Ajustes de la estación Rover	86
9.6.7.- Visionado de resultados	86

9.6.8.- Resumen de interfaz ReachView	86
9.6.9.- Añadir una nueva red Wi-Fi	86
9.7.- PRUEBAS DE CAMPO rtk.....	87
9.7.1.- Prueba de campo 1	87
9.7.2.- Prueba de campo 2	88
9.7.3.- Prueba de campo 3	90
9.7.4.- Prueba de campo 4	92
9.8.- Conclusiones del uso del dispositivo	94
9.9.- Procedimiento de uso en campo	94
10. Lidar	95
10.1.- Introducción	95
10.2.- Conexión con la Raspberry	96
10.2.1.- Conexión hardware: Standard I2C.....	97
10.2.2.- Software: Standard I2C.....	98
10.2.3.- Conexión hardware: PWM	100
10.2.4.- Software: PWM	101
10.3.- Pruebas de funcionamiento	102
10.3.1.- Primera prueba (Toma de medidas con placa Arduino Uno y PWM).....	102
10.3.2.- Segunda prueba (Toma de medidas con placa Raspberry pi3 y I2C).....	104
10.3.3.- Tercera prueba toma de medidas (Raspberry y PWM).....	112
10.3.4.- Exportar medidas tomadas por el Lidar a un fichero .txt.....	115
10.3.5.- Exactitud en las mediciones del dispositivo Lidar.....	116
11. Integración FPV	117
11.1.- Introducción	117
11.2.- Dispositivos requeridos.....	118
11.2.1.- Transmisor (Tx)	118
11.2.2.- OSD (“on screen display”).....	120
11.3.- Captura de video y “Stream” vía Wi-Fi con Raspi-cam.	121
11.4.- Captura de video con GStreamer	122
11.5.- “Stream”.....	123
11.6.- Transmisión de video desde la Raspberry pi mediante el puerto jack.....	124
11.6.1.- Señal de video a transmitir: “Composite” video o video compuesto.....	124
11.6.2.- Introducción	125

11.6.3.- Software	126
11.6.4.- Primera prueba: Emisión de video desde la Raspberry a una pantalla.....	127
11.6.5.- Segunda prueba: Funcionamiento de la emisora de radiofrecuencia.	129
11.6.6.- Transmisión de video desde Rpicam vía radiofrecuencia.....	130
11.7.- Añadido de datos al video (Emulación por medio de la Raspberry del dispositivo OSD).....	133
11.8.- Corrección de errores	137
11.9.- Integración de Launchvid.py en el dron.....	139
11.9.1.- Integración hardware:.....	140
11.9.2.- Integración Software:	141
12. Datos GNSS.....	151
12.1.- Primera prueba: Captación de todos los mensajes provenientes de la controladora de vuelo.	152
12.2.- Obtención de sólo uno o varios tipos de mensajes.....	154
12.3.- Obtención de un único dato dentro de un tipo de mensaje.....	156
12.4.- Segunda prueba: Obtención de posicionamiento GPS.....	158
12.5.- Output de datos de la controladora como .txt	160
12.6.- Output como metadatos de imagen	161
12.7.- Integración con el “main” script (automatico.py)	166
12.7.1.- Incrustado de datos GPS como posprocesado.....	166
12.7.2.- Comando por consola.....	167
12.8.- Nombrado de fotos FLIR con datos GPS.....	174
12.8.1.- Solución de errores:.....	177
12.8.2.- Integración en el proyecto	178
13. Pruebas de vuelo.....	181
13.1.- Introducción	181
13.2.- Prueba de vuelo Rocas 1	181
13.2.1.- Resumen.....	181
13.2.2.- Proyecto al que corresponde la prueba.....	181
13.2.3.- Fecha y lugar de realización.....	181
13.2.4.- Motivo de la prueba	182
13.2.5.- Misión por realizar	182
13.2.6.- Resultados de la prueba.....	184

13.2.7.- Imágenes de archivo	184
13.2.8.- Conclusiones	185
13.2.9.- Vuelo entre los puntos 6 y 7	185
13.2.10.- Inexistencia de fotos de la primera misión	187
13.2.11.- Otros comentarios	188
13.3.- Prueba de vuelo en campo de pruebas LOCIS.....	188
13.3.1.- Resumen.....	188
13.3.2.- Proyecto al que corresponde la prueba	188
13.3.3.- Fecha y lugar de realización	188
13.3.4.- Motivo de la prueba	189
13.3.5.- Misión por realizar	189
13.3.6.- Resultados de la prueba	189
13.3.7.- Conclusiones	191
13.3.8.- Otros comentarios	191
13.4.- Prueba de vuelo Roces 2	191
13.4.1.- Resumen.....	191
13.4.2.- Proyecto al que corresponde la prueba	191
13.4.3.- Fecha y lugar de realización	191
13.4.4.- Motivo de la prueba	191
13.4.5.- Misión por realizar	191
13.4.6.- Resultados de la prueba	193
13.4.7.- Conclusiones	195
13.4.8.- Otros comentarios	196
13.5.- Prueba de vuelo transferencia TIR y RTK.....	196
13.5.1.- Resumen.....	196
13.5.2.- Proyecto al que corresponde la prueba	196
13.5.3.- Fecha y lugar de realización	196
13.5.4.- Motivo de la prueba	197
13.5.5.- Misión por realizar	197
13.5.6.- Resultados de la prueba	199
13.5.7.- Conclusiones	200
13.5.8.- Otros comentarios	200
13.6.- Prueba de vuelo Azotea TSK.....	200

13.6.1.- Resumen	200
13.6.2.- Proyecto al que corresponde la prueba.....	201
13.6.3.- Fecha y lugar de realización.....	201
13.6.4.- Motivo de la prueba	201
13.6.5.- Misión por realizar	201
13.6.6.- Resultados de la prueba.....	203
13.6.7.- Conclusiones	206
13.6.8.- Otros comentarios	206
14. UAV-Torres de tensión	208
14.1.- Selección de componentes	208
14.1.1.- Sensor electromagnético	208
15. Bibliografía:.....	217

ÍNDICE DE ILUSTRACIONES

ILUSTRACIÓN 1 ESQUEMA UAV INSPECTION. FUENTE: SERVIDOR INTERNO LOCIS SIGHTECH.	22
ILUSTRACIÓN 2 ESQUEMA TORRES DE TENSIÓN. FUENTE: ELABORACIÓN PROPIA.	22
ILUSTRACIÓN 3 PAQUETES DE TRABAJO DEL PROYECTO UAV-INSPECTION. FUENTE: ELABORACIÓN PROPIA.	23
ILUSTRACIÓN 4 PAQUETES DE TRABAJO DEL PROYECTO MONITORIZACIÓN DE TORRES DE TENSIÓN. FUENTE: ELABORACIÓN PROPIA.	24
ILUSTRACIÓN 5 AMPLIACIÓN PAQUETES DE TRABAJO. FUENTE: UNIOVI.	25
ILUSTRACIÓN 6 AMPLIACIÓN PAQUETES DE TRABAJO. FUENTE: ELABORACIÓN PROPIA.....	25
ILUSTRACIÓN 7 EJERCITO AUSTRIACO SOBRE LA CIUDAD DE VENECIA. FUENTE: ELDRONE.ES	28
ILUSTRACIÓN 8 "HEWITT SPERRY". FUENTE: ELDRONE.ES.....	29
ILUSTRACIÓN 9 MODELO ALEMÁN VERSIÓN DE "HEWITT SPERRY". FUENTE: ELDRONE.ES	29
ILUSTRACIÓN 10 GB-1 GLIDE. FUENTE: PROYECTO-ALFA.NET	30
ILUSTRACIÓN 11 "FIREBEE" DE LA COMPAÑÍA RYAN. FUENTE: WIKIPEDIA.ORG	30
ILUSTRACIÓN 12 "TOM CAT". FUENTE: ELDRONE.ES	30
ILUSTRACIÓN 13 MODELO DRON AQUILA. FUENTE: ELDRONE.ES	31
ILUSTRACIÓN 14 PREDATOR SOBRE AFGANISTÁN. FUENTE: GOOGLE.	31
ILUSTRACIÓN 15 PRIMER USO DEL "PREDATOR", VISTA FPV. FUENTE: ELDRONE.ES.....	31
ILUSTRACIÓN 16 DRON DE VIGILANCIA EN CARRETERA. FUENTE: HTTP://SORRO.ES/DRONES-POLICIALES-PARA-VIGILANCIA-DEL-TRAFICO/ ACCESO: 20/4/2018.	32
ILUSTRACIÓN 17 DRON PARA ENTRETENIMIENTO. FUENTE: HTTPS://WWW.MEDIATRENDS.ES/A/53050/DRONES-PARA-NINOS/ ACCESO: 20/4/2018.....	33
ILUSTRACIÓN 18 DRON PARA TELEDETECCIÓN AGRÍCOLA. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=DRON+PARA+TELEDETECCION+AGRICOLA&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKewiMrBMMYTlAAHWIAXQKH8UB8GQ_AUICIGB&BIW=1440&BIH=794#imgrc=H8Ak_HBcEk9LM M: ACCESO: 24/4/2018.....	33
ILUSTRACIÓN 19 AQUILA DE FACEBOOK.FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=DRON+AQUILA+FACEBOOK&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKewib04UL1NLAHVJBHQKH8UB8GQ_AUICIGB&BIW=1440&BIH=794#imgrc=WUX4PEdof8LR4M R4M: ACCESO: 24/4/2018.....	34

ILUSTRACIÓN 20 DRONES REPARTIDORES. FUENTE: HTTPS://WWW.ELDIARIO.ES/HOJADEROUTER/DRONES/PLANES-DELIRANTES-AMAZON-DRONES-REPARTIDORES_0_700631063.HTML ACCESO: 2474/2018.	35
ILUSTRACIÓN 21 CLASIFICACIÓN SEGÚN EL MÉTODO DE SUSTENTACIÓN. FUENTE: HTTP://WWW.XDRONES.ES/TIPOS-DE-DRONES-CLASIFICACION-DE-DRONES-CATEGORIAS-DE-DRONES/ ACCESO: 25/4/2918.	37
ILUSTRACIÓN 22 CLASIFICACIÓN DE MULTI-ROTORES. FUENTE: HTTP://WWW.XDRONES.ES/TIPOS-DE-DRONES-CLASIFICACION-DE-DRONES-CATEGORIAS-DE-DRONES/ ACCESO: 25/4/2018.....	37
ILUSTRACIÓN 23 PIXHAWCK PX4. : ACCESO: 24/4/2018. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=PIXHAWK+PX4&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKEWJXHU3G49LAAHXDUHQKHcMLAJCQ_AUICIGB&BIW=805&BIH=720#IMGRC=BD0T5AHOHQHX1M	38
ILUSTRACIÓN 24 RASPBERRY PI3 MODELO B. ACCESO: 24/4/2018. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=RASPBERRY+PI+3+MODEL+B&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKEWJUGBOO5NLAHVMMWHQKHSGoDF4Q_AUICYGC&BIW=805&BIH=720#IMGDII=ARRxXOJPH9JOM:&IMGRC=Bss7Q5KuQNXSUM:	39
ILUSTRACIÓN 25 TARANIS-E X9E. ACCESO: 24/4/2018. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?BIW=805&BIH=720&TBM=ISCH&SA=1&EI=NHLFWSTPNYBfKGXxwzE&Q=TARANIS+E+X9E&OQ=TARANIS+E+X9E&GS_L=PSY-AB.3...18311.27861.0.28628.17.15.2.0.0.0.151.1586.0j14.14.0...0...1c.1.64.PSY-AB..1.6.558...0j0i19k1j0i8l	39
ILUSTRACIÓN 26 SEGUIMIENTO DE UNA MISIÓN DE VUELO DESDE LA ESTACIÓN DE TIERRA USANDO EL SOFTWARE "MISSION PLANNER". ACCESO: 24/4/2018.FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=MISSION+PLANNER&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKEWJ5O4ER59LAAHUCHVsqBRsQ_AUICYGC&BIW=805&BIH=720#IMGRC=4P9oZT9s-VpPRM	40
ILUSTRACIÓN 27 "RASPBERRY PI CAMERA MODULE V2". ACCESO: 24/4/2018.FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=RASPBERRY+PI+CAMERA+MODULE+V2&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKEWIZ9US14NLAHVHSHQKHf1FA6AQ_AUIDCGD&BIW=1440&BIH=794#IMGRC=zQ-VyVBJQfKAdM:	41
ILUSTRACIÓN 28 CÁMARA FLIR VUE PRO. ACCESO: 24/4/2918. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?BIW=805&BIH=720&TBM=ISCH&SA=1&EI=9BTfWsjSEckUkgW59I_oCW&Q=FLIR+VUE+PRO+&OQ=FLIR+VUE+PRO+&GS_L=PSY-AB.3...0j0i30k1L2j0i24k1L7.169984.171987.0.172170.13.10.0.2.2.0.220.976.0j6j1.7.0....0...1c.1.64.PSY-AB..4.9.986...0i67k1j0i10k1.0.CfKTpFXoAC4#IMGRC=UEEGXgDbrvc2KM:	41
ILUSTRACIÓN 29 GNSS ·DR . ACCESO: 24/4/2018.FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?BIW=805&BIH=720&TBM=ISCH&SA=1&EI=ThffWsiXKc6zkWxr9QMADA&Q=3+DR+GPS+&OQ=3+DR+GPS+&GS_L=PSY-AB.3...3846.7882.0.7993.19.16.2.0.0.0.187.1205.0j9.10.0....0...1c.1.64.PSY-AB..8.9.882.0..0j0i10k1j0i30k1j0i67k1j0i	42
ILUSTRACIÓN 30 CHASIS HMF U580 OPRO QUAD. ACCESO: 2474/2018.FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=HMF+U580+QUAD&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKEWIAKDVP-TLAAHWGSHQKHWWzDEWQ_AUICYGC&BIW=805&BIH=720#IMGRC=HN18SRX5IT1NCM:	42
ILUSTRACIÓN 31 HÉLICES T-MOTOR V2. FUENTE: HTTPS://WWW.MULTICOPTERO.COM/ES/TIENDA-ON-LINE/HELICES/T-MOTOR-10X3-3-PAREJA-V2/ ACCESO: 01/08/2018.	43
ILUSTRACIÓN 32 NAVIGATOR MN3508. ACCESO: 01/08/2018. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?BIW=805&BIH=720&TBM=ISCH&SA=1&EI=ZSNfWTGXDYEUsAfBTQWgCW&Q=Tmotor+Navigator+MN3508+&OQ=Tmotor+Navigator+MN3508+&GS_L=PSY-AB.3...283395.293194.0.294192.24.24.0.0.0.157.2536.3j20.23.0....0...1c.1.64.PSY-AB..1.7.845...0j0i67k1j0i10k1j0i30k1j0i10i30k1j0i10i24k1j0i13k1j0i13i30k1j0i8i13i30k1.0.Ezai1uTnB8k#IMGRC=KcQ6Y4NX1TXNJM:	43

ILUSTRACIÓN 33 TMOTOR AIR. ACCESO: 01/08/2018. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=TMOTOR+AIR+20A&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKEWJLINV P_dLAAHWM1HQKHSksCH4Q_AUICYGC&BIW=805&BIH=722#IMGRC=XRQFL3QUEYEKGM:	44
ILUSTRACIÓN 34 BATERÍA. FUENTE: HTTPS://RC-INNOVATIONS.ES/LIPO-BATERIA-4S-8000MAH-14.8V-30C ACCESO: 25/4/2018.	44
ILUSTRACIÓN 35 CONEXIONES EXISTENTES. FUENTE: SERVIDOR LOCIS SIGTECH.	45
ILUSTRACIÓN 36 CONEXIONES EXISTENTES. FUENTE: SERVIDOR LOCIS SIGTECH.	45
ILUSTRACIÓN 37 ESQUEMA DE UN LED. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=PARTES+DE+UN+LED+CATODO+ANODO&TBM=I SCH&SOURCE=IU&ICTX=1&FIR=MRJ9VJMkuW_LQM%253A%252CXQGG4GPCH2LX1M%25 2C_&USG=__RYX_ATBV2jFDRG88CNzXPBoR5PY%3D&SA=X&VED=0AHUKEWJR0r2ZzJOD AAHVNSKQKHZDXDCQQ9QEIKJAA#IM	47
ILUSTRACIÓN 38 ESQUEMA DE CONEXIÓN FÍSICA DE RPI CON LED. FUENTE: ELABORACIÓN PROPIA.	48
ILUSTRACIÓN 39 MODOS DE LOS PINES DE LA RASPBERRY. FUENTE: ELABORACIÓN PROPIA.	49
ILUSTRACIÓN 40 FUNCIÓN ENCARGADA DE ENCENDER EL LED. FUENTE: ELABORACIÓN PROPIA.	49
ILUSTRACIÓN 41 LOCALIZACION_FOTOS.PY PARTE 1. FUENTE: ELABORACIÓN PROPIA.	50
ILUSTRACIÓN 42 LOCALIZACION_FOTOS.PY PARTE 1. FUENTE: ELABORACIÓN PROPIA.	50
ILUSTRACIÓN 43 NOTACIÓN DEL FICHERO GENERADO. FUENTE: ELABORACIÓN PROPIA.	51
ILUSTRACIÓN 44 INFORME GENERADO. FUENTE: ELABORACIÓN PROPIA.	51
ILUSTRACIÓN 45 FOTOS TÉRMICAS EN LA CARPETA COMPARTIDA. FUENTE: ELABORACIÓN PROPIA.....	51
ILUSTRACIÓN 46 SINCRONIZACIÓN MISIÓN DE VUELO. FUENTE: ELABORACIÓN PROPIA.....	52
ILUSTRACIÓN 47 CONEXIÓN CONTROLADORA RASPBERRY VÍA USB. FUENTE: HTTPS://VUELOARTIFICIAL.COM/1-CONEXION- VUELO-CONTROLADO-MEDIANTE-RASPBERRY-PI/	53
ILUSTRACIÓN 48 PINES GPIO DE LA RASPBERRY PI 3. FUENTE: HTTPS://WWW.PROGRAMOERGOSUM.COM/CURSOS- ONLINE/RASPBERRY-PI/238-CONTROL-DE-GPIO-CON-PYTHON-EN-RASPBERRY-PI/INTERMITENTE	54
ILUSTRACIÓN 49 ESQUEMA DE CONEXIONES RASPBERRY PI-PIXHAWCK. FUENTE: ELABORACIÓN PROPIA.....	54
ILUSTRACIÓN 50 ALIMENTACIÓN DE LA RASPBERRY PI VÍA MINI-USB. FUENTE: HTTPS://WWW.PCWORLD.COM/ARTICLE/2598363/HOW-TO-SET-UP-RASPBERRY-PI-THE-LITTLE-COMPUTER-YOU-CAN- COOK-INTO-DIY-TECH-PROJECTS.HTML	55
ILUSTRACIÓN 51 ALIMENTACIÓN DE LA RASPBERRY PI USANDO LA CONTROLADORA. FUENTE: ELABORACIÓN PROPIA.	55
ILUSTRACIÓN 52 LINUX "BOOT PROCESS". FUENTE: ELABORACIÓN PROPIA.	56
ILUSTRACIÓN 53 LOGO DE RASPBIAN. FUENTE: GOOGLE.....	57
ILUSTRACIÓN 54 FICHERO /ETC/RC.LOCAL FUENTE: ELABORACIÓN PROPIA.	58
ILUSTRACIÓN 55 EJEMPLO DE CONSOLA DE COMANDOS BASH. FUENTE: GOOGLE.	59
ILUSTRACIÓN 56 STARTUP.SH LLAMADO DESDE /ETC/RC.LOCAL. FUENTE: ELABORACIÓN PROPIA.....	59
ILUSTRACIÓN 57 CONVERTIR EL FICHERO EN EJECUTABLE. FUENTE: ELABORACIÓN PROPIA.	59
ILUSTRACIÓN 58 LISTADO DE ARCHIVOS EN EL DIRECTORIO /HOME/PI DE LA RPI. FUENTE: ELABORACIÓN PROPIA.....	60
ILUSTRACIÓN 59 LISTA DE PARÁMETROS VISTO DESDE LA PLANIFICADORA DE VUELO (MISSION PLANNER). FUENTE: ELABORACIÓN PROPIA.	61
ILUSTRACIÓN 60 RESULTADO DEL COMANDO "PARAM SHOW" EN LA TERMINAL MULTIPLEXADA EN LA QUE SE EJECUTA MAVPROXY.PY. FUENTE: ELABORACIÓN PROPIA.	62
ILUSTRACIÓN 61 INTERACCIÓN HUMANO-MÁQUINA. FUENTE: ELABORACIÓN PROPIA.	63
ILUSTRACIÓN 62 ESTRUCTURA DE UN PAQUETE MAVLINK. FUENTE: ELABORACIÓN PROPIA.....	64
ILUSTRACIÓN 63 INSTALACIÓN DE MAVPROXY.PY DENTRO DE LA RASPBERRY. FUENTE: ELABORACIÓN PROPIA.	64
ILUSTRACIÓN 64 INSTALACIÓN DEL SOFTWARE SCREEN EN LA RASPBERRY PI. FUENTE: ELABORACIÓN PROPIA.	64
ILUSTRACIÓN 65 COMANDO SCREEN -LS CON MULTIPLEXACIONES ACTIVAS. FUENTE: ELABORACIÓN PROPIA.....	65
ILUSTRACIÓN 66 COMANDO SCREEN -X POR CONSOLA. FUENTE: ELABORACIÓN PROPIA.	65
ILUSTRACIÓN 67 MULTIPLEXACIÓN DE TERMINAL. FUENTE: ELABORACIÓN PROPIA.....	65

ILUSTRACIÓN 68 USUARIO PI VS USUARIO ROOT. FUENTE: ELABORACIÓN PROPIA.	66
ILUSTRACIÓN 69 RPI_TO_PIX_XXX.PY .FUENTE: ELABORACIÓN PROPIA.	66
ILUSTRACIÓN 70 CÓDIGO DE "PASAR_WAYS.PY". FUENTE: ELABORACIÓN PROPIA.	67
ILUSTRACIÓN 71 EJEMPLO DE ARCHIVO .WAYPOINTS. FUENTE: SERVIDOR INTERNO LOCIS.	68
ILUSTRACIÓN 72 FORMATO ARCHIVO .WAYPOINTS. FUENTE: ELABORACIÓN PROPIA.	68
ILUSTRACIÓN 73 BORRADO DEL ARCHIVO XXX.WAYPOINTS. FUENTE: ELABORACIÓN PROPIA.	69
ILUSTRACIÓN 74 ENVIÓ DE WAYPOINTS POR CONSOLA DE COMANDOS. FUENTE: ELABORACIÓN PROPIA.	69
ILUSTRACIÓN 75 COMPROBACIÓN DE LA MISIÓN RECIBIDA EN MISSION PLANNER. FUENTE: ELABORACIÓN PROPIA.	70
ILUSTRACIÓN 76 COMPROBACIÓN DE LA MISIÓN RECIBIDA EN MISSION PLANNER. FUENTE: ELABORACIÓN PROPIA.	71
ILUSTRACIÓN 77 ESQUEMA DE DECISIÓN DEL MECANISMO DE SEGURIDAD. FUENTE: ELABORACIÓN PROPIA.	72
ILUSTRACIÓN 78 ACTUALIZACIÓN DEL PROGRAMA PYTHON PARTE 1. FUENTE: ELABORACIÓN PROPIA.....	72
ILUSTRACIÓN 79 ACTUALIZACIÓN DEL PROGRAMA PYTHON PARTE 2. FUENTE: ELABORACIÓN PROPIA.....	72
ILUSTRACIÓN 80 ACTUALIZACIÓN DEL PROGRAMA PYTHON PARTE 3. FUENTE: ELABORACIÓN PROPIA.....	73
ILUSTRACIÓN 81 ABREVIATURAS DE LOS MESES EN INGLÉS. FUENTE: HTTPS://WWW.APRENDER.ORG/MESES-EN-INGLES/ ACCESO: 24/4/2018.....	74
ILUSTRACIÓN 82 CAMBIO DE PROPIETARIO Y ARCHIVO EJECUTABLE. FUENTE: ELABORACIÓN PROPIA.....	75
ILUSTRACIÓN 83 ERROR AL INTENTAR ESCRIBIR EN EL ARCHIVO CREADO. FUENTE: ELABORACIÓN PROPIA.	75
ILUSTRACIÓN 84 ARCHIVO .TXT EN LA CARPETA COMPARTIDA DE LA RASPBERRY QUE INDICA QUE NO SE ENCUENTRA MISIÓN CONCORDANTE CON EL DÍA ACTUAL. FUENTE ELABORACIÓN PROPIA.	76
ILUSTRACIÓN 85 PRUEBA_NO.PY EN EL DIRECTORIO /HOME/PI LLAMADO DESDE EL NO LÓGICO. FUENTE: ELABORACIÓN PROPIA.	76
ILUSTRACIÓN 86 NO CARGA DE LA MISIÓN. FUENTE: ELABORACIÓN PROPIA.....	76
ILUSTRACIÓN 87 INFORME GENERADO EN LA CARPETA COMPARTIDA DE LA RASPBERRY. FUENTE: ELABORACIÓN PROPIA.	77
ILUSTRACIÓN 88 MISIÓN VISTA EN LA PLANIFICADORA DE VUELO "MISSION PLANNER". FUENTE: ELABORACIÓN PROPIA.	77
ILUSTRACIÓN 89 ARCHIVO .WAYPOINTS CON FECHA ACTUAL. FUENTE ELABORACIÓN PROPIA.	77
ILUSTRACIÓN 90 PRUEBASÍ.PY LLAMADO DESDE EL SÍ LÓGICO. FUENTE: ELABORACIÓN PROPIA.	77
ILUSTRACIÓN 91 CARGA DE UNA MISIÓN DIFERENTE ANTES DE REALIZAR LA PRUEBA Y POSTERIOR REINICIO DE LA RASPBERRY. FUENTE: ELABORACIÓN PROPIA.	78
ILUSTRACIÓN 92 INFORME GENERADO AL INICIO DE LA RASPBERRY. FUENTE: ELABORACIÓN PROPIA.....	79
ILUSTRACIÓN 93 ARCHIVO .TXT DE LA CARPETA COMPARTIDA VS MISIÓN CARGADA EN LA CONTROLADORA. FUENTE ELABORACIÓN PROPIA.	79
ILUSTRACIÓN 94 PARTES DEL SISTEMA RTK. FUENTE: ELABORACIÓN PROPIA.	81
ILUSTRACIÓN 95 ESTADO DE LA SOLUCIÓN. FUENTE: DOCUMENTACIÓN DE EMLID, HTTPS://DOCS.EMLID.COM/REACH/COMMON/REACHVIEW/STATUS/ ACCESO: 01/06/2018.....	82
ILUSTRACIÓN 96 ESQUEMA DE CONEXIONADO FÍSICO RTK. FUENTE: ELABORACIÓN PROPIA.	83
ILUSTRACIÓN 97 "REACHVIEW" VISTO DESDE UN NAVEGADOR WEB. FUENTE: DOCUMENTOS DE EMLID.....	84
ILUSTRACIÓN 98 DISPOSITIVOS CONECTADOS A LA RED DE LA OFICINA DE LOCIS. FUENTE: ELABORACIÓN PROPIA.	85
ILUSTRACIÓN 99 REACHVIEW MÓDULO BASE. FUENTE: ELABORACIÓN PROPIA.....	85
ILUSTRACIÓN 100 PANTALLA EJEMPLO DE LA INTERFAZ REACHVIEW. FUENTE: DOCUMENTACIÓN DE EMLID.	86
ILUSTRACIÓN 101 CREACIÓN DE NUEVA CONEXIÓN. FUENTE: ELABORACIÓN PROPIA.	87
ILUSTRACIÓN 102 NO CORRECCIONES (NO RECEPCIÓN DE COMUNICACIÓN SATELITAL). FUENTE: ELABORACIÓN PROPIA.....	88
ILUSTRACIÓN 103 ERROR AL INTENTAR "GPS INJECT". FUENTE: SERVIDOR INTERNO LOCIS.....	89
ILUSTRACIÓN 104 MÓDULO ROVER DEL SISTEMA RTK, EN MODO "SINGLE". FUENTE: SERVIDOR INTERNO LOCIS.	91
ILUSTRACIÓN 105 VISTA DE DETALLE AL TIPO DE SATELITE. FUENTE: SERVIDOR INTERNO LOCIS.	91
ILUSTRACIÓN 106 MENSAJE DE REACHVIEW DEL MÓDULO BASE. FUENTE: ELABORACIÓN PROPIA.	92
ILUSTRACIÓN 107 SISTEMA RTK CON SOLUCIÓN SINGLE, CON LA BASE CAPTANDO SATELITES. FUENTE: SERVIDOR INTERNO LOCIS.....	93

ILUSTRACIÓN 108 SISTEMA RTK CON SOLUCIÓN FLOAT, RECIBIENDO CORRECCIONES DE LA BASE. FUENTE: SERVIDOR INTERNO LOCIS.	93
ILUSTRACIÓN 109 ESQUEMA PUERTOS DE CONEXIÓN DE LIDAR. FUENTE: STATICGARMIN.COM ACCESO: 3/5/2018.	96
ILUSTRACIÓN 110 DIAGRAMA DE CONEXIONES I2C. FUENTE: HTTPS://MOBIUSSTRIPBLOG.WORDPRESS.COM/2016/12/26/FIRST-BLOG-POST/ ACCESO: 2/5/2018.	97
ILUSTRACIÓN 111 ESQUEMA DE CONEXIONES LIDAR- RASPBERRY. FUENTE: ELABORACIÓN PROPIA.	98
ILUSTRACIÓN 112 RESULTADO DE HABILITAR EN LA RASPBERRY I2C. FUENTE: ELABORACIÓN PROPIA.	99
ILUSTRACIÓN 113 OUTPUT CORRESPONDIENTE COMO INTERFAZ HABILITADA. FUENTE: ELABORACIÓN PROPIA.....	99
ILUSTRACIÓN 114 COMANDO I2CDETECT HACIENDO REFERENCIA AL PERIFÉRICO (LIDAR) FUENTE: ELABORACIÓN PROPIA.	99
ILUSTRACIÓN 115 ACTUALIZACIONES CORRESPONDIENTES. FUENTE: ELABORACIÓN PROPIA.	99
ILUSTRACIÓN 116 ESQUEMA TEÓRICO DE CONEXIONES PWM LIDAR Y RASPBERRY PI. FUENTE: ELABORACIÓN PROPIA.....	100
ILUSTRACIÓN 117 SCRIPT PARA TOMA DE DATOS POR LIDAR. FUENTE: ELABORACIÓN PROPIA	101
ILUSTRACIÓN 118 ESQUEMA TEÓRICO DE LAS CONEXIONES PARA LA PLACA ARDUINO UNO. FUENTE: HTTP://STATIC.GARMIN.COM/PUMAC/LIDAR_LITE_V3_OPERATION_MANUAL_AND_TECHNICAL_SPECIFICATIONS.PDF	102
ILUSTRACIÓN 119 MONTAJE DEL DISPOSITIVO LIDAR EN LA PLACA ARDUINO. FUENTE: ELABORACIÓN PROPIA.	103
ILUSTRACIÓN 120 MEDICIÓN DE LA ALTURA AL TECHO EN CM USANDO EL LIDAR VS MEDICIÓN PONIENDO LA MANO. FUENTE: ELABORACIÓN PROPIA.	104
ILUSTRACIÓN 121 CONEXIÓN I2C A RASPBERRY PI. FUENTE: ELABORACIÓN PROPIA.	105
ILUSTRACIÓN 122 CÓDIGO DE LIDAR_I2C.PY FUENTE: ELABORACIÓN PROPIA.	106
ILUSTRACIÓN 123 ERROR AL EJECUTAR EL PROGRAMA. FUENTE: ELABORACIÓN PROPIA.	106
ILUSTRACIÓN 124 EJECUCIÓN DEL PROGRAMA. FUENTE: ELABORACIÓN PROPIA.	107
ILUSTRACIÓN 125 VERSIÓN ACTUAL DEL KERNEL. FUENTE: ELABORACIÓN PROPIA.....	107
ILUSTRACIÓN 126 CONTROLADOR DEL BUS I2C ACTUAL, VISTO CON EL COMANDO "LSMOD". FUENTE: ELABORACIÓN PROPIA.	108
ILUSTRACIÓN 127 DESCARGA DEL ARCHIVO CORRESPONDIENTE (EL LINK DE DESCARGA SE ENCUENTRA EN LA BIBLIOGRAFÍA). FUENTE: ELABORACIÓN PROPIA.	108
ILUSTRACIÓN 128 ARCHIVO EN LA RUTA /BOOT/OVERLAYS. FUENTE: ELABORACIÓN PROPIA.	108
ILUSTRACIÓN 129 FICHERO EDITADO EN /BOOT/CONFIG.TXT. FUENTE: ELABORACIÓN PROPIA.	108
ILUSTRACIÓN 130 FICHERO EDITADO EN /ETC/MODULES. FUENTE: ELABORACIÓN PROPIA.	109
ILUSTRACIÓN 131 LOADED OVERLAY DE FORMA CORRECTA. FUENTE: ELABORACIÓN PROPIA.	109
ILUSTRACIÓN 132 ALIMENTACIÓN CON FUENTE EXTERNA. FUENTE: ELABORACIÓN PROPIA.	110
ILUSTRACIÓN 133 VERSIÓN CORRECTA. FUENTE: GITHUB.	111
ILUSTRACIÓN 134 COMANDO POR CONSOLA PARA "DOWN GRADE" DEL "KERNEL". FUENTE: ELABORACIÓN PROPIA.	111
ILUSTRACIÓN 135 DOWN GRADE DE LA VERSIÓN DEL KERNEL EXITOSA. FUENTE: ELABORACIÓN PROPIA.	111
ILUSTRACIÓN 136 NUEVA VERSIÓN DEL KERNEL. FUENTE: ELABORACIÓN PROPIA.	111
ILUSTRACIÓN 137 ESQUEMA TEÓRICO DE CONEXIONES PWM LIDAR Y RASPBERRY PI (USANDO LA PLACA ARDUINO COMO FUENTE DE ALIMENTACIÓN EXTERNA). FUENTE: ELABORACIÓN PROPIA.	112
ILUSTRACIÓN 138 ESQUEMA DE CONEXIÓN REAL DE LIDAR Y RASPBERRY PI (USANDO LA PLACA ARDUINO COMO FUENTE DE ALIMENTACIÓN EXTERNA). FUENTE: ELABORACIÓN PROPIA.	113
ILUSTRACIÓN 139 SCRIPT Y REPETICIÓN DE LA PRUEBA CON ARDUINO FUENTE: ELABORACIÓN PROPIA.	113
ILUSTRACIÓN 140 MONTAJE REAL CABLEADO LIDAR PWM CON RASPBERRY. FUENTE: ELABORACIÓN PROPIA.....	114
ILUSTRACIÓN 141 EXPORTACIÓN DE DATOS DE ALTURA A LA CARPETA COMPARTIDA. FUENTE: ELABORACIÓN PROPIA.	115
ILUSTRACIÓN 142 OUTPUT EN FORMA DE .TXT PARA EL LIDAR, SIGUIENDO EL ESQUEMA ALTURA REGISTRA EN UNA MARCA TEMPORAL ESPECÍFICA. FUENTE: ELABORACIÓN PROPIA.	116
ILUSTRACIÓN 143 ACTUACIÓN DEL DISPOSITIVO. FUENTE: HTTPS://STATIC.GARMIN.COM/PUMAC/LIDAR_LITE_V3_OPERATION_MANUAL_AND_TECHNICAL_SPECIFICATIONS.PDF ACCESO: 8/5/2018.....	116
ILUSTRACIÓN 144 ESQUEMA DE CONEXIONES. FUENTE: ELABORACIÓN PROPIA.	117

ILUSTRACIÓN 145 TX DE RC INNOVATIONS. FUENTE: HTTPS://RC-INNOVATIONS.ES/IMMERSIONRC-TRANSMISOR-FPV-5.8GHZ-600MW-RACE-BAND	118
ILUSTRACIÓN 146 TX DE RC INNOVATIONS (PARTE TRASERA). FUENTE: ELABORACIÓN PROPIA.....	118
ILUSTRACIÓN 147 CONECTOR JST CON CABLE. FUENTE: RCINNOVATIONS ACCESO:17/5/2018.	119
ILUSTRACIÓN 148 TABLA DE FRECUENCIAS DEL DISPOSITIVO. FUENTE: ELABORACIÓN PROPIA.	120
ILUSTRACIÓN 149 CONEXIONES DISPOSITIVO OSD FUENTE: HTTPS://OSCARLIANG.COM/FPV-GUIDE/ FUENTE: HTTP://ARDUPILOT.ORG/COPTER/DOCS/COMMON-MINIM-OSD-QUICK-INSTALLATION-GUIDE.HTML	121
ILUSTRACIÓN 150 SUDO RASPI-CONFIG. FUENTE: ELABORACIÓN PROPIA.....	122
ILUSTRACIÓN 151 ARCHIVO DE VIDEO GENERADO. FUENTE: ELABORACIÓN PROPIA.....	123
ILUSTRACIÓN 152 COMANDO POR CONSOLA PARA INICIAR "LIVE". FUENTE: ELABORACIÓN PROPIA.....	123
ILUSTRACIÓN 153 CAMBIOS A REALIZAR EN REPRODUCTOR VLC. FUENTE: ELABORACIÓN PROPIA.....	124
ILUSTRACIÓN 154 OUTPUT COMO "LIVE" VIDEO. FUENTE: ELABORACIÓN PROPIA.....	124
ILUSTRACIÓN 155 VISTA DETALLE "EDIT OPTION". FUENTE: ELABORACIÓN PROPIA.	124
ILUSTRACIÓN 156 CONECTOR JACK EN LA RASPBERRY PI. FUENTE: HTTPS://ES.GEARBEST.COM/RASPBERRY-PI/PP_488334.HTML	125
ILUSTRACIÓN 157 DIAGRAMA DE UN CABLE DESTINADO A UN PUERTO JACK DE 3.5MM Y UN CABLE ADAPTADOR RCA. FUENTE: HTTPS://DESCUBRIENDOLAORANGEPI.WORDPRESS.COM/2017/01/10/SALIDAS-DE-AUDIO-Y-VIDEO-HDMI-Y-RCA/ ACCESO: 21/05/18.	125
ILUSTRACIÓN 158 DIAGRAMA PUERTO JACK TRRS (TIP, RING, RING, SLEEVE) EN RASPBERRY PI 3. FUENTE: HTTPS://WWW.RASPBERRYPI-SPY.CO.UK/2014/07/RASPBERRY-PI-MODEL-B-3-5MM-AUDIOVIDEO-JACK/ .ACCESO: 21/5/2018.	125
ILUSTRACIÓN 159 TEST POINTS RASPBERRY. FUENTE: HTTP://ANDIDINATA.COM/2017/02/HACK-IT-NO-AV-CABLE-NO-PROBLEM/ . ACCESO: 21/5/2018.	126
ILUSTRACIÓN 160 MODO SDTV. FUENTE: HTTPS://WWW.RASPBERRYPI.ORG/DOCUMENTATION/CONFIGURATION/CONFIG-TXT/VIDEO.MD	127
ILUSTRACIÓN 161 ASPECTO DE SDTV FUENTE: HTTPS://WWW.RASPBERRYPI.ORG/DOCUMENTATION/CONFIGURATION/CONFIG-TXT/VIDEO.MD	127
ILUSTRACIÓN 162 SDTV EN /BOOT/CONFIG.TXT. FUENTE: ELABORACIÓN PROPIA.	127
ILUSTRACIÓN 163 MONTAJE DE LA PANTALLA. FUENTE: ELABORACIÓN PROPIA.	128
ILUSTRACIÓN 164 DESKTOP DE LA RASPBERRY PI DESDE LA PANTALLA EXTERNA. FUENTE: ELABORACIÓN PROPIA.	128
ILUSTRACIÓN 165 IMAGEN DE VIDEO PROCEDENTE DE LA "RPICAM". FUENTE: ELABORACIÓN PROPIA.	128
ILUSTRACIÓN 166 DIFERENTES ESTÁNDARES DE JACK TRSS (EN FUNCIÓN DEL DIFERENTE FABRICANTE). FUENTE: HTTPS://WWW.CABLECHICK.COM.AU/BLOG/UNDERSTANDING-TRRS-AND-AUDIO-JACKS/ ACCESO: 21/5/2018.	129
ILUSTRACIÓN 167 PRUEBA DE FUNCIONAMIENTO DE LA EMISORA DE RADIOFRECUENCIA. FUENTE: ELABORACIÓN PROPIA.	129
ILUSTRACIÓN 168 ESQUEMA TEÓRICO DE CONEXIONES. FUENTE: ELABORACIÓN PROPIA.	130
ILUSTRACIÓN 169 MONTAJE EXPERIMENTAL. FUENTE: ELABORACIÓN PROPIA.	131
ILUSTRACIÓN 170 CONTACTO ENTRE EL PIN DEDICADO A VIDEO EN EL JACK TRRS DE LA RASPBERRY Y LA TX. FUENTE: ELABORACIÓN PROPIA.	132
ILUSTRACIÓN 171 VISIÓN POR PANTALLA DE LA RPICAM. FUENTE: ELABORACIÓN PROPIA.	132
ILUSTRACIÓN 172 SALIDAS DE VIDEO VÍA "TEST PADS" PP24 O VIDEO PIN MARCADO. FUENTE: ELABORACIÓN PROPIA.	133
ILUSTRACIÓN 173 SCRIPT PYTHON PARA AÑADIR FECHA AL "LIVE" VIDEO. FUENTE: ELABORACIÓN PROPIA.	133
ILUSTRACIÓN 174 TIMESTAMP VIDEO RPICAM. FUENTE: ELABORACIÓN PROPIA.	133
ILUSTRACIÓN 175 SCRIPT PARA AÑADIR FECHA Y GPS AL VIDEO "LIVE". FUENTE: ELABORACIÓN PROPIA.	134
ILUSTRACIÓN 176 GPS "OVERLAY" EN VIDEO RPICAM. FUENTE: ELABORACIÓN PROPIA.....	134
ILUSTRACIÓN 177 ACTUALIZACIÓN EN EL SCRIPT. FUENTE: ELABORACIÓN PROPIA.....	¡ERROR! MARCADOR NO DEFINIDO.
ILUSTRACIÓN 178 "OVERLAY" GPS Y TIMESTAMP. FUENTE: ELABORACIÓN PROPIA.....	¡ERROR! MARCADOR NO DEFINIDO.
ILUSTRACIÓN 179 INSTALACIÓN DE UTILIDADES REFERENTES A LA MULTIPLEXACIÓN "SCREEN". FUENTE: ELABORACIÓN PROPIA.	135

ILUSTRACIÓN 180 CAMBIO DE DERECHOS DE LA MULTIPLEXADORA GNU SCREEN. FUENTE: ELABORACIÓN PROPIA.	135
ILUSTRACIÓN 181 LAUNCHVID.PY . FUENTE: ELABORACIÓN PROPIA.	135
ILUSTRACIÓN 182 "OVERLAY" GPS TIMESTAMP Y ALTURA. FUENTE: ELABORACIÓN PROPIA.	137
ILUSTRACIÓN 183 EJEMPLO DE LECTURA CORTADA. FUENTE: ELABORACIÓN PROPIA.	137
ILUSTRACIÓN 184 "LIST INDEX OUT OF RANGE". FUENTE: ELABORACIÓN PROPIA.	137
ILUSTRACIÓN 185 MODO DE APERTURA DE FICHERO. FUENTE: HTTP://WWW.PYTHONFORBEGINNERS.COM/FILES/READING-AND-WRITING-FILES-IN-PYTHON	138
ILUSTRACIÓN 186 ACTUALIZACIÓN DE LA FUNCIÓN. FUENTE: ELABORACIÓN PROPIA.	138
ILUSTRACIÓN 187 "RPIVIDEO" CON CORRECCIÓN DE ERRORES. FUENTE: ELABORACIÓN PROPIA.	138
ILUSTRACIÓN 188 EJEMPLO DE ERROR DE CLIENTE ÚNICO. FUENTE: HTTPS://RASPBERRYPI.STACKEXCHANGE.COM/QUESTIONS/26829/PICAMERA-NOT-WORKING	139
ILUSTRACIÓN 189 DISPARO DE FOTO MIENTRAS SE PRODUCE LA GRABACIÓN. FUENTE: ELABORACIÓN PROPIA.	139
ILUSTRACIÓN 190 TOMA DE FOTOS DURANTE LA EMISIÓN DE VIDEO. FUENTE: ELABORACIÓN PROPIA.	139
ILUSTRACIÓN 191 BATERÍA QUE ALIMENTA LA RADIO. FUENTE: HTTPS://WWW.BANGGOOD.COM/ES/ZOP-POWER-7_4V-2200MAH-2S-35C-LIPO-BATTERY-T-PLUG-P-992114.HTML?GMCOUNTRY=ES&CURRENCY=EUR&utm_source=GOOGLESHOPPING&utm_medium=CPC_ODS&utm_content=HEATH&utm_campaign=PLA-HELI-ES-PC&GCL	140
ILUSTRACIÓN 192 ALGUNOS DE LOS "TEST PADS" REFERENTES A MASA. FUENTE: ELABORACIÓN PROPIA.	141
ILUSTRACIÓN 193 EJECUCIÓN DE PROGRAMA PYTHON PARA "MULTIPLEXAR" EL LANZAMIENTO DE VIDEO. FUENTE: ELABORACIÓN PROPIA.	142
ILUSTRACIÓN 194 COMANDO QUE EJECUTA LA MULTIPLEXACIÓN DE VID. FUENTE: ELABORACIÓN PROPIA.	142
ILUSTRACIÓN 195 TERMINALES MULTIPLEXADAS. FUENTE: ELABORACIÓN PROPIA.	142
ILUSTRACIÓN 196 "RE-ATTACH" A UNA TERMINAL MULTIPLEXADA. FUENTE: ELABORACIÓN PROPIA.	143
ILUSTRACIÓN 197 CONEXIÓN CON LA CONTROLADORA MEDIANTE PUERTO SERIAL TTYAMA0. FUENTE: ELABORACIÓN PROPIA.	143
ILUSTRACIÓN 198 OTORGAMOS PRIVILEGIOS SOBRE EL ARCHIVO .TXT (USADO PARA ALMACENAR DATOS) A TODOS LOS USUARIOS. FUENTE: ELABORACIÓN PROPIA.	144
ILUSTRACIÓN 199 CAMBIO DE DUEÑO DEL ARCHIVO. FUENTE: ELABORACIÓN PROPIA.	144
ILUSTRACIÓN 200 VOLCADO DE DATOS EN GPS.TXT FUENTE: ELABORACIÓN PROPIA.	144
ILUSTRACIÓN 201 SCRIPT SACANDO FOTOS CONTINUAMENTE A LA DERECHA (TERMINAL VIDEO), CON SCRIPT DE VIDEO MULTIPLEXADO A LA IZQUIERDA (TERMINAL PRINCIPAL). FUENTE: ELABORACIÓN PROPIA.	145
ILUSTRACIÓN 202 TERMINAL MULTIPLEXADO VISTO CON SUDO SCREEN -X VIDEO: EL SCRIPT DE VIDEO NO PUEDE USAR LA CÁMARA OCUPADA EN TOMAR FOTOS. FUENTE: ELABORACIÓN PROPIA.	145
ILUSTRACIÓN 203 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	145
ILUSTRACIÓN 204 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	146
ILUSTRACIÓN 205 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	147
ILUSTRACIÓN 206 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	147
ILUSTRACIÓN 207 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	148
ILUSTRACIÓN 208 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	148
ILUSTRACIÓN 209 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	148
ILUSTRACIÓN 210 ACTUALIZACIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	149
ILUSTRACIÓN 211 EJECUCIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	149
ILUSTRACIÓN 212 VIDEO GENERADO DURANTE LA EJECUCIÓN DE AUTOMATICO.PY (PRINCIPIO DE VIDEO). FUENTE: ELABORACIÓN PROPIA.	150
ILUSTRACIÓN 213 VIDEO GENERADO DURANTE LA EJECUCIÓN DE AUTOMATICO.PY (FINAL DE VIDEO). FUENTE: ELABORACIÓN PROPIA.	150
ILUSTRACIÓN 214 FOTO TOMADA DURANTE LA EJECUCIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	150
ILUSTRACIÓN 215 METADATOS DE UNA DE LAS FOTOS TOMADAS DURANTE LA EJECUCIÓN DE AUTOMATICO.PY FUENTE: ELABORACIÓN PROPIA.	150

ILUSTRACIÓN 216 CONEXIÓN HARDWARE NECESARIA ENTRE AL CONTROLADORA Y LA RASPBERRY PI. FUENTE: ELABORACIÓN PROPIA.	151
ILUSTRACIÓN 217 CONECTARSE A VEHÍCULO CON DRONEKIT. FUENTE: HTTP://PYTHON.DRONEKIT.IO/GUIDE/CONNECTING_VEHICLE.HTML.....	152
ILUSTRACIÓN 218 CONEXIÓN SATISFACTORIA. FUENTE: ELABORACIÓN PROPIA.	152
ILUSTRACIÓN 219 LISTENER PARA "RANGEFINDER" TRAS IMPORTAR DRONEKIT.PY COMO LIBRERÍA. FUENTE: HTTP://PYTHON.DRONEKIT.IO/GUIDE/MAVLINK_MESSAGES.HTML.....	152
ILUSTRACIÓN 220 CLASS "RANGEFINDER" DENTRO DE DRONEKIT.PY. FUENTE:HTTPS://GITHUB.COM/DRONEKIT/DRONEKIT-PYTHON/BLOB/MASTER/DRONEKIT/___INIT__.PY	152
ILUSTRACIÓN 221 DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.	153
ILUSTRACIÓN 222 OUTPUT DE DRONEKIT_TEST. FUENTE: ELABORACIÓN PROPIA.	153
ILUSTRACIÓN 223 GLOBAL_POSITION_INT CON DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.....	154
ILUSTRACIÓN 224 GPS_RAW_INT CON DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.	154
ILUSTRACIÓN 225 ATTITUDE CON DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.	154
ILUSTRACIÓN 226 SYSTEM_TIME CON DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.	154
ILUSTRACIÓN 227 RAW_IMU CON DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.	154
ILUSTRACIÓN 228 FUENTE: ELABORACIÓN PROPIA.	154
ILUSTRACIÓN 229 DRONEKIT_TEST.PY FUENTE: ELABORACIÓN PROPIA.....	155
ILUSTRACIÓN 230 EJEMPLO DE MENSAJE MAVLINK. FUENTE: HTTP://MAVLINK.ORG/MESSAGES/COMMON	156
ILUSTRACIÓN 231 OUTPUT DRONEKIT_TEST.PY PARA UN SOLO OYENTE. FUENTE: ELABORACIÓN PROPIA.....	156
ILUSTRACIÓN 232 CÓDIGO EJEMPLO PARA SOLO OBTENER UN VALOR. FUENTE: HTTP://PYTHON.DRONEKIT.IO/GUIDE/VEHICLE_STATE_AND_PARAMETERS.HTML.....	156
ILUSTRACIÓN 233 PARÁMETROS DE AERONAVE EN COMANDO MAVLINK (ROLL DENTRO DE ATTITUDE). FUENTE: HTTP://MAVLINK.ORG/MESSAGES/COMMON.....	157
ILUSTRACIÓN 234 CÓDIGO PARA OBTENER EL "ROLL" DE LA AERONAVE. FUENTE: ELABORACIÓN PROPIA.....	157
ILUSTRACIÓN 235 OUTPUT ROLL. FUENTE: ELABORACIÓN PROPIA.	157
ILUSTRACIÓN 236 OUTPUT LONGITUD, LATITUD Y NÚMERO DE SATÉLITES DENTRO DE LA OFICINA. FUENTE: ELABORACIÓN PROPIA.	158
ILUSTRACIÓN 237 REPETICIÓN DE LA PRUEBA FUERA DE LA OFICINA. FUENTE: ELABORACIÓN PROPIA.....	159
ILUSTRACIÓN 238 MENSAJE MAVLINK. FUENTE: HTTP://MAVLINK.ORG/MESSAGES/COMMON	159
ILUSTRACIÓN 239 VERSIÓN ACTUALIZADA DE DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.	160
ILUSTRACIÓN 240 VERSIÓN ACTUALIZADA DE DRONEKIT_TEST.PY. FUENTE: ELABORACIÓN PROPIA.	161
ILUSTRACIÓN 241 EJECUCIÓN DEL PROGRAMA DENTRO DE LA OFICINA. FUENTE: ELABORACIÓN PROPIA.....	161
ILUSTRACIÓN 242 INFORMACIÓN DE LA CONTROLADORA CON REFERENCIACIÓN HORARIA, VISTO DESDE UN .TXT EN LA ESTACIÓN DE TIERRA. FUENTE: ELABORACIÓN PROPIA.	161
ILUSTRACIÓN 243 SCRIPT CORRESPONDIENTE A METADATOS.PY, PARA MANIPULACIÓN DE METADATOS. FUENTE: ELABORACIÓN PROPIA.	162
ILUSTRACIÓN 244 INSTALACIÓN CORRESPONDIENTE PARA PODER IMPORTAR "GIREPOSITORY". FUENTE: ELABORACIÓN PROPIA.	162
ILUSTRACIÓN 245 INSTALACIÓN CORRESPONDIENTE PARA PODER IMPORTAR "GIREPOSITORY". FUENTE: ELABORACIÓN PROPIA.	162
ILUSTRACIÓN 246 EJECUCIÓN Y METADATOS POR PANTALLA 1. FUENTE: ELABORACIÓN PROPIA.	163
ILUSTRACIÓN 247 EJECUCIÓN Y METADATOS POR PANTALLA 2. FUENTE: ELABORACIÓN PROPIA.	163
ILUSTRACIÓN 248 PROPIEDADES DE LA IMAGEN VISTO EN WINDOWS. FUENTE: ELABORACIÓN PROPIA.	163
ILUSTRACIÓN 249 PROPIEDADES DE LA IMAGEN VISTO EN WINDOWS. FUENTE: ELABORACIÓN PROPIA.	163
ILUSTRACIÓN 250 INSTALACIÓN DE UN ARCHIVO COMPRIMIDO .TAR. FUENTE: ELABORACIÓN PROPIA.	164
ILUSTRACIÓN 251 SCRIPT DD_TO_DMS.PY, DEDICADO A ABRIR UN CAMPO GPS PARA LAS IMÁGENES RGB. FUENTE: ELABORACIÓN PROPIA.	164

ILUSTRACIÓN 252 EJECUCIÓN DEL PROGRAMA ANTERIOR. FUENTE: ELABORACIÓN PROPIA	165
ILUSTRACIÓN 253 IMAGEN ESTÁNDAR CON METADATOS REFERENTES A INFORMACIÓN GPS OCULTOS. FUENTE: ELABORACIÓN PROPIA.	165
ILUSTRACIÓN 254 IMAGEN OBJETIVO DEL SCRIPT TRAS EJECUTARLO. FUENTE: ELABORACIÓN PROPIA.	165
ILUSTRACIÓN 255 METADATOS REFERENTES A GPS DE LA IMAGEN OBJETIVO, HASTA AHORA OCULTOS. FUENTE: ELABORACIÓN PROPIA.	165
ILUSTRACIÓN 256 SCRIPT DEDICADO A INCRUSTAR DATOS GPS EN POSPROCESADO. FUENTE: ELABORACIÓN PROPIA.	167
ILUSTRACIÓN 257 SCRIPT DEDICADO A INCRUSTAR DATOS GPS EN POSPROCESADO. FUENTE: ELABORACIÓN PROPIA.	167
ILUSTRACIÓN 258 SCRIPT DEDICADO A INCRUSTAR DATOS GPS EN POSPROCESADO. FUENTE: ELABORACIÓN PROPIA.	167
ILUSTRACIÓN 259 COMANDO POR CONSOLA, ADAPTADO A UN SCRIPT PARA USAR "EXIFTOOL". FUENTE: ELABORACIÓN PROPIA.	168
ILUSTRACIÓN 260 IMPORTACIÓN DE NUEVAS LIBRERÍAS. FUENTE: ELABORACIÓN PROPIA.	168
ILUSTRACIÓN 261 CONEXIÓN DE LA CONTROLADORA DE VUELO CON LA RASPBERRY PI. FUENTE: ELABORACIÓN PROPIA.....	168
ILUSTRACIÓN 262 OYENTE DENTRO DEL BUCLE INFINITO DE TOMA DE FOTOS. FUENTE: ELABORACIÓN PROPIA.	168
ILUSTRACIÓN 263 FUNCIÓN QUE SE DISPARA Y PROPORCIONA LAS POSICIONES AL ACTIVAR EL OYENTE. FUENTE: ELABORACIÓN PROPIA.	169
ILUSTRACIÓN 264 RESULTADO DE EJECUTAR EL SCRIPT AUTOMATICO.PY. FUENTE: ELABORACIÓN PROPIA.	169
ILUSTRACIÓN 267 ACTUALIZACIÓN AUTOMATICO.PY INICIO. FUENTE: ELABORACIÓN PROPIA.	170
ILUSTRACIÓN 268 ACTUALIZACIÓN AUTOMATICO.PY FINAL . FUENTE: ELABORACIÓN PROPIA.....	170
ILUSTRACIÓN 273 IMAGEN ORIGINAL E IMAGEN CON DATOS GPS INCRUSTADOS. FUENTE. ELABORACIÓN PROPIA.	171
ILUSTRACIÓN 274 GPSINFO.PY FUENTE: ELABORACIÓN PROPIA.	171
ILUSTRACIÓN 275 OUTPUT DE GPSINFO.PY EN LA OFICINA. FUENTE: ELABORACIÓN PROPIA.....	172
ILUSTRACIÓN 274 GPSINFO.PY FUENTE: ELABORACIÓN PROPIA.	172
ILUSTRACIÓN 275 OUTPUT DE GPSINFO.PY EN LA OFICINA. FUENTE: ELABORACIÓN PROPIA.....	172
ILUSTRACIÓN 276 EJECUCIÓN MANUAL DEL PROGRAMA PARA PODER VER RESULTADOS POR PANTALLA. FUENTE: ELABORACIÓN PROPIA.	173
ILUSTRACIÓN 277 FOTO CON DATOS GPS INCRUSTADOS. FUENTE: ELABORACIÓN PROPIA.....	173
ILUSTRACIÓN 278 OUTPUT GPSINFO.PY EN LA CARPETA COMPARTIDA. FUENTE: ELABORACIÓN PROPIA	173
ILUSTRACIÓN 279 RENAMEFLIR.PY EN CARPETA COMPARTIDA. FUENTE: ELABORACIÓN PROPIA.....	175
ILUSTRACIÓN 280 RENAMEFLIR.PY EN CARPETA COMPARTIDA. FUENTE: ELABORACIÓN PROPIA.....	175
ILUSTRACIÓN 281 RENAMEFLIR.PY EN CARPETA COMPARTIDA. FUENTE: ELABORACIÓN PROPIA.....	175
ILUSTRACIÓN 282 EJEMPLO DE NOMBRADO DE FOTO TÉRMICA. FUENTE: ELABORACIÓN PROPIA.	175
ILUSTRACIÓN 283 EJEMPLO RENAMEFLIR.PY . FUENTE. ELABORACIÓN PROPIA.	176
ILUSTRACIÓN 284 PROPIEDADES DE LA FOTO GENERADA . FUENTE: ELABORACIÓN PROPIA.	176
ILUSTRACIÓN 285 PROPIEDADES DE LA FOTO GENERADA . FUENTE: ELABORACIÓN PROPIA.	176
ILUSTRACIÓN 286 FOTO GENERADA (VISIÓN TÉRMICA DE LA MESA, POR ESO ES UNIFORME). FUENTE: ELABORACIÓN PROPIA. .	177
ILUSTRACIÓN 287 USO DE RENAMEFLIR.PY. FUENTE: ELABORACIÓN PROPIA.	178
ILUSTRACIÓN 288 ACTUALIZACIÓN DE AUTOAMATICO.PY FUENTE: ELABORACIÓN PROPIA.	178
ILUSTRACIÓN 289 EJECUCIÓN DE AUTOMATICO.PY ACTUALIZADO. FUENTE: ELABORACIÓN PROPIA.....	179
ILUSTRACIÓN 290 PERMISOS EN LA CARPETA DESTINO. FUENTE: ELABORACIÓN PROPIA.	179
ILUSTRACIÓN 291 IMAGEN GENERADA POR AUTOMATICO.PY ACTUALIZADO. FUENTE: ELABORACIÓN PROPIA.	180
ILUSTRACIÓN 292 IMAGEN GENERADA POR AUTOMATICO.PY ACTUALIZADO. FUENTE: ELABORACIÓN PROPIA.	180
ILUSTRACIÓN 293 IMAGEN GENERADA POR AUTOMATICO.PY ACTUALIZADO. FUENTE: ELABORACIÓN PROPIA.	180
ILUSTRACIÓN 294 MISIÓN A REALIZAR DURANTE LA PRUEBA. FUENTE: SERVIDOR INTERNO LOCIS..... ¡ERROR! MARCADOR NO DEFINIDO.	
ILUSTRACIÓN 295 WAYPOINTS O CORRESPONDIENTES A LA PRUEBA. FUENTE: SERVIDOR INTERNO LOCIS. ¡ERROR! MARCADOR NO DEFINIDO.	
ILUSTRACIÓN 296 ARCHIVO .TXT CON LOS DATOS DE LA PRIMERA MISIÓN DE VUELO. FUENTE: SERVIDOR INTERNO LOCIS.	183

ILUSTRACIÓN 297 ARCHIVO .TXT CON LOS DATOS DE LA SEGUNDA MISIÓN DE VUELO. FUENTE: SERVIDOR INTERNO LOCIS.....	183
ILUSTRACIÓN 298 LOG DEL PARÁMETRO ATT YAW PARA LA MISIÓN 1. FUENTE: SERVIDOR INTERNO LOCIS.	185
ILUSTRACIÓN 299 LOG DEL PARÁMETRO ATT YAW PARA LA MISIÓN 2. FUENTE: SERVIDOR INTERNO LOCIS.	186
ILUSTRACIÓN 300 GRÁFICO DE ENTRADA DEL CANAL 4 MOSTRADO A TRAVÉS DEL LOG DE LA CONTROLADORA ENTRE LOS PUNTOS 6-7. FUENTE: SERVIDOR INTERNO LOCIS.....	187
ILUSTRACIÓN 301 EJEMPLO DE GUARDADO DE FOTOS RGB POR CONTADOR. FUENTE: SERVIDOR INTERNO LOCIS.	187
ILUSTRACIÓN 302 DRON A PUNTO DE DESPEGAR EN EL CAMPO DE PRUEBAS LOCIS. FUENTE: SERVIDOR INTERNO LOCIS.	190
ILUSTRACIÓN 303 CAMPO DE PRUEBAS A VISTA DE DRON (RGB + TÉRMICA). FUENTE: SERVIDOR INTERNO LOCIS.....	190
ILUSTRACIÓN 304 MISIÓN A REALIZAR DURANTE LA PRUEBA. FUENTE: SERVIDOR INTERNO LOCIS.....	192
ILUSTRACIÓN 305 WAYPOINTS O CORRESPONDIENTES A LA PRUEBA. FUENTE: SERVIDOR INTERNO LOCIS.	192
ILUSTRACIÓN 306 ARCHIVO .TXT CON LOS DATOS DE LA PRIMERA MISIÓN DE VUELO. FUENTE: SERVIDOR INTERNO LOCIS.	193
ILUSTRACIÓN 307 ARCHIVO .TXT CON LOS DATOS DE LA SEGUNDA MISIÓN DE VUELO. FUENTE: SERVIDOR INTERNO LOCIS.....	193
ILUSTRACIÓN 308 EJEMPLO PROPIEDADES IMAGEN DURANTE LA PRUEBA. FUENTE: SERVIDOR INTERNO LOCIS.	194
ILUSTRACIÓN 309 DRON ANTES DEL DESPEGUE. FUENTE: SERVIDOR INTERNO LOCIS.	195
ILUSTRACIÓN 310 DRON DURANTE EL VUELO. FUENTE: SERVIDOR INTERNO LOCIS.	195
ILUSTRACIÓN 311 PRIMERA MISIÓN UBICADA EN EL CAMPO DE PRUEBAS LOCIS. FUENTE: SERVIDOR INTERNO LOCIS.	197
ILUSTRACIÓN 312 WAYPOINTS CORRESPONDIENTES A LA PRIMERA MISIÓN. FUENTE: SERVIDOR INTERNO LOCIS.....	198
ILUSTRACIÓN 313 WAYPOINTS CORRESPONDIENTES A LA PRIMERA MISIÓN. FUENTE: SERVIDOR INTERNO LOCIS.....	199
ILUSTRACIÓN 314 IMAGEN RGB DE LA PRUEBA. FUENTE: SERVIDOR INTERNO LOCIS.	200
ILUSTRACIÓN 315 IMAGEN TÉRMICA DE LA PRUEBA. FUENTE: SERVIDOR INTERNO LOCIS.	200
ILUSTRACIÓN 316 MISIÓN INSPECCIÓN DE PANELES SOLARES EN LA AZOTEA DE EDIFICIO. FUENTE: SERVIDOR INTERNO LOCIS.	202
ILUSTRACIÓN 317 CONTROL VISUAL DESDE EL TEJADO DE LA AZOTEA (VISTA DESDE EL DRON). FUENTE: SERVIDOR INTERNO LOCIS.	204
ILUSTRACIÓN 318 PUESTA A PUNTO DE LOS EQUIPOS. FUENTE: SERVIDOR INTERNO LOCIS.	205
ILUSTRACIÓN 319 AZOTEA A VISTA DE DRON. FUENTE: SERVIDOR INTERNO LOCIS.	205
ILUSTRACIÓN 320 IMAGEN TÉRMICA DE UNO DE LOS VUELOS, CON DETALLE DE UN POSIBLE PUNTO CALIENTE. FUENTE: SERVIDOR INTERNO LOCIS.	205
ILUSTRACIÓN 321 RUTA EN LA QUE SE BUSCAN LAS IMÁGENES TÉRMICAS DE FORMA AUTOMÁTICA. FUENTE: ELABORACIÓN PROPIA.	206
ILUSTRACIÓN 322 RUTA DE /MEDIA/PI DENTRO DE LA RASPBERRY. FUENTE: ELABORACIÓN PROPIA.	206
ILUSTRACIÓN 323 COMANDO LS -L PARA VISIONADO POR CONSOLA. FUENTE: ELABORACIÓN PROPIA.	207
ILUSTRACIÓN 324 SENSOR FM10LS. FUENTE: HTTPS://WWW.VITALITOOLS.NL/FAUSER-FM10LS	209
ILUSTRACIÓN 325 ESPECIFICACIONES TÉCNICAS FM10LS. FUENTE: UNIOVI.	210
ILUSTRACIÓN 326 SOFTWARE FM-DATA. FUENTE: HTTPS://WWW.FAUSER.BIZ/SE/REG_SW_E_ANTWORT.HTM ACCESO:24/4/2018.	210
ILUSTRACIÓN 327 SENSOR NFA1000. FUENTE: HTTPS://WWW.GOOGLE.ES/SEARCH?Q=NFA1000&SOURCE=LNMS&TBM=ISCH&SA=X&VED=0AHUKEWI_9BUJT6DAAHULVBQKHd71CKkQ_AUICyGC&BIW=1440&BIH=745#IMGRC=bY3T5sv93V-pYM : ACCESO: 24/4/2018.	211
ILUSTRACIÓN 328 ESPECIFICACIONES TÉCNICAS. FUENTE: UNIOVI.	212
ILUSTRACIÓN 329 NFASOFT. FUENTE: HTTPS://WWW.GIGAHERTZ-SOLUTIONS.DE/EN/RF-AND-EMF-METERS/LOW-FREQUENCY- EMF/NFA-SERIES/334/NFASOFT ACCESO: 24/4/2018.	212
ILUSTRACIÓN 330 ESPECIFICACIONES TÉCNICAS. FUENTE: UNIOVI.	213
ILUSTRACIÓN 331 ESPECIFICACIONES TÉCNICAS. FUENTE: UNIOVI.	215
ILUSTRACIÓN 332 ESPECIFICACIONES TÉCNICAS. FUENTE: UNIOVI.	215

1. Introducción

En los últimos años el público se ha ido familiarizando cada vez más con el término UAV (“Unmanned Aerial Vehicle”, vehículo aéreo no tripulado) hasta el punto de que es posible verlos frecuentemente en parques y otros lugares de la geografía española.

Los también denominados drones comienzan su andadura como artículos dedicados al ocio y con escaso impacto en el ámbito empresarial. Pero recientemente y debido a su abaratamiento empiezan a adquirir importancia en tareas de mantenimiento preventivo, esenciales en la industria.

Uno de los principales campos de aplicación de este tipo de vehículos dentro del mantenimiento preventivo se encuentra en la vigilancia e inspección aérea, principalmente debido a la posibilidad de automatizar los procesos de vuelo y a no necesitar casi ningún tipo de interacción humana.

En el presente TFM se intentará dar una visión de la capacidad de estos vehículos en dos situaciones diferentes.

Primero en instalaciones industriales de generación de energía fotovoltaica, y segundo en mantenimiento de torres de alta tensión.

Para el primer tipo de instalaciones se opta por la realización de dos tipos de fotografías (térmicas y RGB).

Las imágenes termográficas permiten detectar cualquier anomalía referida a punto caliente en la superficie de los paneles solares. Mientras que la imagen RGB permite la inspección visual convencional.

En el segundo tipo de instalaciones se opta por el uso de un módulo que cuenta con: un sensor térmico, sensores de campo eléctrico y magnético, y una cámara RGB.

En este caso se buscarán anomalías respecto a lo considerado como aceptable en las mediciones de campo y temperatura producidas por cualquier tipo de defecto ocurrido en las líneas. Asimismo se cuenta con una cámara RGB para inspección visual.

De este modo el uso de drones constituye una herramienta predictiva que permite evaluar la aparición de fallos y problemas sin llegar a necesitar la parada del sistema, ya sea este el “huerto” solar o una línea de tensión. Contribuyendo también a la seguridad humana debido a la realización a distancia de cualquier tipo de medición.

Tradicionalmente este tipo de tareas requería una inversión elevada tanto en tiempo de operación como en seguridad del personal, debido a la peligrosidad de cualquier aparato eléctrico.

Por lo tanto el uso de UAVs presenta elevadas ventajas como:

- Disminución del uso de recursos humanos y reducción de costes en temas relativos a su seguridad

- Optimización de recursos
- Toma instantánea de datos en menor tiempo

Asimismo la posibilidad de dotar a las mediciones de los sensores y las imágenes de las cámaras de geotiquetado en tiempo real permite la elaboración de mapas tanto térmicos como de campo electromagnético estableciendo así separación por zonas de especial problemática o importancia.

2. Objetivos Generales

El objetivo principal de este TFM es estudiar la implementación del uso de drones en tareas de inspección y mantenimiento preventivo en instalaciones industriales.

Se deben conseguir “inputs” en forma de imágenes y mediciones que de forma remota se envíen a una plataforma donde se procederá a su análisis para la generación de “outputs” en forma de informes y soluciones.

Los objetivos parciales que deben lograrse a lo largo del desarrollo del TFM pueden resumirse de la siguiente manera:

- Sistema de captación de datos de forma automatizada en cada una de las misiones de vuelo
- Envío de los datos en forma de imágenes o mediciones de forma automática
- Lectura e interpretación de los datos

3. Contexto, alcance y planificación

3.1.- CONTEXTO

En el presente TFM se describe la mecánica del funcionamiento de un sistema de monitorización de instalaciones industriales mediante el uso de UAVs.

Se trata de un proyecto desarrollado dentro del entorno de la investigación, con título “Desarrollo y ensayo de un sistema de detección de fallos en plantas de generación solar mediante inspección aérea automática” y abarcando determinadas partes de dos proyectos más extensos:

El primero “UAV Inspection: Estudio de un sistema para la gestión automática de misiones aéreas”, proyecto realizado por la empresa Locis Sigtech Soluciones Sostenibles SSL y TSK Electrónica y Electricidad S.A.

Y, el segundo “Sistema de monitorización de torres de tensión mediante vehículos aéreos no tripulados” realizado por la colaboración de las empresas Locis y Gomeru.

Ambos proyectos están regulados a través de programas orientados al I+D+I, que cumplen tres objetivos principales: afrontar nuevos procesos de innovación por parte de las empresas, conseguir una mejora competitiva de cara a una excelente especialización y por último una focalización en su diversificación llegando incluso a la internalización de su mercado.

El organismo regulador del primer proyecto es el Instituto de Desarrollo Económico del Principado de Asturias (IDEPA), y del segundo la Universidad de Oviedo (UNIOVI).

3.2.- ALCANCE

Al realizar un TFM en base a la adaptación de dos proyectos más extensos es conveniente delimitar las partes de trabajo de cada uno de los interesados.

En la línea del proyecto se pretende continuar con la mejora e implementación de sistemas del proyecto UAV Inspection, como asimismo realizar las correspondientes pruebas de vuelo.

Con esto se pretende dotar a la empresa colaboradora de un sistema capaz de tomar fotos en tiempo real que permitan la posterior generación de informes que ayuden en la toma de decisiones de las labores de mantenimiento.

Posteriormente, se pretende realizar el acoplado de los sistemas pertinentes y montaje del vehículo en el proyecto de monitorización de torres de tensión.

Los objetivos generales del primer proyecto se desglosan en pequeños ítems para facilitar su consecución, del modo:

- Automatización del proceso de toma de imágenes y posterior compartición con la estación de tierra.
- Marca temporal y geotiquetado de las imágenes térmicas y RGB.
- Generación y envío automático de las misiones a la controladora de vuelo.
- Pruebas y ensayos
- Implementación de dispositivos.

En referencia al segundo proyecto su desglose quedaría de la manera:

- Elección de los sensores y de las partes que compondrán el sistema.
- Pruebas y ensayos.

El esquema general de ambos proyectos se resume del siguiente modo:

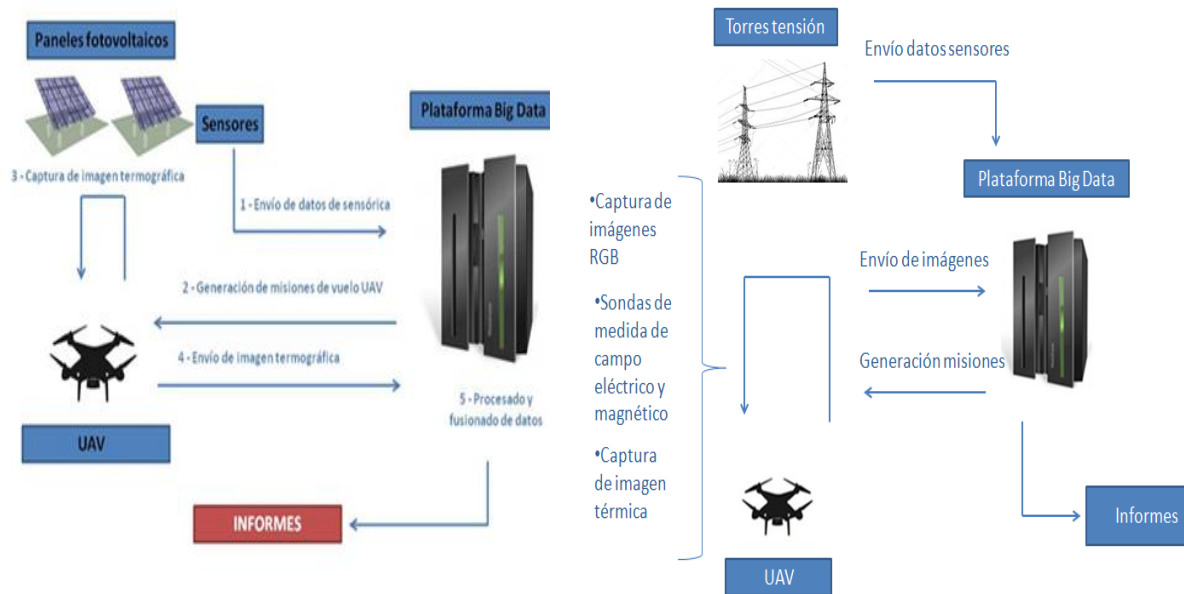


Ilustración 1 Esquema UAV Inspection. Fuente: Servidor interno Locis Sigtech.

Ilustración 2 Esquema torres de tensión. Fuente: Elaboración propia.

3.3.- PLANIFICACIÓN

Cada uno de los proyectos colaborativos se corresponde con diferentes fases secuenciadas e independientes unas de otras, del modo:

En UAV Inspection se puede diferenciar entre:

- Fase I: Especificación: La fase inicial del proyecto se caracteriza por diferenciar el problema a resolver, analizando los posibles requisitos que podamos tener para lograr su consecución como también las soluciones ya existentes en otras partes del mundo. Esta fase es meramente investigadora, en la cual se requiere una solución y también el desglose de medios, mecanismos y tecnologías usadas para su consecución.
- Fase II: Investigación en la plataforma aérea. Esta fase también conlleva un alto trabajo de investigación. Se pretende definir la plataforma del dron que mejor realice el proceso de toma de imágenes y su posterior integración con el resto de los equipos. En esta fase se incluye la fabricación y diseño del prototipo, como también su completa automatización.
- Fase III: Investigación en la plataforma de procesamiento: Esta fase consistirá mayormente en el pos-procesado de la información recopilada en cada una de las misiones de vuelo, con el principal objetivo de encontrar soluciones óptimas.

Dentro de este proyecto podemos desglosar las tareas del modo siguiente:

Arquitectura del sistema	Captura de imagen	UAV	Sistemas	Información GNSS
Definición y diseño de la arquitectura del sistema	Definición y diseño de la alternativa	Definición y diseño del dron	Envío Misión	Definición y requerimientos
Elaboración esquema de arquitectura del sistema UAV	Diseño y configuración de la Pixhawk como fuente de órdenes	Configuración física del dron y configuración de la controladora	LED	Investigación y diseño
	Pruebas	Pruebas de vuelo	RTK	Geo-tag fotos térmicas
	Entregable: Módulo funcional de toma de imágenes.		Lidar	Geo-tag fotos RGB
			FPV	Pruebas

Ilustración 3 Paquetes de trabajo del proyecto UAV-Inspection. Fuente: Elaboración propia.

Con el siguiente porcentaje de realización:

- Arquitectura del sistema: Esta fase se encontraba desarrollada al 100%.
- Captura de imagen: Esta fase se encontraba desarrollada al 100%.
- UAV: La fase pendiente por realizar dentro de este apartado es la correspondiente a las pruebas de vuelo.

- Dentro del apartado de “Sistemas” se realizarán las fases correspondientes con:
 - Envío de la misión
 - RTK
 - Lidar
 - FPV
 - LED
- Información GNSS: Dentro de este apartado es necesaria la realización completa de todas sus fases.

Por su parte dentro del proyecto de monitorización de torres se puede diferenciar entre:

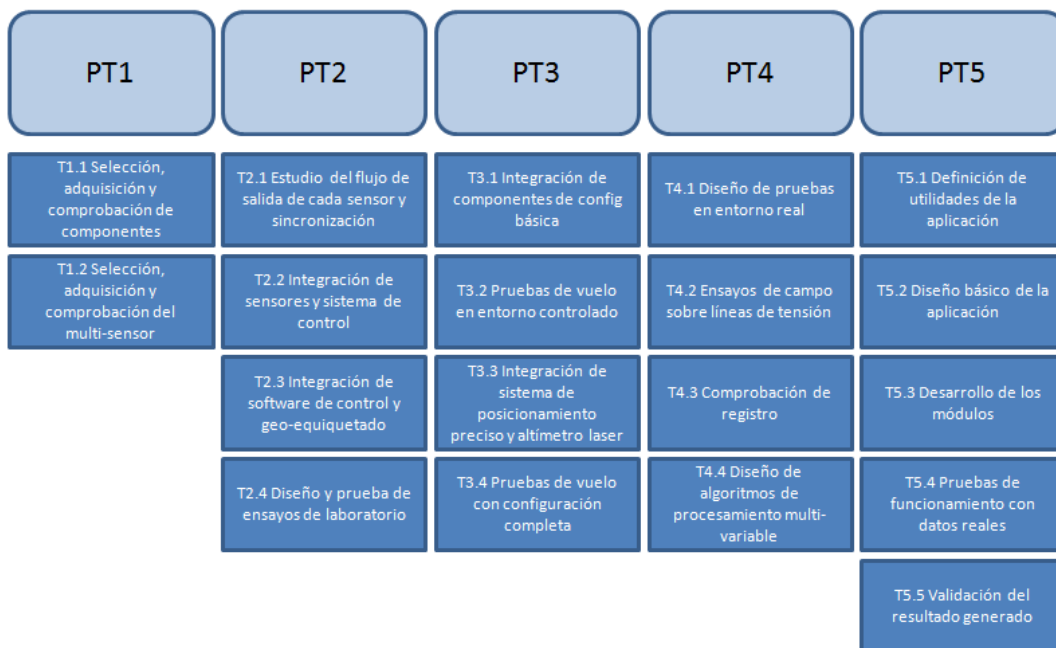


Ilustración 4 Paquetes de trabajo del proyecto monitorización de torres de tensión. Fuente: Elaboración propia.

Paquetes de Trabajo (PT)	Nº	TAREA	ENTIDAD Responsable	ENTIDAD Colaboradora	2017	2018												2019																
						D	E	F	M	A	M	J	J	A	S	O	N	D	E	F	M	A												
PT1	TL1	Selección, adquisición y comprobación de componentes del RPA	LOCIS	LOCIS	X	X																												
	TL2	Selección, adquisición y comprobación de componentes del multisensor	LOCIS	UNIOVI	X	X	H1																											
Entregable 1																																		
PT2	T2.1	Estudio del flujo de salida de cada sensor y su sincronización con el resto	LOCIS	UNIOVI			X	X	X																									
	T2.2	Integración de los sensores y el sistema de control	LOCIS	UNIOVI						X	X	X																						
	T2.3	Implementación del software de control y geotiquetado	LOCIS	UNIOVI											X	X																		
	T2.4	Diseño y prueba de ensayos de funcionamiento en laboratorio	LOCIS	UNIOVI													X	X	H2															
Entregable 2																																		
PT3	T3.1	Integración de componentes del RPA de configuración básica	LOCIS	LOCIS			X	X	X																									
	T3.2	Pruebas de vuelo iniciales en entorno controlado	LOCIS	LOCIS						X																								
	T3.3	Integración de sistema de posicionamiento preciso y altímetro láser	LOCIS	LOCIS						X	X																							
	T3.4	Pruebas de vuelo con configuración completa	LOCIS	LOCIS								X	X	H3																				
Entregable 3																																		
PT4	T4.1	Diseño de pruebas en entorno real	LOCIS	UNIOVI														X																
	T4.2	Ensayos de campo sobre líneas en tensión	LOCIS	LOCIS														X	X	X														
	T4.3	Comprobación de datos de registro en campo	LOCIS	UNIOVI																					X	X								
	T4.4	Diseño de algoritmos de procesamiento multivariable	LOCIS-GOMERU	UNIOVI																				X	X	H4								
Entregable 4																																		
PT5	T5.1	Definición de utilidades de la aplicación	LOCIS-GOMERU	LOCIS-GOMERU							X	X																						
	T5.2	Diseño básico de la aplicación	GOMERU	GOMERU										X	X																			
	T5.3	Desarrollo de los módulos	GOMERU	GOMERU																X	X	X												
	T5.4	Pruebas de funcionamiento con datos reales	GOMERU	GOMERU																		X	X											
	T5.5	Validación del resultado generado	LOCIS-GOMERU	LOCIS-GOMERU																											X	H5		

Ilustración 5 Ampliación paquetes de trabajo. Fuente: Uniovi.

El proyecto de torres de tensión se divide en cinco paquetes de trabajo diferenciados, cada uno de ellos con un ejecutor diferente.

En concreto la parte a desarrollar por el alumno constará de las partes de selección y adquisición de componentes, en base a las opciones de referencia proporcionadas por la UNIOVI.

Se puede comprobar cómo la mayoría de las actividades a realizar implican la participación de dos grupos de trabajo ya sea de alguna de las empresas colaboradoras o UNIOVI, por lo que las tareas realizadas por el alumno se desglosan en mayor medida en las siguientes tablas.

Además, debido a la incertidumbre en cuanto a tiempos de esta clase de proyecto puede ser que se observen movimientos en las fechas de realización, o tareas sin cumplimiento al 100%.

T1.1: Selección adquisición y comprobación de componentes del multisensor.	
Duración	Aproximadamente 2 meses
Objetivos	Selección de posibles sensores del módulo
	Tabla de características y precios
	Selección final

Ilustración 6 Ampliación paquetes de trabajo. Fuente: Elaboración propia.

Para entender el proyecto es necesario especificar el tiempo de prácticas del alumno dentro de la empresa LOCIS SIGTECH, ya que dicho periodo formativo se corresponde con la ejecución de dos proyectos reales de la compañía.

El horizonte de ejecución de las prácticas se extiende desde el 13 de febrero al 31 de junio, alcanzando un total de 720 horas.

Además, la realización del proyecto UAV-Inspection corresponde a la continuación de un proyecto anterior, los límites de dicho proyecto se esclarecerán en partes posteriores de la memoria.

4. Estado del arte

4.1.- CONCEPTO

Un vehículo aéreo no tripulado (VANT) o UAV (del inglés “unmanned aerial vehicle”) es una aeronave que vuela sin tripulación.

Dicha aeronave cuenta con capacidad de mantenerse en el aire de forma autónoma en un vuelo controlado y sostenido, propulsado por algún tipo de motor, generalmente eléctrico.

4.2.- ORIGEN E HISTORIA

El término UAV se hizo común en la década de los años 90 para hacer referencia a las aeronaves robóticas, reemplazando de este modo al término vehículo aéreo con control por piloto remoto (Remotely Piloted Vehicle, RPV) usado mayormente durante la guerra de Vietnam y momentos posteriores.

Tanto la nomenclatura UAV como RPV corresponden a dos de los muchos nombres que sirven para hacer referencia a las aeronaves robóticas no tripuladas a lo largo de su existencia.

Otros de los acrónimos más usados son:

- UMA= Unmanned Aircraft
- APV= Automatically Piloted Vehicle
- UTA= Unmanned Tactical Aircraft
- UCAV= Unmanned Combat Vehicle
- ROA= Remotely Operated Aircraft

A pesar de la creencia popular la idea del avión no tripulado es antigua. De hecho, los UAVs, de una u otra forma, se han usado durante décadas.

Uno de los primeros usos de este tipo de tecnologías se atribuye al asedio austriaco en julio de 1849 sobre la ciudad de Venecia, en el cual se desplegaron alrededor de doscientos globos aero-estáticos no tripulados armados con bombas.

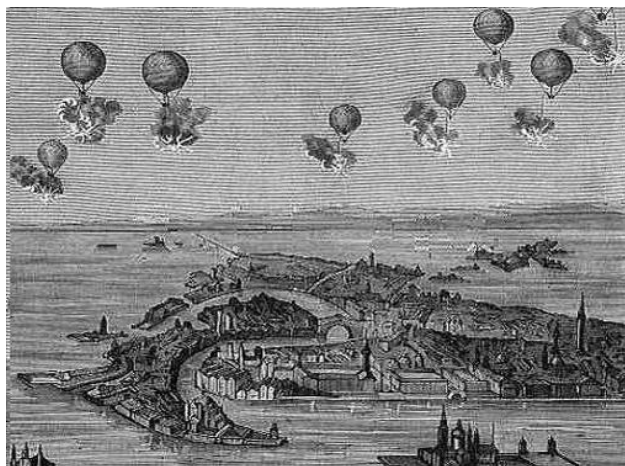


Ilustración 7 Ejército Austriaco sobre la ciudad de Venecia. Fuente: eldrone.es

El siguiente paso en la evolución de los vehículos no tripulados se produce cuando en 1898 militares americanos equipan una cámara a una cometa, para controlar al enemigo en la guerra Hispanoamericana, dando lugar así a los albores de la práctica de la vigilancia aérea con una de las primeras fotografías de reconocimiento aéreo.

Este nuevo hito fue ampliamente utilizado durante la segunda Guerra Mundial para seguir el rastro a los movimientos de las tropas enemigas sin comprometer vidas humanas, para, de este modo componer mapas de situación con las posiciones enemigas.

La evolución y desarrollo del uso del dron se produce en gran manera gracias a innovaciones producidas durante conflictos armados mayormente pasando por tres etapas principales.

Primero, su uso como diana en prácticas de tiro, segundo como bomba volante no tripulada, y tercero como mero sistema de plataforma de vigilancia y toma de datos para lugares de difícil acceso.

Pero no fue hasta el descubrimiento de la radio cuando el mundo de la aviación no tripulada vivió su apogeo.

En 1858 se completa el primer telégrafo transoceánico. Este aparato quedaba limitado por el uso del propio cable, por lo que no tenía muchas posibilidades en el mundo de la aviación. La irrupción de la radio, sin embargo, permitía el envío de información a través de la atmósfera. Esta nueva posibilidad probó ser fundamental para el posterior desarrollo de los UAVs actuales.

A finales del siglo XIX, en uno de los estanques de Madison Square Garden de Nueva York en el año 1898 el inventor americano Nicola Tesla controla por primera vez en la historia un barco con una señal de radio.

Más tarde, en 1916 el capitán Archibald Low de la Royal Flying Corps de Reino Unido manda desarrollar la primera flota de aviones bomba teledirigidos. Posteriormente, su invención sería clave en el entorno de la aviación teledirigida durante la primera guerra mundial.

En concreto, durante esta guerra surgirían nuevos vehículos no tripulados y manejados por radio control como la “Hewitt Sperry” también conocida como bomba volante, la cual era equipada con una bomba de aproximadamente 300 libras de peso y con una autonomía de 50 millas de distancia.



Ilustración 8 "Hewitt Sperry". Fuente: eldrone.es

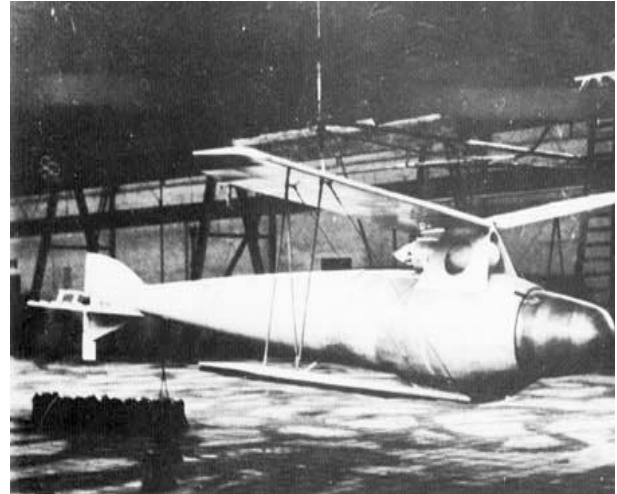


Ilustración 9 Modelo alemán versión de "Hewitt Sperry". Fuente: eldrone.es

Este tipo de tecnología vivió su apogeo durante la Segunda Guerra Mundial, cuando la Marina estadounidense desarrolló el programa Anvil, usando aviones bomba radio controlados capaces de estrellarse en las defensas alemanas.

Otro tipo de elementos radiocontrol fueron usados también durante la guerra, como por ejemplo la bomba GB- Glide. Una versión ligeramente modificada del anterior, más grande, con mayor autonomía de vuelo y capacidad para llevar bombas más pesadas.



Ilustración 10 GB-1 Glide. Fuente: proyecto-alfa.net

Durante la década de los años 50 y durante la guerra de Vietnam se popularizó el uso de la aeronave “Firebee” producida por la compañía aeronáutica Ryan, el modelo contaba con una autonomía de dos horas y era capaz de alcanzar los 60.000 pies.

La guerra de Vietnam siempre será recordada como la primera “guerra tecnológica” de la historia. El campo de batalla se plagó de dispositivos electrónico- tecnológicos, con la popularización de los “Firebee” como sistemas de vigilancia en las espesas selvas de Vietnam.

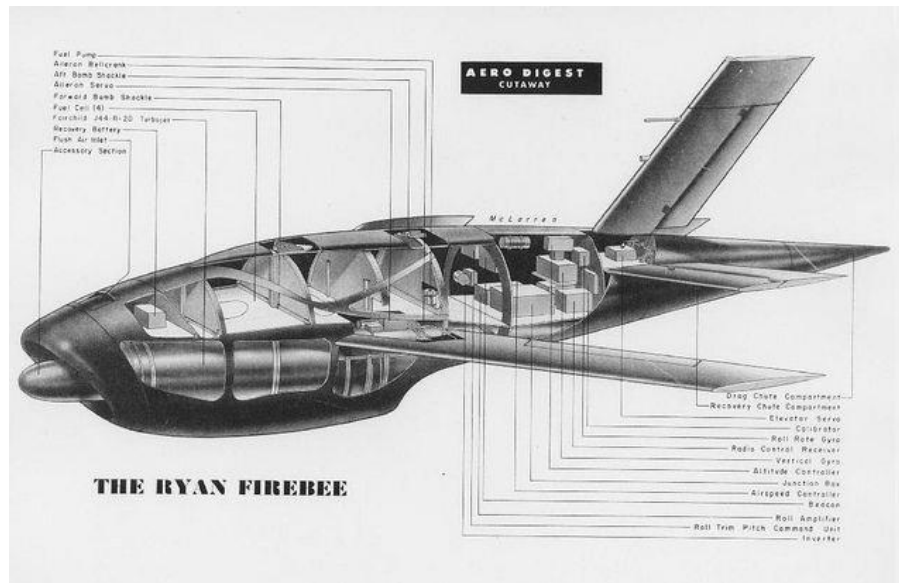


Ilustración 11 "Firebee" de la compañía Ryan. Fuente: wikipedia.org

Durante la Guerra Fría se produjo la evolución técnica y tecnológica del modelo “Firebee” dando lugar a multitud de aeronaves. Siendo uno de los más populares el modelo “Tom Cat”.



Ilustración 12 "Tom Cat". Fuente: eldrone.es

El “campo de batalla tecnológico” de la Guerra de Vietnam resultó fundamental para el desarrollo del dron actual, y sobre todo para el desarrollo de las aeronaves no tripuladas civiles.

En 1972 se equipa por primera vez en la historia aviones teledirigidos con un sistema LORAN (Long Range Navigation o navegación de largo alcance), este sistema de navegación calcula la posición del usuario por recepción de ondas de radio (de modo similar a un GPS de hoy en día).

Durante la década de los años 70 la fuerza aérea norteamericana impulsa el uso de vehículos teledirigidos (RPVs) como el modelo Aquila, este tipo de vehículos no tripulados eran capaces de volar a gran altitud convirtiéndose así en los modelos más ambiciosos de la historia en ese momento.



Ilustración 13 Modelo dron Aquila. Fuente: eldrone.es

Posteriormente una colaboración norteamericana-israelí dio origen al “Predator”.

Este avión no tripulado de combate inicia la era contemporánea en la historia de los drones, siendo especialmente usado en tareas de incursiones sobre objetivos peligrosos, como se puede ver en la siguiente imagen:



Ilustración 14 Predator sobre Afganistán. Fuente: Google.



Ilustración 15 Primer uso del "Predator", vista FPV. Fuente: eldrone.es

4.3.- ACTUALIDAD

En la actualidad el uso de drones no se limita únicamente a la industria militar (aunque sus inicios se deban a ella), de hecho, podemos ver cada vez más aplicaciones para los vehículos no tripulados tales como, toma de imágenes, seguridad, investigación y entretenimiento.

4.3.1.- Seguridad

En el campo de la seguridad se aprecian muchos avances en cuanto a monitorización, por ejemplo, para la vigilancia del tráfico.

Este tipo de iniciativas son las que busca desarrollar la Ertzaintza (policía autonómica vasca) que actualmente estudia el uso de drones alacrán (de la empresa española INDA) para su empleo en misiones de seguridad, rescate y vigilancia de tráfico.



Ilustración 16 Dron de vigilancia en carretera. Fuente: <http://sorro.es/drones-policiales-para-vigilancia-del-trafico/> Acceso: 20/4/2018.

Los drones de este tipo son capaces de alcanzar los 100 km por hora y cuentan con la posibilidad de compartir datos a la estación de tierra en tiempo real.

4.3.2.- Entretenimiento

En el campo del entretenimiento la moda de los drones conquista incluso a los más pequeños, con vehículos no tripulados acordes a su tamaño.



Ilustración 17 Dron para entretenimiento. Fuente: <https://www.mediatrends.es/a/53050/drones-para-ninos/> Acceso: 20/4/2018.

Este tipo de vehículos vuelven al radiocontrol, por razones de sencillez y precio, siendo especialmente capaces de realizar divertidas piruetas y acrobacias.

4.3.3.- Ciencia e investigación

Dentro del campo de la ciencia y la investigación encontramos a drones realizando tareas tales como la documentación arqueológica, topografía y cartografía o la teledetección agrícola.

La documentación arqueológica es un trabajo que depende en gran medida del registro de imágenes y la geolocalización de estas, normalmente en terrenos de difícil acceso a pie. Debido a esto encontramos aquí un nicho de mercado para los fabricantes de vehículos aéreos no tripulados.

Para tareas de topografía existe software especializado en la generación de plantillas y puntos de interés, creando de este modo, una ruta de fotografías solapadas que permitan establecer un modelo 3D de la superficie.

Gracias a la opción de implantar diferentes tipos de hardware en drones tenemos la posibilidad de aumentar mucho sus prestaciones, como por ejemplo se está haciendo en las grandes plantaciones de flores de México.

Allí se usan cámaras multi-espectrales embarcadas en drones para identificar y cuantificar la variabilidad de los cultivos, determinar si hay plagas, o falta de riegos y nutrientes.



Ilustración 18 Dron para teledetección agrícola. Fuente: <https://www.google.es>

De esta manera se consiguen mejorar los métodos satelitales usados actualmente para este tipo de tareas, aumentando la resolución de las imágenes y la efectividad de la metodología.

4.4.- PROYECTOS FUTUROS

4.4.1.- Aquila de Facebook

Uno de los proyectos más interesantes y ambiciosos es el que planea desarrollar la empresa americana Facebook.

La multinacional cuenta actualmente con un modelo prototipo de dron capaz de moverse usando energía solar.

La aeronave, denominada Aquila, tiene como principal objetivo dotar de conexión a internet áreas remotas, haciendo así honor a la promesa del dueño de la compañía de globalizar la red.



Ilustración 19 Aquila de Facebook. Fuente: <https://www.google.es>

4.4.2.- Drones repartidores de Amazon

La empresa americana Amazon planea habilitar en los próximos años una flota prototipo de drones repartidores.

El cliente, de este modo podrá ordenar su producto en la popular tienda online para esperar a su reparto en la comodidad del sofá de su casa.

Esta idea sin embargo cuenta con algunas trabas, como por ejemplo la legislación que concierne a este tipo de vehículos, las limitaciones de uso en ciudades y la gestión y automatización de la inmensa cantidad de pedidos de la compañía.



Ilustración 20 Drones repartidores. Fuente: https://www.eldiario.es/hojaderouter/drones/planes-delirantes-Amazon-drones-repartidores_0_700631063.html Acceso: 24/7/2018.

4.5.- MISIONES DE INSPECCIÓN CON DRONES EN INSTALACIONES INDUSTRIALES

Actualmente existen multitud de operaciones de vigilancia industrial desarrolladas con el uso de drones, como pueden ser:

- Inspección de bornes en transformadores eléctricos.
- Inspección en calderas de biomasa.
- Inspección en depósitos de agua.
- Inspección termográfica de granjas solares y tendidos eléctricos mediante análisis de puntos calientes para la detección y prevención de averías.

4.6.- NORMATIVA

Desde el 30 de diciembre de 2017 la ley que aplica al uso civil de aeronaves pilotadas por control remoto (RPAs) en España es el Real Decreto 1036/2017 (el cual se puede consultar en la bibliografía).

5. GENERALIDADES

5.1.- ELEMENTOS QUE CONSTITUYEN UN UAV

5.1.1.- Chasis o plataforma

Componente inicial sobre el que se sustentan el resto de los subsistemas, que por lo general son:

- Sistema de propulsión: algún tipo de motor, siendo eléctrico el más común.
- Navegación: o gestor de vuelo.
- Sistema de posicionamiento: por lo general se trata de un sistema GPS.

5.1.2.- Carga

Diferentes equipos necesarios para la consecución de los objetivos de la misión, como por ejemplo cámaras o sensores de medición.

5.1.3.- Controladora

Unidad especial dotada de sensores para analizar el entorno, con capacidad para controlar los diferentes actuadores (motores) para conseguir un tipo de movimiento.

5.1.4.- Comunicación

Transmisión de información desde la estación de control en tierra y el vehículo en el aire, normalmente esta comunicación se produce por radio.

5.2.- CLASIFICACIÓN DE UAVS

Normalmente este tipo de vehículos se clasifican según el método de sustentación utilizado, del modo siguiente:

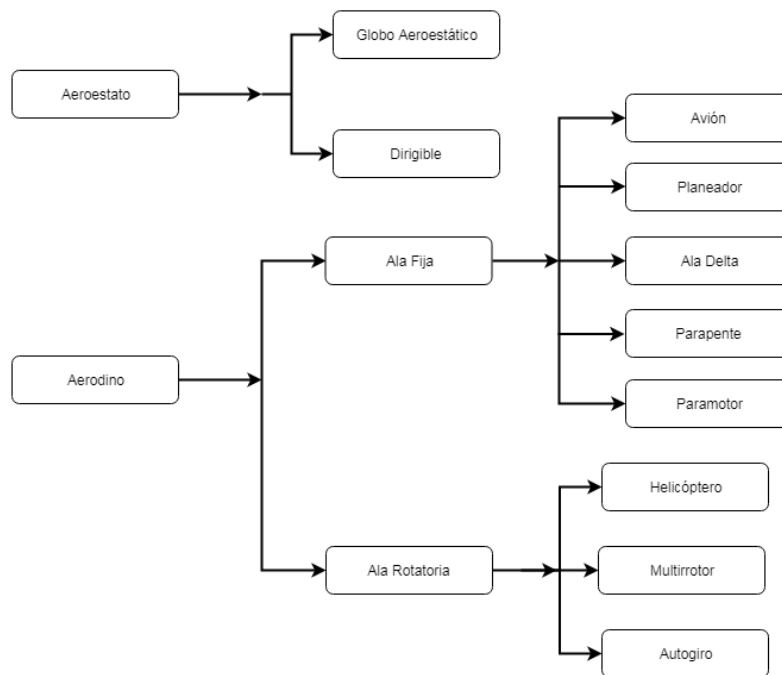


Ilustración 21 Clasificación según el método de sustentación. Fuente: <http://www.xdrones.es/tipos-de-drones-clasificacion-de-drones-categorias-de-drones/> Acceso: 25/4/2018.

Para ambos proyectos se usa el tipo de drones conocido como “multi-rotor”, cuya clasificación se extiende a continuación:

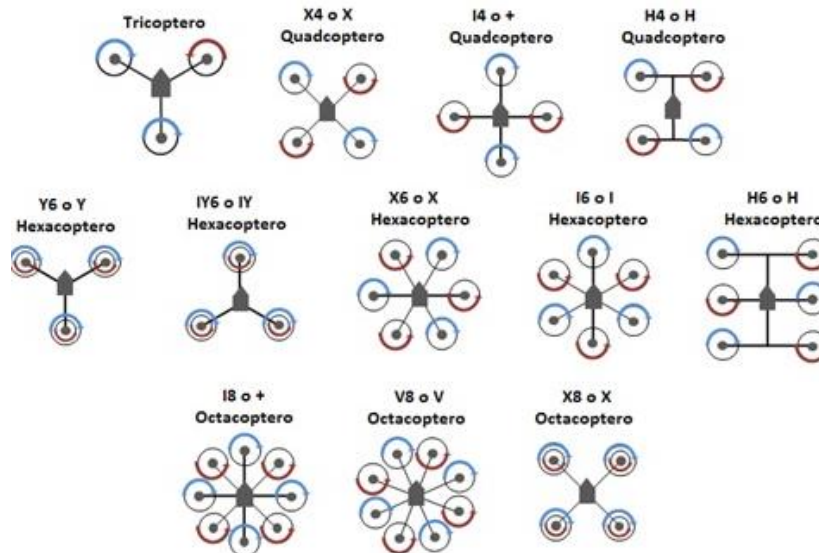


Ilustración 22 Clasificación de multi-rotores. Fuente: <http://www.xdrones.es/tipos-de-drones-clasificacion-de-drones-categorias-de-drones/> Acceso: 25/4/2018.

Más concretamente, el dron del proyecto UAV-Inspection se corresponde con un octacóptero X8, mientras que el del proyecto de las torres de tensión se corresponde con un octacóptero V8.

6. UAV-Inspection

6.1.- DEFINICIÓN DEL SISTEMA PREVIO

6.1.1.- Partes del sistema

A continuación, se presenta un breve resumen de cada uno de los componentes del proyecto.

6.1.1.1.- Controladora de vuelo

Es el elemento encargado de regular la potencia de los motores, computarizar la información recogida por los sensores, estabilizar el vuelo, y en definitiva actuar como el cerebro de la aeronave.

Una controladora de vuelo se compone así mismo de:

- Acelerómetro o medidor de la aceleración.
- Giroscopio ó medidor de la velocidad angular.
- Magnetómetro o brújula.
- Procesador o “cerebro” del dron.

Para la realización del proyecto se optó por el modelo Pixhawk px4.



Ilustración 23 Pixhawk px4. Acceso: 24/4/2018. Fuente: <https://www.google.es>

6.1.1.2.- Raspberry pi

Es un computador de placa reducida, por lo que, dado su pequeño tamaño resulta especialmente interesante en el mundo de los drones.

Para la realización del proyecto se optó por el modelo B de Raspberry pi 3, principalmente por su bajo precio y por la reciente incorporación de módulo Wi-Fi.

Este módulo resulta especialmente interesante para muchas especificaciones del proyecto, como por ejemplo la transmisión de imágenes a la estación de tierra.



Ilustración 24 Raspberry pi3 modelo b. Acceso: 24/4/2018. Fuente: <https://www.google.es>

6.1.1.3.- Sistema de radiocontrol TARANIS

El sistema de radiocontrol consta de dos partes, la primera (el receptor) se conecta directamente con la controladora de vuelo para, de este modo, poder darle órdenes a la aeronave.

La segunda parte consta de un mando, dispositivo que actúa como interfaz con el usuario, a través de la cual se transfieren las órdenes.

Para la realización de este proyecto se optó por el modelo E X9E de la marca Taranis.



Ilustración 25 Taranis-E X9E. Acceso: 24/4/2018. Fuente: <https://www.google.es>

6.1.1.4.- Radio de telemetría

La telemetría es una tecnología que permite el envío de información a distancia al operador del sistema.

El principal uso de este tipo de tecnología en el proyecto es que permite monitorizar el desarrollo en tiempo real de la misión desde la estación de tierra



Ilustración 26 Seguimiento de una misión de vuelo desde la estación de tierra usando el software "Mission Planner". Acceso: 24/4/2018.Fuente: <https://www.google.es>

6.1.1.5.- Cámara RGB

El uso de la llamada cámara RGB, del inglés “Red, Green, Blue” o cámara convencional tiene el objetivo doble de facilitar la inspección visual de los paneles solares y facilitar la comprobación del terreno a inspeccionar.

De este modo ante el descubrimiento de un punto caliente en alguna de las imágenes térmicas, se deberán inspeccionar las imágenes RGB de la misma misión para intentar detectar el posible fallo (rotura, obstrucción...)

Para el presente proyecto se optó por la inclusión del módulo para Raspberry denominado “Raspberry Pi camera module v2”.

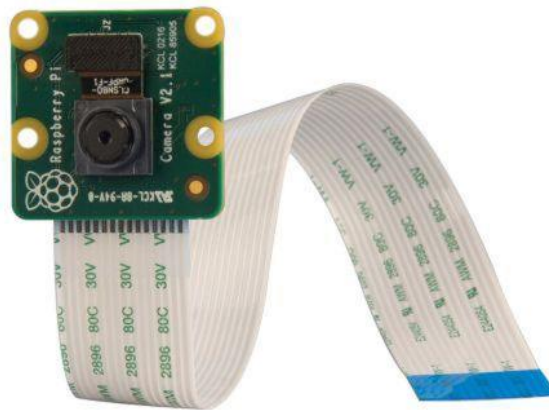


Ilustración 27 "Raspberry pi camera module v2". Acceso: 24/4/2018.Fuente: <https://www.google.es>

:

6.1.1.6.- Cámara térmica

Constituye uno de los elementos más importantes del proyecto, pues, como se comentaba anteriormente resulta esencial para la detección de posibles puntos calientes en las placas solares.

Para el presente proyecto se optó por el uso del modelo VUE PRO de la marca FLIR.



Ilustración 28 Cámara FLIR VUE PRO. Acceso: 24/4/2018. Fuente: <https://www.google.es>

6.1.1.7.- GNSS

Un sistema global de navegación por satélite es una constelación de satélites que permite establecer un sistema de posicionamiento y localización en cualquier parte del globo terrestre.

De este modo se permite con ello determinar las coordenadas geográficas y la altitud de la aeronave donde se instale este dispositivo.

Para el presente proyecto se optó por el modelo 3DR.



Ilustración 29 GNSS ·DR . Acceso: 24/4/2018.Fuente:
<https://www.google.es>

Nota: A lo largo de la realización del presente trabajo se denominará a este componente como GNSS o GPS indistintamente. Siendo GPS la red de satélites norteamericana (la más usada en GNSS).

6.1.1.8.- Chasis

Actúa como un esqueleto para el dron, dando soporte y estructura al resto de los componentes



Ilustración 30 Chasis HMF U580 OPRO Quad. Acceso: 2474/2018.Fuente:
<https://www.google.es>

6.1.1.9.- Hélices

El dron usa cuatro hélices (dos orientadas a la izquierda y dos a la derecha) para permitir su sustentación en el aire.

Se trata del modelo T-motor V2, su fabricación en sándwich de carbono permite una mejor sustentación en un mayor tiempo de vuelo.



Ilustración 31 Hélices T-motor V2. Fuente: <https://www.multicoptero.com/es/tienda-on-line/helices/t-motor-10x3-3-pareja-v2/> Acceso: 01/08/2018.

6.1.1.10.- Motores

Se trata de un motor de corriente eléctrica, se encargan de la transmisión de la energía necesaria para sustentar la aeronave.



Ilustración 32 Navigator MN3508. Acceso: 01/08/2018. Fuente: <https://www.google.es>

6.1.1.11.- Variadores electrónicos

Controlador de velocidad electrónico, define la velocidad de giro de un motor mediante la generación de pulsos.

Un elemento de este tipo tiene como misión controlar en cada momento la posición del rotor. Para así, hacerle llegar la cantidad de corriente requerida para provocar la rotación adecuada.



Ilustración 33 Tmotor AIR. Acceso: 01/08/2018. Fuente: <https://www.google.es>

6.1.1.12.- Batería

Capacita al dron de autonomía a la hora de realizar un vuelo, es necesario encontrar un equilibrio entre potencia de las baterías y peso del dron.

Para el presente proyecto se optó por el modelo Desire Power 8000mAh de la marca RC Innovations, que se muestra en la siguiente imagen:



Ilustración 34 Batería. Fuente: <https://rc-innovations.es/Lipo-bateria-4s-8000mAh-14.8v-30c> Acceso: 25/4/2018.

6.1.2.- Conexiones previas

A continuación se muestra una vista resumida de las conexiones existentes en el dron, sobre las cuales se irán introduciendo nuevos elementos.

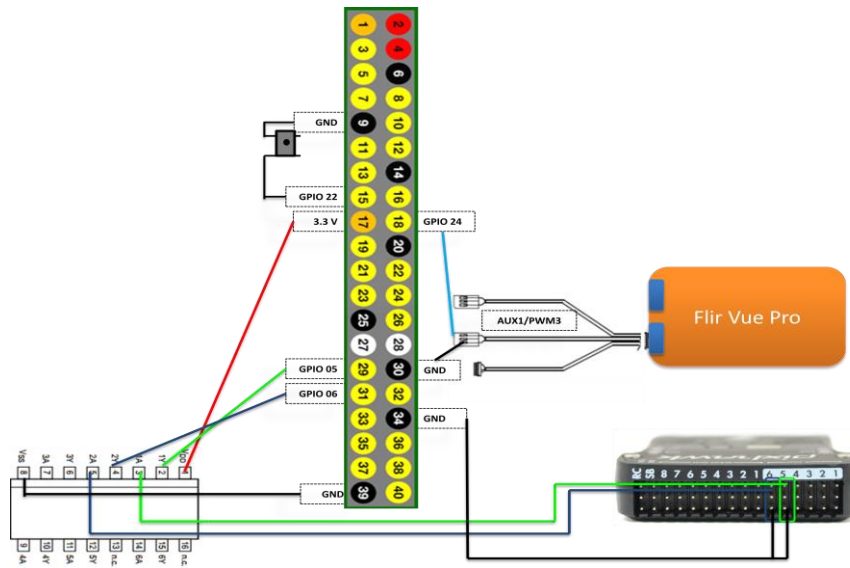


Ilustración 35 Conexiones existentes. Fuente: Servidor Locis Sigtech.

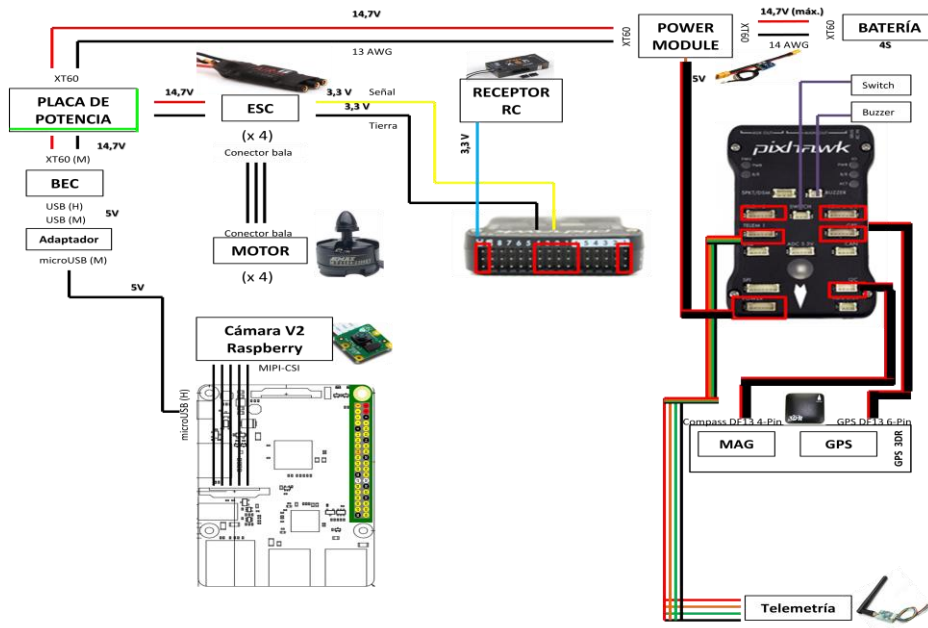


Ilustración 36 Conexiones existentes. Fuente: Servidor Locis Sigtech.

6.2.- SISTEMAS

A continuación se pretenden recoger las mejoras aplicadas al proyecto UAV-Inspection, en concreto:

- | | | | | | |
|-------|-------|--------------------|---------|---------------------------------|-------------------|
| 1-LED | 2-RTK | 3-Información GNSS | 4-Lidar | 5-Envío automático de la misión | 6-Integración FPV |
|-------|-------|--------------------|---------|---------------------------------|-------------------|

7. LED

7.1.- PROBLEMÁTICA EXISTENTE

Después de la realización de una de las misiones de vuelo en el campo de fútbol de TSK Roces el 21/02/2018 se llega a la conclusión de que no se tiene confirmación en campo de si el dron había sacado fotos o no.

Debido a esto se piensa en posibles soluciones:

Una posible solución pasa por el envío de un mensaje email usando la funcionalidad wifi de la RPI. Esta opción es rápidamente descartada debido a la más que probable inexistencia de wifi en las zonas donde se realizan las misiones de vuelo.

Otra posible solución pasaba por instalar un dispositivo acústico conocido como PIEZO, con una implementación sencilla y muy poco peso. Esta solución fue rechazada debido a la excesiva estridencia del aparato.

Finalmente se llegó a la solución de implementar un LED (“light-emitting diode”) para recibir confirmación visual de la existencia de fotos.

¿Qué es un LED?

Se conoce como diodo emisor de luz o “light-emitting diode” y se compone de una fuente de luz constituida por un semiconductor dotado de dos terminales.

Sigue un sistema de funcionamiento basado en un diodo de unión p-n que emite luz cuando esta activado.

Al aplicarse una tensión en los terminales se produce una recombinación de los electrones con los huecos en la región de la unión p-n del dispositivo, liberando energía en forma de fotones. Por medio de un efecto denominado electroluminiscencia.

Los colores generados al alimentar el dispositivo dependen de la energía de los fotones emitidos y la anchura de banda del semiconductor.

En el presente caso se elige el LED rojo por comodidad en su visualización.

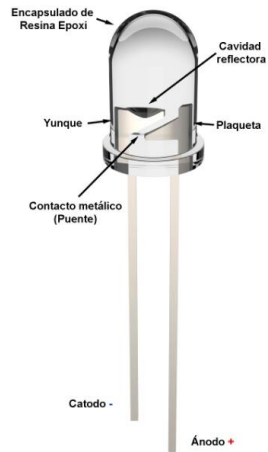


Ilustración 37 Esquema de un LED.
Fuente:<https://www.google.es>

7.2.- CONEXIÓN HARDWARE

La conexión hardware implica el uso de cables de conexión hembra-hembra, una resistencia de 220 ohmios y el propio LED.

Se conectará el terminal positivo del LED con el pin número 27, que emitirá tensión en su estado HIGH y dejará de emitirla en su estado LOW.

El terminal negativo del LED se conectará al pin 25 según designación BCM, dicho pin se comporta como GROUND.

Como es lógico la resistencia se conecta entre el pin de salida y el terminal positivo del LED, para mitigar así el paso de corriente eléctrica y proteger al LED.

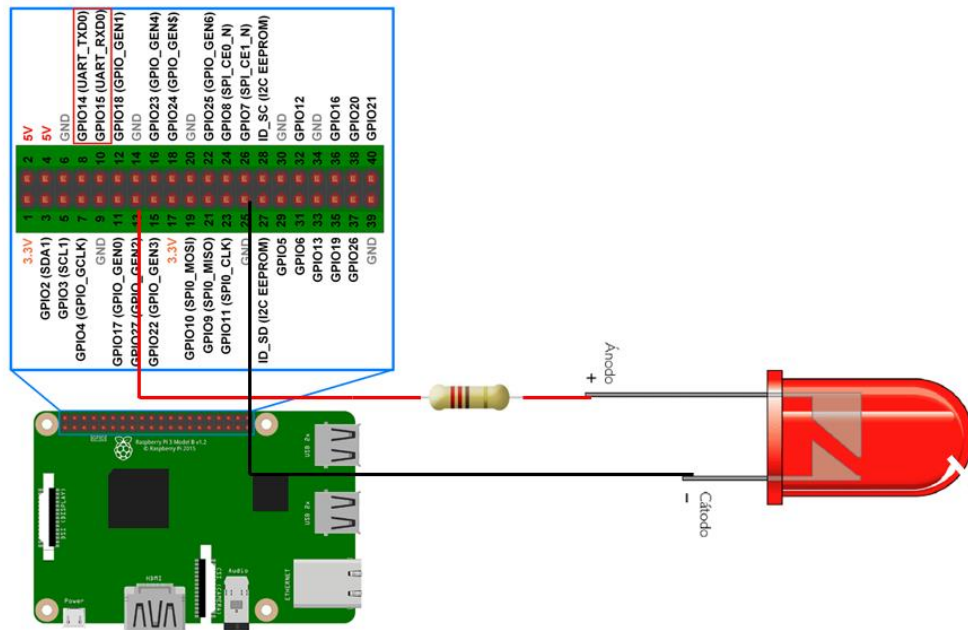


Ilustración 38 Esquema de conexión física de RPI con LED. Fuente: Elaboración propia.

El montaje en el propio dron sigue el esquema expuesto abajo, del siguiente modo:



Ilustración 3 Detalle de conexión de LED, apagado VS encendido. Fuente: Elaboración propia.

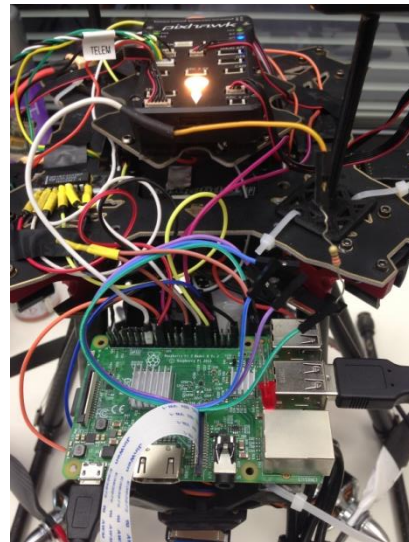


Ilustración 3 Detalle de conexión de LED, apagado VS encendido. Fuente: Elaboración propia.

7.3.- SOFTWARE

El código usado en esta solución es el siguiente:


```
gpio.setmode(gpio.BCM) #designacion de pines GPIO en modo BCM
buzzer_pin=27
gpio.setup(buzzer_pin, gpio.OUT)
gpio.output(buzzer_pin, gpio.LOW)
```

Ilustración 39 modos de los pines de la Raspberry. Fuente: Elaboración propia.

Primero se designa la forma de contar los pines GPIO en formato BCM, posteriormente asignamos a la variable `buzzer_pin` el valor 27.

De este modo el GPIO número se comportará como un output inicializado en LOW.

Posteriormente se define la función encargada del encendido y apagado del LED:

```
def beep ():
    global buzzer_pin
    gpio.output(buzzer_pin,gpio.HIGH)
    time.sleep(2)
    gpio.output(buzzer_pin,gpio.LOW)
    time.sleep(2)
```

Ilustración 40 función encargada de encender el LED. Fuente: Elaboración propia.

La función `beep`, una vez llamada, ilumina el LED (con el output del pin 27 en HIGH). Se mantiene el LED encendido durante 2 segundos para luego apagarlo durante otros dos.

En la rutina de interrupción se implementa un bucle encargado de llamar a la función `beep`, produciendo así la intermitencia luminosa:

- `while (i != 0):`
 - `beep()`

La variable `i` permite saber si se ha realizado disparo en la rutina correspondiente, de este modo cuando `i=0` implica que ninguna foto fue tomada. Mientras que en caso de que se entre al bucle, `i` es distinto de cero, el LED se encenderá. Recibiendo el usuario confirmación lumínica.

7.3.1.- Optimización del código

El LED solamente se encendía al detectar órdenes de disparo, no siendo de este modo todo lo eficaz que podría ser.

Se pretende realizar el encendido intermitente solamente cuando detecte la existencia de fotos. De los dos tipos de fotos de los que disponemos se dará más importancia a las imágenes térmicas, por lo que se pretende que el LED brille cuando detecte imágenes de este tipo.

Para ello se implementará un subprograma Python, al cual se llama desde el programa principal `automatico.py` cuando se aplica la rutina de interrupción, mediante el siguiente comando:

```
os.system('sudo python /home/pi/Desktop/CarpetaCompartidaPRUEBA/localizacion_fotos.py')
```

El subprograma `localizacion_fotos.py` se muestra en el siguiente conjunto de imágenes:

```

#-*- coding: utf-8 -*-
import RPi.GPIO as gpio
import time
import os
import sys
import numpy as np
x=0
#Realiza un listado de fotos en la ruta especificada en path

gpio.setmode(gpio.BCM)

gpio.setup(27, gpio.OUT)
path= '/media/pi'
lstFile=[]
lstDir=os.walk(path)
for root, dirs, files in lstDir:
    for fichero in files:
        (nombreFichero, extension) = os.path.splitext(fichero)
        if(extension == ".jpg"):
            lstFile.append(nombreFichero+extension)
print (lstFile)
print ('listado finalizado')
print "longitud de la lista =" ,len(lstFile)

```

Ilustración 41 localizacion_fotos.py parte 1. Fuente: Elaboración propia.

```

ru='/home/pi/Desktop/CarpetaCompartidaPRUEBA/informes_fotos_FLIR/'+time.strftime('%d-%b-%y')+'.txt'
fi=open(ru,"w")
for x in range (0,len(lstFile)):
    print lstFile[x]

if len(lstFile) > 0:
    print ('existen %s fotos FLIR en la ruta especificada, se ilumina el LED' %len(lstFile))
    fi=open(ru,"w+")
    fi.write("existen %s fotos FLIR que se copiaron a la carpeta compartida, si no mirar en la FLIR" %len(lstFile))
    for x in range (0,len(lstFile)):
        gpio.output(27,gpio.HIGH)
        time.sleep(5)
        gpio.output(27,gpio.LOW)
        time.sleep(5)
else:
    print "Existen cero fotos"
    fi=open(ru,"w+")
    fi.write("Nunca se sacaron fotos ó existen cero fotos en la camara FLIR")

gpio.cleanup()

```

Ilustración 42 localizacion_fotos.py parte 1. Fuente: Elaboración propia.

Primero se busca dentro de la ruta especificada en “path” (la ruta de la cámara FLIR) los archivos con extensión .jpg. Se produce de este modo un vector que contiene los nombres de todas las fotos dentro de la memoria de la cámara FLIR.

Contamos posteriormente la longitud de dicho vector (o lo que es lo mismo la cantidad de fotos térmicas de la misión).

En caso de que la longitud de dicho vector sea mayor que cero (que se hayan sacado más de cero fotos) se entrara en la rutina del Sí lógico.

En donde se genera un archivo .txt en la ruta /home/pi/Desktop/CarpetaCompartidaPRUEBA/informes_fotos_FLIR que anuncia el número de fotos de este tipo de las que se dispone, después hace parpadear el LED a intervalos de cinco segundos un tiempo proporcional a la longitud del vector de fotos.

En caso de no tener ninguna foto FLIR se entrará en la rutina del No lógico, en donde se generará un informe en la ruta antes mencionada que nos advertirá de la inexistencia de fotos. Además el LED no brillará.

Debido a que es probable que se realicen varias misiones de vuelo en el mismo día se actualizo el archivo generado para renombrarse con la hora y los minutos del final de misión, del modo:

 27-04-18-13-22.txt	27/04/2018 13:22
 27-Apr-18.txt	27/04/2018 12:54

Ilustración 43 Notación del fichero generado. Fuente: Elaboración propia.

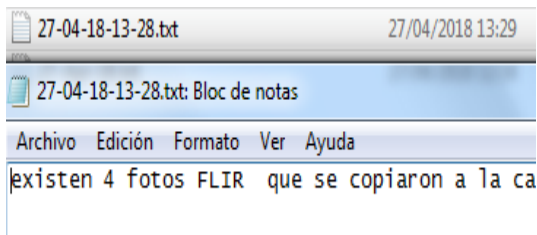


Ilustración 44 Informe generado. Fuente: Elaboración propia.





 20180427_122614.jpg	27/04/2018 13:29	Archivo JPG
 20180427_122616.jpg	27/04/2018 13:29	Archivo JPG
 20180427_122619.jpg	27/04/2018 13:29	Archivo JPG
 20180427_122620.jpg	27/04/2018 13:29	Archivo JPG

Ilustración 45 Fotos térmicas en la carpeta compartida. Fuente: Elaboración propia.

8. Envío automático de la misión

8.1.- INTRODUCCIÓN

Cualquier proyecto requiere en menor o mayor medida de cierto grado de automatización. Debido a que muchas veces la actuación humana provoca errores que rara vez son cometidos por una máquina, o peca de excesiva ineficiencia.

Por esto es de vital importancia ser capaces de limitar al máximo la intervención humana, sobre todo en proyectos de monitorización de actividades industriales, en donde un simple fallo puede acarrear un parón en la producción o grandes pérdidas en concepto de maquinaria.

En el presente documento se detalla la automatización del envío de la misión de vuelo de la Raspberry pi 3 a la controladora de vuelo PIXHAWCK.

De este modo la misión generada desde la estación Big data se transferirá en formato .WAYPOINTS a la carpeta compartida entre el PC/ET (estación de tierra) y la propia Raspberry Pi.

Para mayor facilidad y menor error, la misión de vuelo, compuesta por una sucesión de “Waypoints” o puntos que debe seguir el dron, se carga en la controladora de vuelo PIXHAWCK de forma automática al arrancarse esta.

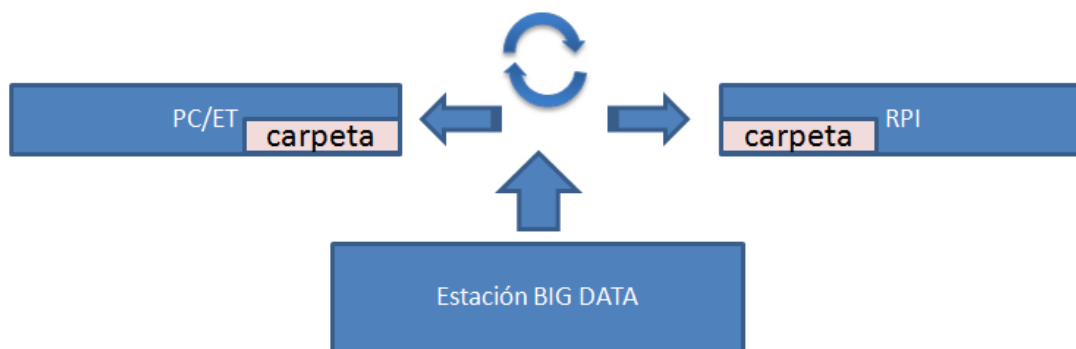


Ilustración 46 Sincronización misión de vuelo. Fuente: Elaboración propia.

La extensión del archivo contribuye a mayor facilidad de uso dado que el software usado para planificar el vuelo (“Mission Planner”) usa ese tipo de formatos.

8.2.- HARDWARE

Cualquier sistema informático o electrónico o en general cualquier sistema tiene un cierto nivel de requerimiento en cuanto a partes físicas o “hardware” que secunden todo el sistema de subrutinas o programas (“software”).

Para el caso concreto del envío de la misión de vuelo desde la Raspberry a la controladora es necesaria una conexión física entre ambas. En este caso, las opciones más populares son:

- Conexión USB.
- Conexión directa usando los pines GPIO.

En la primera opción simplemente se usa un adaptador, bien sea comprado o fabricado, entre la entrada de telemetría dos (TELEM2) y la conexión USB de la Raspberry pi, del modo siguiente:



Ilustración 47 Conexión controladora Raspberry vía USB. Fuente: <https://vueloartificial.com/1-conexion-vuelo-controlado-mediante-raspberry-pi/>

En la segunda opción se opta por una conexión directa con los pines de la Raspberry, este método aporta más fiabilidad al conexionado y en general es el más recomendado.

Los GPIO (del inglés general purpose input/output o entradas/salidas de propósito general) son un conjunto de pines genéricos en un chip, una tableta Raspberry pi 3 en este caso, cuyo comportamiento (incluyendo si se trata de un pin de entrada o salida) se puede controlar o programar en tiempo de ejecución.

Desde su irrupción en el mercado, el modelo Pi 3 de Raspberry cuenta con 40 de estos pines distribuidos de la forma que se aprecia en la imagen:

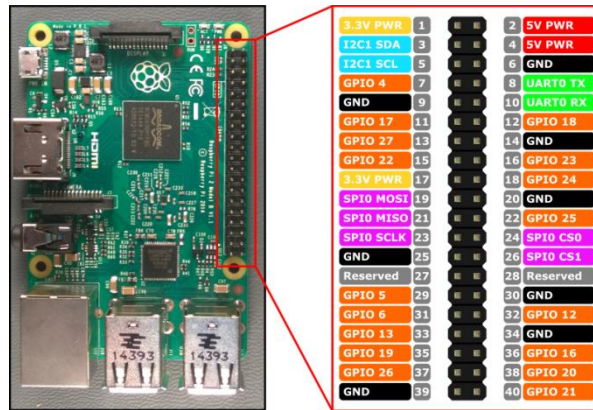


Ilustración 48 Pines GPIO de la Raspberry pi 3. Fuente: <https://www.programoergosum.com/cursos-online/raspberry-pi/238-control-de-gpio-con-python-en-raspberry-pi/intermitente>

La esquemática de la conexión de este método, que fue la implementada en el proyecto, se resume en la siguiente imagen:

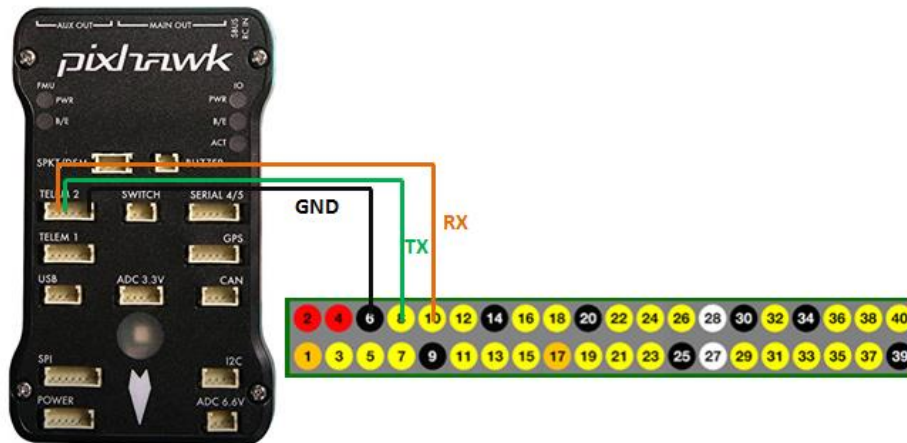


Ilustración 49 Esquema de conexiones Raspberry pi-Pixhawk. Fuente: Elaboración propia.

Es conveniente destacar, y sobre todo si se han seguido los manuales de conexión de la Raspberry con la controladora encontrados en la página web de Ardupilot (Ardupilot.org), que según la disposición de los componentes del sistema que forma el proyecto existen dos modos de dar energía a la Raspberry para permitir su encendido:

- Uno es mediante la conexión mini-USB que trae la tableta de serie. Es el modo más común de encendido y el que actualmente se usa en el proyecto:

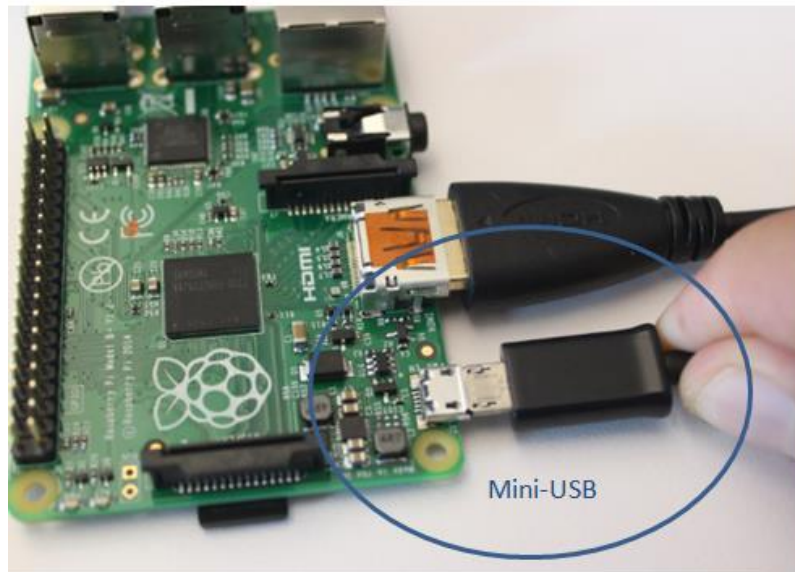


Ilustración 50 Alimentación de la Raspberry pi vía mini-USB. Fuente: <https://www.peworld.com/article/2598363/how-to-set-up-raspberry-pi-the-little-computer-you-can-cook-into-diy-tech-projects.html>

- El otro método es mediante los pines GPIO de la Raspberry, este método requiere añadir una nueva conexión física desde el primer pin del puerto de telemetría dos hasta el pin destinado a la entrada de voltaje de los GPIO de la Raspberry, del modo siguiente:

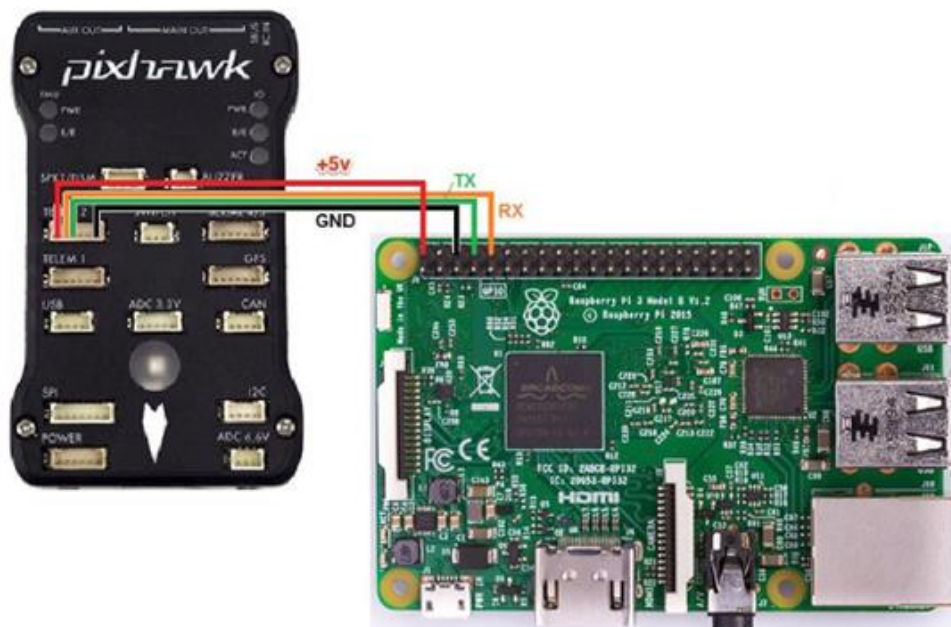


Ilustración 51 Alimentación de la Raspberry pi usando la controladora. Fuente: Elaboración propia.

Como es obvio, la controladora en este caso debe ser la que reciba voltaje de algún generador, presumiblemente una batería.

8.3.- SOFTWARE

Después de tener la conexión física es necesario un conjunto de rutinas o programas que realicen las tareas digitales que comande el usuario, para ello es necesario hablar de la implementación software.

En este apartado se detallará la fase referida a la programación del envío automático de la misión y de la comunicación de la controladora con la Raspberry, para lo que será necesario profundizar primero en el arranque y jerarquía de los ficheros de sistemas operativos basados en Linux.

8.3.1.- Introducción al “boot process” de Linux (“startup sequence”)

Cualquier dispositivo con un sistema operativo Linux o alguna de sus distribuciones usa un sistema de encendido basado en seis fases, en concreto:



Ilustración 52 Linux "Boot Process". Fuente: Elaboración propia.

Cada uno de los niveles ejecuta opciones esenciales y da paso al siguiente, de arriba a abajo.

En el caso concreto de la Raspberry Pi del proyecto de UAV-Inspection se usa Raspbian como sistema operativo (una de las distribuciones de Linux, especialmente diseñada para el uso de Raspberry).

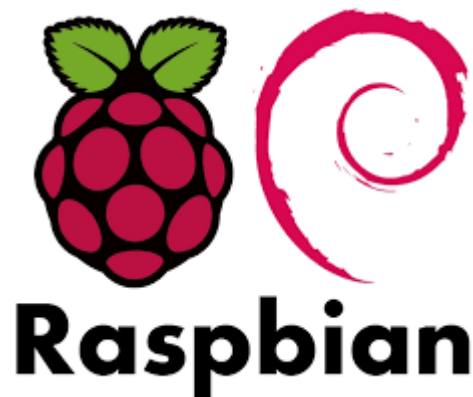


Ilustración 53 Logo de Raspbian. Fuente: Google.

En el proceso de arranque es posible establecer “scripts” (líneas de código que realizan una actividad) para que se ejecuten en el inicio.

De este modo, primero se elegiría el nivel en el cual querríamos ejecutar el “script” de forma automática. Se optó por el Runlevel, más concretamente el directorio `/etc`.

Dentro del sistema de ficheros y archivos Linux el directorio `etc` de “edit to configure” ó “editar para configurar” contiene todos los archivos responsables de la configuración del sistema operativo.

Un archivo de configuración simplemente se define como un archivo local usado para el control de las operaciones de un programa.

Dentro del directorio `/etc` encontramos muchos otros directorios y carpetas, como por ejemplo `rc.local` (ubicado en la ruta `/etc/rc.local`) que contiene archivos del tipo “daemon” (del inglés Disk And Execution Monitor) que permite la invocación de otros programas en la secuencia de arranque.

Un “daemon” consiste en un tipo especial de proceso informático no interactivo, es decir ejecutado en segundo plano y sin interacción con el usuario que continua en el sistema y establece las condiciones de arranque de otros programas.

8.3.2.- Programación

Para permitir la ejecución de ciertos programas es necesario editar el archivo `daemon rc.local`.

```

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

exec > /tmp/rc-local.out 2>&1; set -x

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

/home/pi/startup.sh
#/home/pi/test > /home/pi/test.log 2>&1 &

exit 0

```

Ilustración 54 Fichero /etc/rc.local Fuente: Elaboración propia.

Este archivo, que por defecto no hace nada (como se puede leer en los comentarios) se ha editado para que:

- Ejecute un archivo rc-local.out en el directorio /tmp : esta línea permite visualizar opciones de “debug” o depuración de errores
- Imprima la dirección de la propia Raspberry, que se puede ver en el archivo rc-local.out : esta línea únicamente tenía como objetivo la realización de pruebas
- Lanza el ejecutable startup.sh dentro del directorio /home/pi
- Al final devuelve un cero para poder continuar con el siguiente Runlevel

El ejecutable startup.sh al que se hace referencia dentro de rc.local es un “bash script”. Bash (del inglés Bourne-again Shell). Es un programa informático encargado de interpretar órdenes mediante consola. Constituye el intérprete de comandos más usados por las aplicaciones Linux, aunque ahora también está disponible en Windows y Android.

```
chris@ubuntu: ~
chris@ubuntu:~$ bash --version
GNU bash, version 4.3.46(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
chris@ubuntu:~$
```

Ilustración 55 Ejemplo de consola de comandos BASH. Fuente: Google.

```
#!/bin/sh
exec 5> debug_output.txt
BASH_XTRACEFD="5"
screen -s /bin/bash -d -m python /home/pi/RPI_to_PIX_xxx.py
```

Ilustración 56 startup.sh llamado desde /etc/rc.local. Fuente: elaboración propia.

Dentro del citado tipo de programa es importante incluir un “Shebang”. Este es el nombre que, en jerga Unix, recibe la sucesión de caracteres ubicados en la línea uno.

La importancia del “Shebang” radica en que permite interpretar el programa como ejecutable por uno de los intérpretes conocidos, haciendo, en este caso referencia al intérprete de comandos Bash.

Existen multitud de Shebang diferentes como por ejemplo `#!/usr/bin/perl` ó `#!/usr/bin/python`, cada uno haciendo referencia a su propio intérprete. Siendo el más común el referido a Bash.

Es muy importante que además de usar el “Shebang” correspondiente se convierta el archivo en ejecutable usando el comando:

- `chmod +x fichero`, o, en este caso

```
pi@raspi_locis:~ $ chmod +x startup.sh
```

Ilustración 57 Convertir el fichero en ejecutable. Fuente: Elaboración propia.

Es importante ubicarse dentro de la ruta donde se encuentra el archivo al que queremos llamar, o, en su defecto usar una referenciación absoluta a dicha ruta.

Se puede comprobar que este comando se llevó a cabo de forma correcta ubicándose en el directorio correspondiente y tecleando por pantalla:

- `ls -l`

De esta manera se muestra en formato de lista los diferentes elementos dentro del directorio actual, en el presente caso:

```
-rw-r--r-- 1 root root 1294 abr 10 13:26 RPI_to_PIX_xxx.py
-rwxr-xr-x 1 pi pi 612 abr 9 12:08 screen_mavproxy.py
-rw----- 1 pi pi 5273 mar 29 2017 sent
-rw-r--r-- 1 pi pi 327 mar 31 2017 shutdown.py
-rwxr-xr-x 1 pi pi 115 abr 10 12:25 startup.sh
drwx----- 3 pi pi 4096 mar 15 17:23 Sync
drwxr-xr-x 2 pi pi 4096 nov 25 2016 Templates
-rwxr-xr-x 1 pi pi 55 abr 3 11:58 test
-rw-r--r-- 1 root root 15 abr 6 13:33 test.dat
```

Ilustración 58 Listado de archivos en el directorio /home/pi de la RPI. Fuente: elaboración propia.

En el listado de arriba se puede ver como `startup.sh` es un archivo que admite lectura y escritura (marcados con `r` y `x` respectivamente, del inglés “read” “write”, y además es ejecutable (marcado con la `x` del inglés “executable”), a diferencia de otros archivos.

Además también podemos ver cómo es un archivo propiedad del usuario “pi” a diferencia de otros archivos que solo pueden ser accedidos con privilegios “root” o de administrador (son propiedad de este súper usuario).

Si en algún momento quisiéramos revertir la propiedad de ser ejecutable del archivo se teclearía por consola:

- `chmod -x fichero`

El archivo `startup.sh` presenta una nomenclatura muy sencilla:

- Primero, el ya citado “Shebang” encargado de llamar al intérprete. Es especialmente importante que esta línea se mantenga en la cabecera del programa y sin ningún tipo de espacios
- Las líneas dos y tres corresponden a comandos de “debugging” o depuración de errores, generando un `debug.txt`
- La última línea es la encargada de la llamada a la función “screen” responsable de una multiplexación de terminal en donde se ejecutará el programa python `RPI_to_PIX_xxx.py`

8.3.3.- Multiplexación de terminal

`RPI_to_PIX_xxx.py` llama a la ejecución de otro programa denominado `Mavproxy.py`

Dicho programa, en el que se profundizará más adelante, consiste en un bucle infinito diseñado para dar comandos por teclado que serán ejecutados por la controladora.

Debido al diseño de este programa y su forma de uso común, se requiere una interacción constante humano-máquina. Es decir el operario introducirá comandos del estilo “ARM” (para realizar el armado o comprobación de la disponibilidad sin errores), “PARAM SHOW” (para mostrar por pantalla los diferentes parámetros de la controladora) u otros.

Command	Δ	Value	Units	Options
ACCEL_Z_P		0,5		0.500 1.500
ACRO_BAL_PITCH		1		0 3

Ilustración 59 Lista de parámetros visto desde la planificadora de vuelo (Mission Planner). Fuente: Elaboración propia.

Dentro del bucle infinito que corresponde a mavproxy.py podemos ejecutar comandos, como se puede ver en la siguiente imagen:

```

SR3_POSITION      2.000000
SR3_RAW_CTRL      2.000000
SR3_RAW_SENS      2.000000
SR3_RC_CHAN       2.000000
STAT_BOOTCNT      211.000000
STAT_FLTTIME      3304.000000
STAT_RESET        59839848.000000
STAT_RUNTIME      410930.000000
SUPER_SIMPLE      0.000000
SYSID_ENFORCE     0.000000
SYSID_MYGCS       255.000000
SYSID_SW_MREV     120.000000
SYSID_SW_TYPE     10.000000
SYSID_THISMAV    1.000000
TELEM_DELAY       0.000000
TERRAIN_ENABLE    1.000000
TERRAIN_FOLLOW    0.000000
TERRAIN_SPACING   100.000000
THROW_MOT_START   0.000000
THROW_NEXTMODE    18.000000
THROW_TYPE        0.000000
THR_DZ            100.000000
TUNE              0.000000
TUNE_HIGH         1000.000000
TUNE_LOW          0.000000
VEL_XY_FILT_HZ    5.000000
VEL_XY_I          0.500000
VEL_XY_IMAX       1000.000000
VEL_XY_P          1.000000
VEL_Z_P           5.000000
VISO_ORIENT       0.000000
VISO_POS_X        0.000000
VISO_POS_Y        0.000000
VISO_POS_Z        0.000000
VISO_TYPE         0.000000
WPNAV_ACCEL       100.000000
WPNAV_ACCEL_Z     100.000000
WPNAV_LOIT_JERK   1000.000000
WPNAV_LOIT_MAXA   250.000000
WPNAV_LOIT_MINA   25.000000
WPNAV_LOIT_SPEED  500.000000
WPNAV_RADIUS      100.000000
WPNAV_RFND_USE    1.000000
WPNAV_SPEED       500.000000
WPNAV_SPEED_DN    150.000000
WPNAV_SPEED_UP    250.000000
WP_NAVALT_MIN     0.000000
WP_YAW_BEHAVIOR   2.000000

```

Ilustración 60 Resultado del comando "param show" en la terminal multiplexada en la que se ejecuta mavproxy.py. Fuente: Elaboración propia.

Pero como se introdujo con anterioridad, esta interacción constante es justo lo que se pretende evitar en el presente proyecto, asimismo el "script" que controla las cámaras se basa en otro bucle infinito. De modo que se opta por la solución de abrir varios terminales de comunicación.

En el terminal nuevo, se ejecutará el programa Mavproxy.py, al que solo se recurrirá para el envío automático de misión, dejando así el terminal original disponible para la ejecución del script automático.

8.3.4.- Mavproxy.py

Mavproxy es una GCS (del inglés Ground control station), escrita en el lenguaje de programación Python, especialmente diseñada por la universidad australiana de Camberra para su uso en UAVs.

Es capaz de, a través de una interfaz basada en comandos escritos en inglés, generar un mensaje MAVLink, entendible por la controladora de vuelo, produciéndose así la comunicación humano máquina.

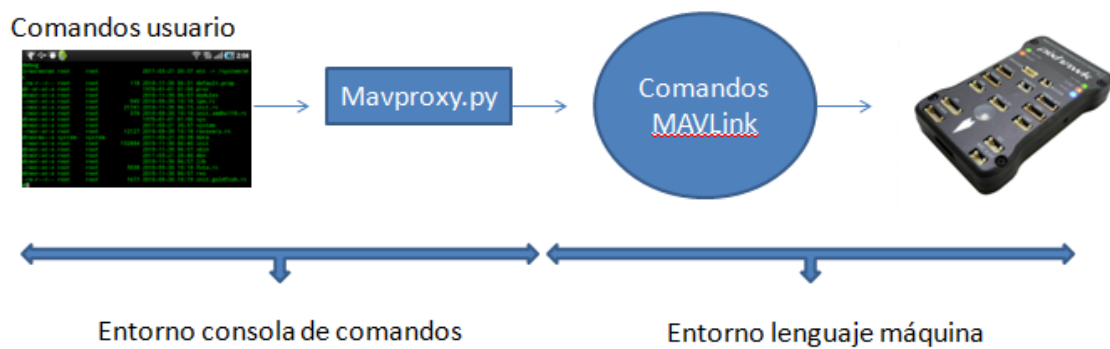


Ilustración 61 Interacción humano-máquina. Fuente: Elaboración propia.

8.3.5.- MAVLink

MAVLink (del inglés “Micro Air Vehicle Link” o conexión entre micro vehículos aéreos) es un protocolo de comunicación usado con vehículos no tripulados de tamaño reducido.

Este protocolo es mayormente usado para establecer una comunicación entre una GCS o estación de control de tierra y cualquier tipo de vehículos no tripulados, con el objetivo de transmitir comandos, orientaciones, velocidades o Waypoints.

8.3.6.- ¿Cómo se realiza la comunicación usando el protocolo MAVLink? Estructura de un paquete MAVLink.

En informática un paquete se refiere a una unidad de datos que se pone en ruta a través de un origen y un destino en Internet, o cualquier otra red.

Cuando cualquier archivo o comando se envía de un lugar a otro se divide entre partes de la forma más eficiente posible para el encaminamiento.

Cada uno de estos paquetes de información se numera por separado y contiene la dirección de su destino.

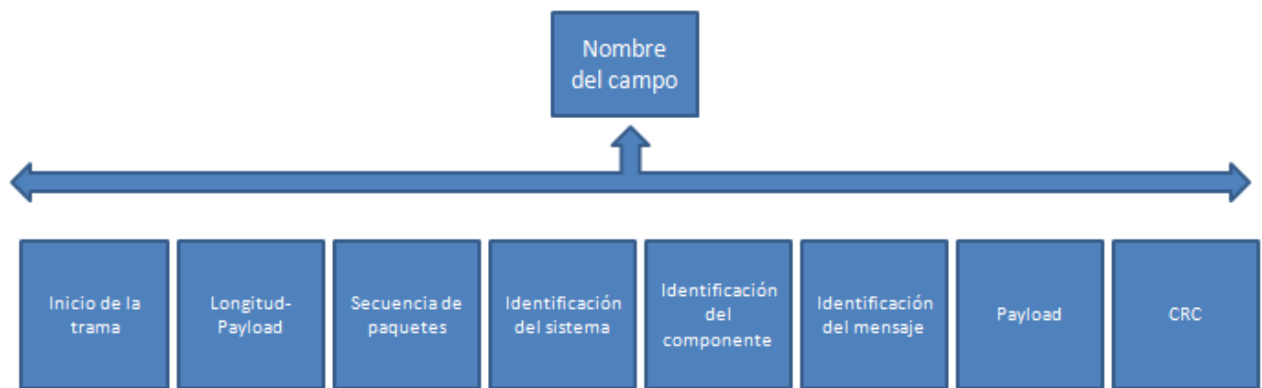


Ilustración 62 Estructura de un paquete MAVLink. Fuente: Elaboración propia.

Los paquetes de un mismo origen pueden viajar por diferentes rutas hasta llegar a su destino en donde se reunificarán para formar el archivo o comando correspondiente.

8.3.7.- Instalación de Mavproxy y GNU screen

Para poder acceder a la comunicación MAVLink a través de mavproxy.py primero debemos descargarlo, para ello será necesario introducir en la consola de comandos de la Raspberry la siguiente línea:

```
root@raspi_locis:/home/pi# sudo apt-get install python-dev python-opencv python-wxgtk3.0 python-pip python-matplotlib python-pygame python-lxml python-yaml
```

Ilustración 63 Instalación de mavproxy.py dentro de la Raspberry. Fuente: Elaboración propia.

Otra opción es obtener el código directamente de un web repositorio como GitHub, plataforma líder mundial en desarrollo de software. En este caso habría que descargarse el .txt correspondiente y hacer un archivo python, al que se llamaría posteriormente desde alguno de los directorios de la Raspberry.

Para poder realizar la multiplexación es necesario descargar el programa Screen, un software usado para multiplicar una consola física entre varios procesos (normalmente Shell interactivas).

El proceso de instalación de este software es sencillo, simplemente lanzaremos la siguiente línea de códigos desde la consola de comandos de la Raspberry pi:

```
root@raspi_locis:/home/pi# sudo pip install screen
```

Ilustración 64 Instalación del software screen en la Raspberry pi. Fuente: Elaboración propia.

Su utilidad reside mayormente en los siguientes casos:

- Permite manejar varios programas desde la consola de comandos
- Permite separar los programas ejecutados desde el Shell

- Permite compartir las sesiones con otros usuarios

Algunos de los comandos más importantes usados por la función son:

- `screen -ls`: muestra las multiplexaciones activas, así como su estado.

```
pi@raspi_locis:~ $ screen -ls
There is a screen on:
      1693..raspi_locis      (11/04/18 11:30:13)      (Detached)
1 Socket in /var/run/screen/S-pi.
```

Ilustración 65 Comando `screen -ls` con multiplexaciones activas. Fuente: Elaboración propia.

- `screen -x nombre`: conecta con la terminal especificada, en caso de solo haber dos activas el nombre es redundante pues se cambiaría de una a otra.

```
root@raspi_locis:/home/pi# screen -x
```

Ilustración 66 Comando `screen -x` por consola. Fuente: Elaboración propia.

Al teclear este comando entramos en la terminal multiplexada, que está ejecutando `mavproxy.py` para subir de forma automática una misión de vuelo:

```
Inicio de programa
Connect /dev/ttyAMA0 source_system=255
no script Registro_de_vuelo/mavinit.scr
Loaded 15 waypoints from /Registro_de_vuelo/wp/xxx.WAYPOINTS
Log Directory: Registro_de_vuelo/logs/2018-04-11/flight3
Telemetry log: Registro_de_vuelo/logs/2018-04-11/flight3/flight.tlog
Waiting for heartbeat from /dev/ttyAMA0
ACGot MAVLink msg: MISSION_ACK {target_system : 255, target_component : 0, type : 0}
Sent waypoint 0 : MISSION_ITEM {target_system : 0, target_component : 0, seq : 0, frame : 0, com
0, param4 : 0.0, x : 43.523037, y : -5.609175, z : 40.0}
online system 1
ALT_HOLD> Mode ALT_HOLD
Sent waypoint 1 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 1, frame : 10, com
.0, param4 : 0.0, x : 0.0, y : 0.0, z : 20.0}
Sent waypoint 2 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 2, frame : 10, com
.0, param4 : 0.0, x : 43.523037, y : -5.609175, z : 40.0}
Sent waypoint 3 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 3, frame : 10, com
.0, param4 : 0.0, x : 43.522982, y : -5.610008, z : 40.0}
Sent waypoint 4 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 4, frame : 10, com
.0, param4 : 0.0, x : 43.522982, y : -5.610008, z : 30.0}
Sent waypoint 5 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 5, frame : 10, com
0.0, param4 : 0.0, x : 43.522982, y : -5.610008, z : 30.0}
Sent waypoint 6 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 6, frame : 10, com
.0, param4 : 0.0, x : 43.523062, y : -5.609912, z : 30.0}
Sent waypoint 7 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 7, frame : 10, com
.0, param4 : 0.0, x : 43.523157, y : -5.609818, z : 30.0}
Sent waypoint 8 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 8, frame : 10, com
.0, param4 : 0.0, x : 43.523274, y : -5.609801, z : 30.0}
Sent waypoint 9 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 9, frame : 10, com
.0, param4 : 0.0, x : 43.523277, y : -5.609802, z : 30.0}
Sent waypoint 10 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 10, frame : 10, c
0.0, param4 : 0.0, x : 43.523277, y : -5.609802, z : 40.0}
Sent waypoint 11 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 11, frame : 10, c
0.0, param4 : 0.0, x : 43.523037, y : -5.609175, z : 40.0}
Sent waypoint 12 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 12, frame : 10, c
: 0.0, param4 : 0.0, x : 0.0, y : 0.0, z : 0.0}
Sent waypoint 13 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 13, frame : 10, c
: 0.0, param4 : 0.0, x : 43.523037, y : -5.609175, z : 40.0}
Sent waypoint 14 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 14, frame : 10, c
0.0, param4 : 0.0, x : 43.523037, y : -5.609175, z : 40.0}
Sent all 15 waypoints
Got MAVLink msg: MISSION_ACK {target_system : 255, target_component : 0, type : 0}
APM: Flight plan received
```

Ilustración 67 Multiplexación de terminal. Fuente: Elaboración propia.

8.3.8.- RPI_to_PIX_xxx

Corresponde a un programa escrito en python que se encarga de la ejecución en otra terminal de mavproxy.py. Se pueden ver las líneas que lo componen en la siguiente imagen:

Es conveniente aclarar que el comando sudo simplemente se encarga de realizar la acción que se le escribe a continuación llamando al usuario su, es decir con privilegios de súper usuario.

```
pi@raspi_locis:~ $ sudo -s
root@raspi_locis:/home/pi#
```

Ilustración 68 Usuario pi vs usuario root. Fuente: Elaboración propia.

Con este comando se consigue control de los archivos propiedad del súper usuario, como también del usuario pi (usuario estándar).

```
import os
import sys
import subprocess
import pasar_ways
import time
os.system('sudo -s exec > /home/pi/RPI_to_PIX.out 2>&1; set -x')
time.sleep(20)
print("Inicio de programa")
os.system(' sudo -s mavproxy.py --master=/dev/ttyAMA0 --aircraft Registro_de_vuelo --cmd="wp load /Registro_de_vuelo/wp/xxx.WAYPOINTS; "')
```

Ilustración 69 RPI_to_PIX_xxx.py .Fuente: elaboración propia.

Este programa, escrito en el lenguaje de programación Python realiza las siguientes acciones:

Primero llama a las librerías necesarias para su correcta ejecución. Una librería en Python, o cualquier otro lenguaje de programación consiste en conjuntos pre-escritos de código que aportan ciertas funcionalidades a nuestros programas.

Como es lógico el objetivo de las librerías es aumentar la eficiencia en la programación, pues al incorporar código escrito por otras personas se reduce la cantidad de líneas escritas por el usuario.

Todas las librerías usadas fueron descargadas e instaladas con comandos del estilo sudo pip-install nombre de la librería, a excepción de la llamada “pasar_ways”.

Dicha librería se trata únicamente de un nuevo programa escrito en lenguaje Python, que simplemente busca el archivo con extensión .WAYPOINTS dentro de la carpeta compartida entre la Raspberry y la estación de tierra, para copiarlo dentro de la carpeta desde la cual cargamos los Waypoints a la controladora.

```
import os
import sys

#pasa los waypoints de la carpeta compartida en formato wp cualquier .waypoints en la carpeta compartida
os.system("sudo find /home/pi/Desktop/CarpetaCompartidaPRUEBA -type f -name '*.WAYPOINTS' -exec cp -rf {} /Registro_de_vuelo/wp \;")
```

Ilustración 70 Código de "pasar_ways.py". Fuente: Elaboración propia.

En el código propio y referente al programa primero se da una orden al sistema mediante el comando “os.system” que hace referencia a opciones de “debugging” o depuración de errores.

Posteriormente se espera un tiempo mediante el comando `time.sleep`(tiempo de espera). Este retardo obedece a optimizar la secuencia de arranque del sistema en su conjunto. Sería imposible cargar la misión en la controladora si esta se encontrase apagada, por lo que es esencial retrasar el encendido de la Raspberry para adaptarlo al de la controladora.

La línea posterior da inicio al programa, pidiendo luego mediante otra orden al sistema que se busque, a través del comando “find” dentro de la ruta especificada los archivos con extensión WAYPOINTS. Para posteriormente ejecutar (exec) el comando copiar en la carpeta del directorio de vuelo, siguiendo la ruta especificada.

Es conveniente destacar que la llamada a `Mavproxy.py`, que se encuentra instalado en el directorio del usuario (es decir `/home/pi`, razón por la cual es innecesario especificar su ruta), crea un directorio de vuelo al usar el comando “Aircraft” llamado “Registro_de_vuelo”.

A este nuevo directorio se accede con la ruta `/Registro_de_vuelo`, debido a que se encuentra debajo del directorio raíz (designado por `/`) según el sistema de carpetas de Linux.

El complemento `–master` hace referencia a `/dev/ttyAMA0` (del inglés “device”, aparato), para establecer el tipo y la relación (determinando al maestro y al esclavo) de la conexión establecida.

AMA0 es el puerto mediante el cual se comunica la Raspberry con la controladora, habiendo sido sustituido el puerto `ttyS0` (el puerto por defecto) por razones de eficacia.

Para poder comunicarle a la controladora la carga de los datos de la misión es necesario que este tipo de datos esté disponible en el directorio de vuelo creado anteriormente, carpeta donde además se descargarán los datos correspondientes a los “logs” de las misiones de vuelo.

De este modo el programa `Mavproxy.py` mediante el comando `cmd` o ejecutar, realizara la “wp load” o carga de los Waypoints de la misión con los datos presentes en el directorio de vuelo.

Un archivo .WAYPOINTS es simplemente un tipo de archivo similar a .txt con una serie de encabezados y reglas:

El primer punto fundamental es saber que un Waypoints es un punto definido por una posición geográfica, coordenadas de latitud, longitud, y en la mayoría de los casos altura. Son principalmente usados en instrumentos de navegación GPS.

Una sucesión de Waypoints permite al piloto del dron establecer rutas de puntos de referencia para recorrer diferentes trayectorias.

Para asegurar una lectura correcta de este tipo de archivos debemos seguir el siguiente formato:

QGC WPL 110											
0	1	0	16	0	0	0	43.523037	-5.609175	40.000000	1	
1	0	10	22	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	20.000000
2	0	10	16	0.000000	0.000000	0.000000	0.000000	0.000000	43.523037	-5.609175	40.000000
3	0	10	16	0.000000	0.000000	0.000000	0.000000	0.000000	43.522982	-5.610008	40.000000
4	0	10	16	0.000000	0.000000	0.000000	0.000000	0.000000	43.522982	-5.610008	30.000000
5	0	10	181	0.000000	1.000000	0.000000	0.000000	0.000000	43.522982	-5.610008	30.000000
6	0	10	16	3.000000	0.000000	0.000000	0.000000	0.000000	43.523062	-5.609912	30.000000
7	0	10	16	3.000000	0.000000	0.000000	0.000000	0.000000	43.523157	-5.609818	30.000000
8	0	10	16	3.000000	0.000000	0.000000	0.000000	0.000000	43.523274	-5.609801	30.000000
9	0	10	16	3.000000	0.000000	0.000000	0.000000	0.000000	43.523277	-5.609802	30.000000
10	0	10	16	0.000000	0.000000	0.000000	0.000000	0.000000	43.523277	-5.609802	40.000000
11	0	10	16	0.000000	0.000000	0.000000	0.000000	0.000000	43.523037	-5.609175	40.000000
12	0	10	181	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
13	0	10	181	1.000000	1.000000	0.000000	0.000000	0.000000	43.523037	-5.609175	40.000000
14	0	10	20	0.000000	0.000000	0.000000	0.000000	0.000000	43.523037	-5.609175	40.000000

Ilustración 71 Ejemplo de archivo .Waypoints. Fuente: servidor interno LOCIS.

Que se corresponde con un patrón de este tipo:

QGC WPL <VERSION>	<INDICE>	<WP ACTUAL>	<MARCO COORD>	<COMANDO>	<PARAM1>	...	<PARAM5/LONGITUD>	<LATITUD>	<ALTITUD>	<AUTOCONTINUAR>
-------------------	----------	-------------	---------------	-----------	----------	-----	-------------------	-----------	-----------	-----------------

Ilustración 72 Formato archivo .Waypoints. Fuente: Elaboración propia.

Primero, siempre se debe empezar con un encabezado de la forma QGC WPL 110, sintaxis que representa Qgroundcontrol (plataforma de código abierto que proporciona configuración para vehículos controlados por PIXHAWCK PX4), el número posterior implica la versión del software.

El siguiente punto que empieza en cero, representa la coordenada HOME o de inicio. Se representa con el índice cero, para seguir con el WP actual (siguiente en la lista).

Las siguientes líneas, para cada uno de los Waypoints contienen información de latitud longitud, altura y tiempo de espera en el punto (delay).

8.4.- PRUEBA DE FUNCIONAMIENTO

Las bases de la prueba deben ser tales que al iniciarse la Raspberry se consigan los siguientes objetivos:

- Copiado de archivos .waypoint

- Carga de archivo .waypoint

Para ello, primero se copien los archivos .Waypoints a los que posteriormente accederá mavproxy.py, para realizar la prueba lo primero será borrar, en caso de existir el archivo que espera encontrar el programa en la ubicación deseada. Para ello:

```
pi@raspi_locis:/Registro_de_vuelo/wp $ ls -l
total 16
-rw-r--r-- 1 root root 143 abr  9 13:04 2archivowaypoints.txt
-rwxr-xr-x 1 root root 220 mar 26 12:41 archivowaypoints.txt
-rw-r--r-- 1 root root 327 abr  9 12:36 test_way.txt
-rw-r--r-- 1 root root 1172 abr 11 11:30 xxx.WAYPOINTS
pi@raspi_locis:/Registro_de_vuelo/wp $ sudo rm xxx.WAYPOINTS
pi@raspi_locis:/Registro_de_vuelo/wp $ ls -l
total 12
-rw-r--r-- 1 root root 143 abr  9 13:04 2archivowaypoints.txt
-rwxr-xr-x 1 root root 220 mar 26 12:41 archivowaypoints.txt
-rw-r--r-- 1 root root 327 abr  9 12:36 test_way.txt
pi@raspi_locis:/Registro_de_vuelo/wp $ █
```

Ilustración 73 Borrado del archivo xxx.WAYPOINTS. Fuente: Elaboración propia.

De este modo se evita que el archivo ya exista en la ubicación destino y altere la prueba.

Posteriormente debemos asegurarnos de que exista un archivo llamado “xxx.WAYPOINTS” en la carpeta compartida, para ello:

- cd /home/pi/Desktop/CarpetaCompartidaPRUEBA
- ls -l

El tercer paso es cargar en la controladora otro archivo .waypoint ó .txt para ver si es reemplazado en el arranque, para ello:

Se ejecuta un archivo Python que carga un fichero .txt con coordenadas waypoint a la controladora de vuelo.

```
pi@raspi_locis:~ $ python RPI_to_PIX.py
Inicio de programa
Connect /dev/ttyAMA0 source_system=255
no script Registro_de_vuelo/mavinit.scr
Loaded 3 waypoints from /Registro_de_vuelo/wp/archivowaypoints.txt
Log Directory: Registro_de_vuelo/logs/2018-04-11/flight4
Telemetry log: Registro_de_vuelo/logs/2018-04-11/flight4/flight.tlog
Waiting for heartbeat from /dev/ttyAMA0
nGot MAVLink msg: MISSION_ACK {target_system : 255, target_component : 0, type : 0}
Sent waypoint 0 : MISSION_ITEM {target_system : 0, target_component : 0, seq : 0, frame : 0, command : 16, current : 1, autocontinue : 0, param4 : 0.0, x : 43.523027, y : -5.612192, z : 35.982169}
Sent waypoint 1 : MISSION_ITEM {target_system : 0, target_component : 0, seq : 1, frame : 3, command : 16, current : 0, autocontinue : 0, param4 : 0.0, x : 43.523344, y : -5.610733, z : 40.0}
Sent waypoint 2 : MISSION_ITEM {target_system : 0, target_component : 0, seq : 2, frame : 3, command : 181, current : 0, autocontinue : 0, param4 : 0.0, x : 43.523144, y : -5.609867, z : 40.0}
Sent all 3 waypoints
Got MAVLink msg: MISSION_ACK {target_system : 255, target_component : 0, type : 0}
APM: Flight plan received
fence breach
)8\Inonline system 1
ALT_HOLD> Mode ALT_HOLD
z#M_X BWPNAV_RADIUS OIM_TC ttt_JSTICK_SPDFlight battery 90 percent
26GFND2 MAX_CMERSD<tHDCRC9_MINFIDRC16_MINGJ ;>|< ;|;(7\2\H
{tSx:(:)}h:.F:#u █
```

Ilustración 74 Envío de Waypoints por consola de comandos. Fuente: Elaboración propia.

A continuación, comprobamos la recepción de la nueva misión en la controladora usando el software “Mission Planner”

Mission Planner es un software soportado por Windows que actúa como planificador del vuelo, es capaz de presentar una GUI (interfaz gráfica de usuario) que permite interactuar de una forma cómoda con la controladora PIXHAWCK, modificando sus parámetros, cargando Waypoints, y otras muchas funcionalidades.

Para ello necesitamos primero conectar el software con la controladora de vuelo por medio de dos radios de telemetría y mediante la función “connect”.

Luego dentro de la pestaña “Flight Plan” pulsar en “Leer Wps”, de la forma que se muestra en la imagen.



Ilustración 75 Comprobación de la misión recibida en Mission Planner. Fuente: Elaboración propia.

El último paso de la prueba consiste en reiniciar la Raspberry, para ello usamos el código sudo reboot en la consola de comandos.

Posteriormente, y del mismo modo anterior, volvemos a revisar si se han cargado los puntos correctos:



Ilustración 76 Comprobación de la misión recibida en Mission Planner. Fuente: Elaboración propia.

Efectivamente la prueba se puede calificar como exitosa.

8.5.- IMPLEMENTACIÓN DE MECANISMO DE SEGURIDAD

Por seguridad se propone que se realice la carga de la misión únicamente si su nombre coincide con el día actual, esta medida espera poder corregir errores humanos como por ejemplo la subida de otra misión posterior, confusión entre operarios...

Aun así se recomienda comprobar la misión cargada actual usando el software Mission Planner antes de realizar cualquier vuelo.

Se seguirá un diagrama de bloques que sigue el siguiente esquema de decisión:

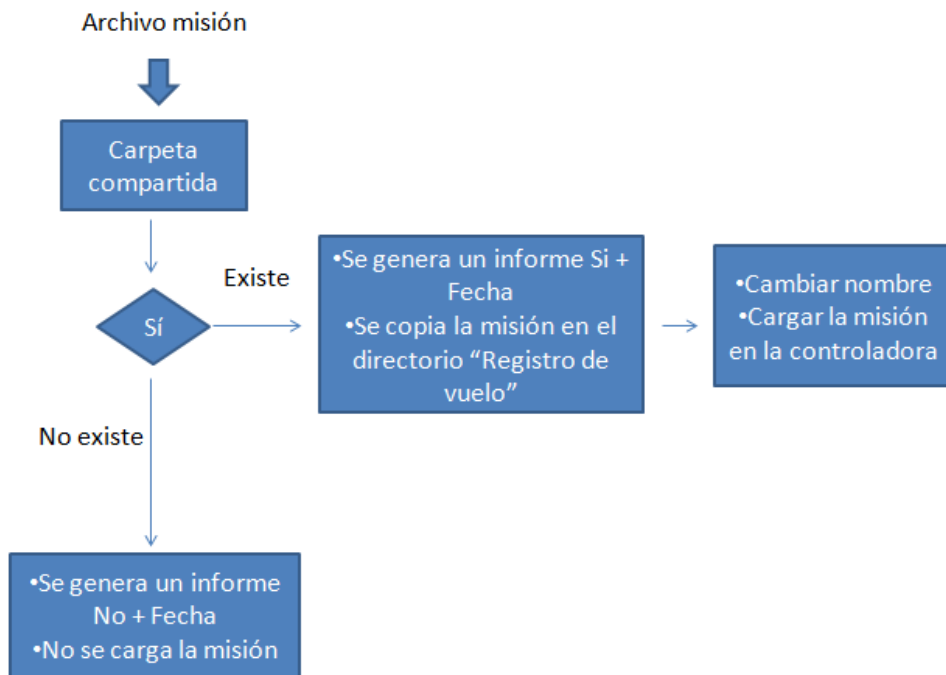


Ilustración 77 Esquema de decisión del mecanismo de seguridad. Fuente: Elaboración propia.

Para ello se actualizará el programa python llamado RPI_to_PIX_xxx.py usado para subir la misión.

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
import time
import os
#mostrar fecha en formato dia mes año
import sys

fecha=time.strftime('%d-%b-%y')#formato fecha dia-mes-año
k='/home/pi/CarpetaCompartidaPRUEBA/informes_mision/'+fecha+'.txt'
verificar=os.path.exists("/home/pi/Desktop/CarpetaCompartidaPRUEBA/%s.WAYPOINTS" %fecha)
print("comprobamos la existencia de /home/...%s.WAYPOINTS en la carpeta compartida" %fecha)
  
```

Ilustración 78 Actualización del programa Python parte 1. Fuente: Elaboración propia.

```

if verificar==True:
    print("Existe")
    os.system('sudo python /home/pi/pruebaSI.py')
    os.system("sudo find /home/pi/Desktop/CarpetaCompartidaPRUEBA/%s.WAYPOINTS -exec cp -rf {} /Registro_de_vuelo/wp \;" %fecha)
    old_file = os.path.join("/Registro_de_vuelo/wp", "%s.WAYPOINTS" %fecha)
    new_file = os.path.join("/Registro_de_vuelo/wp", "xxx.WAYPOINTS")
    os.system('sudo chown -R pi:pi /Registro_de_vuelo ')
    os.rename(old_file, new_file)
    os.system('sudo -s exec > /home/pi/RPI_to_PIX.out 2>&1; set -x')
    time.sleep(30)
  
```

Ilustración 79 Actualización del programa Python parte 2. Fuente: Elaboración propia.


```
print("Inicio de programa")
os.system('sudo -s mavproxy.py --master=/dev/ttyAMA0 --aircraft Registro_de_vuelo --cmd="wp load /Registro_de_vuelo/wp/xxx.WAYPOINTS;")

else:
os.system('sudo python /home/pi/pruebaNO.py')
```

Ilustración 80 Actualización del programa Python parte 3. Fuente: Elaboración propia.

Primero, y justo después de incluir las librerías oportunas, se llamará a la función “time.strftime” para averiguar la fecha actual en el formato día-mes-año. Para con ello, usando las siguientes líneas de comandos comprobar si la ruta a la citada misión existe:

- fecha=time.strftime()
- verificar=os.path.exists('%s' %fecha) Este comando devuelve un “TRUE” o cierto en caso de que la ruta especificada exista o un “FALSE” o falso en caso contrario.

Es conveniente destacar la importancia de los diferentes formatos, para poder nombrar a la misión de forma correcta. Ya que el nombre debe coincidir con la ruta absoluta para obtener un “True” en la verificación.

De este modo:

- %d: Representa el día del mes con un espacio precediendo a los dígitos simples, del 01 al 31.
- %b: Representa el mes del año, usando para ello las abreviaturas inglesas. A continuación se muestra una tabla con las nomenclaturas de los doce meses.

	Mes	Abreviatura	Días	Temporada
1	January [Enero]	Jan.	31	Invierno
2	February [Febrero]	Feb.	28/29	
3	March [Marzo]	Mar.	31	Primavera
4	April [Abril]	Apr.	30	
5	May [Mayo]	May	31	
6	June [Junio]	Jun.	30	Verano
7	July [Julio]	Jul.	31	
8	August [Agosto]	Aug.	31	
9	September [Septiembre]	Sep.	30	Otoño
10	October [Octubre]	Oct.	31	
11	November [Noviembre]	Nov.	30	
12	December [Diciembre]	Dec.	31	Invierno

Ilustración 81 Abreviaturas de los meses en inglés. Fuente: <https://www.aprender.org/meses-en-ingles/> Acceso: 24/4/2018.

- %y: Representación de dos dígitos del año del modo 18 para representar el año actual.

Posteriormente se entra en el “if” en caso de existir un fichero con mismo nombre al día actual o al “else” en caso contrario.

El bloque correspondiente al Sí lógico busca la misión en la carpeta compartida nombrada con el día actual (día-mes-año) y la copia en el directorio de /Registro_de_vuelo (donde el programa Mavproxy busca los archivos .Waypoints). Para evitar cambiar el código posterior se renombra la misión a subir como xxx.WAYPOINTS, para finalmente cargarla en la controladora.

Por su parte el bloque correspondiente al No lógico simplemente genera, en la carpeta compartida entre la Raspberry y la estación de tierra, un archivo .txt nombrado No+fecha y no intenta cargar ninguna misión en la controladora.

8.5.1.- Prueba de funcionamiento

Primero se plantea la opción de que tengamos archivos .WAYPOINTS nombrados de forma diferente al día actual, de este modo al iniciar la Raspberry no debería producirse la carga de la misión en la controladora y, además deberíamos ver un informe explicando la situación.

Para ello primero, volvemos a hacer el archivo python ejecutable con el comando:

- `sudo chmod +x <nombre_del_archivo>`

También transferimos la propiedad al usuario pi, con:

- `sudo chown pi: pi <nombre_del_archivo>`

```
-rwxr-xr-x 1 pi pi 1323 abr 25 09:22 RPI_to_PIX_xxx.py
```

Ilustración 82 Cambio de propietario y archivo ejecutable. Fuente: Elaboración propia.

Es necesario cambiar de propietario todo el directorio de forma recursiva, no llega con solo hacerlo al fichero sobre el que queremos hacer el renombrado.

Sera necesario además especificar donde se abrirá el fichero con el comando `f.open`, pues por defecto se crea en el directorio desde el cual se dio la orden. En este caso será el directorio `/etc` (que es donde se encuentra `rc.local`).

Es conveniente entonces o bien generar los informes en la carpeta compartida o bien copiarlos.

En este caso se optó por abrir el fichero en la propia carpeta compartida, por:

- Sencillez de uso, ya que así nos evitamos el copiado.
- Problemas con los privilegios “root” del directorio `/etc` (no era posible crear el archivo al iniciar la Raspberry)

Sin embargo, por desconocimiento, para las primeras pruebas se generó el archivo de texto en el directorio `/etc`, sucediéndose los siguientes inconvenientes:

No se conseguía la generación de informes `.txt` en inicio (a menos que previamente el archivo estuviera creado y solo se reescribiera)

Además, al intentar escribir en alguno de los archivos.txt generados usando el comando `f.write(“Texto a añadir”)` se producía el siguiente error:

```
Traceback (most recent call last):
  File "prueba.py", line 4, in <module>
    f.write="hello"
AttributeError: 'file' object attribute 'write' is read-only
```

Ilustración 83 Error al intentar escribir en el archivo creado. Fuente: Elaboración propia.

A falta de encontrar una solución mejor en ese momento se optó por generar un archivo.txt nombrado con el día en caso de encontrar la misión. Y con `No+día` en caso contrario.

Finalmente con este arreglo se consigue, al iniciar la Raspberry, un archivo de texto en blanco nombrado de la siguiente manera:

Nombre	Fecha de modifica...	Tipo	Tamaño
~syncthing~No25-Apr-18.txt.tmp	25/04/2018 12:40	Archivo TMP	0 KB
No25-Apr-18.txt	25/04/2018 12:40	Documento de tex...	0 KB

Ilustración 84 Archivo .txt en la carpeta compartida de la Raspberry que indica que no se encuentra misión concordante con el día actual. Fuente: Elaboración propia.

Estos errores, en pruebas subsiguientes, se subsanan creando directamente el archivo en la carpeta compartida, propiedad del usuario pi.

Usando para ello el siguiente subprograma, llamado desde el bloque del no lógico del programa principal (RPI_to_PIX_xxx.py).

```
import os
import time

k=time.strftime('%d-%b-%y')
h='No' +k+'.txt'
ru='/home/pi/Desktop/CarpetaCompartidaPRUEBA/informes_mision/'+h
file=open(ru, "w+")
file.write("El nombre de la mision no coincide con la fecha actual, NO se produce carga de la mision")
```

Ilustración 85 Prueba_NO.py en el directorio /home/pi llamado desde el no lógico. Fuente: Elaboración propia.

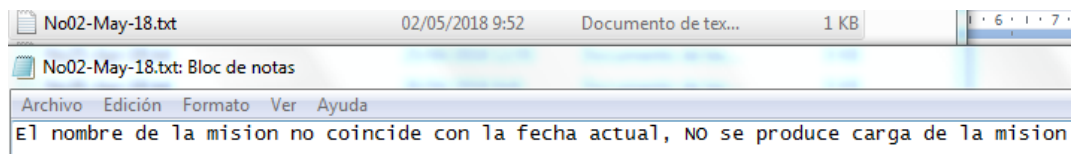


Ilustración 86 No carga de la misión. Fuente: Elaboración propia.

De este modo, esta parte de la prueba puede ser calificada como exitosa.

Ahora, para la siguiente parte, se generará un archivo .WAYPOINTS nombrado con el día actual para así probar el caso contrario.

Para calificar esta prueba como exitosa deberá generarse al encenderse la Raspberry pi un archivo.txt con nombre Si+Dia, y además producirse la carga de la misión.

Por lo que primero se carga una misión aleatoria a la controladora (observable a través de Mission Planner) y se sitúa en la carpeta compartida otra misión con formato fecha del día actual. WAYPOINTS.

Al reiniciar la Raspberry se observa el archivo de la misión cargado de forma correcta, como se aprecia en las siguientes imágenes:

Si25-Apr-18.txt	25/04/2018 13:48	Documento de tex...
Si26-Apr-18.txt	26/04/2018 10:15	Documento de tex...

Ilustración 87 Informe generado en la carpeta compartida de la Raspberry. Fuente: Elaboración propia.

	Comandos				Lat	Long	Alt	Borrar	Amb	Abajo
1	TAKEOFF	0	0	0	0	0	20	X	🏠	⬇️
2	WAYPOINT	0	0	0	43.5230368	-5.6091752	40	X	🏠	⬇️
3	WAYPOINT	0	0	0	43.5229824	-5.6100076	40	X	🏠	⬇️
4	WAYPOINT	0	0	0	43.5229824	-5.6100076	30	X	🏠	⬇️
5	DO_SET_RELAY	0	1	0	0	0	0	X	🏠	⬇️
6	WAYPOINT	3	0	0	43.5230624	-5.609912	30	X	🏠	⬇️
7	WAYPOINT	3	0	0	43.5231552	-5.609818	30	X	🏠	⬇️
8	WAYPOINT	3	0	0	43.5232736	-5.6098008	30	X	🏠	⬇️
9	WAYPOINT	3	0	0	43.5232768	-5.6098016	30	X	🏠	⬇️
10	WAYPOINT	0	0	0	43.5232768	-5.6098016	40	X	🏠	⬇️
11	WAYPOINT	0	0	0	43.5230368	-5.6091752	40	X	🏠	⬇️
12	DO_SET_RELAY	0	0	0	0	0	0	X	🏠	⬇️
13	DO_SET_RELAY	1	1	0	0	0	0	X	🏠	⬇️
14	RETURN_TO_L...	0	0	0	0	0	0	X	🏠	⬇️

Archivo	Edición	Formato	Ver	Ayuda
WGC WPL 110				
0	1	0	16	0
1	0	10	22	0
2	0	10	16	0
3	0	10	16	0
4	0	10	16	0
5	0	10	181	0
6	0	10	16	3
7	0	10	16	3
8	0	10	16	3
9	0	10	16	3
10	0	10	16	0
11	0	10	16	0
12	0	10	181	0
13	0	10	181	1
14	0	10	20	0

Ilustración 88 Misión vista en la planificadora de vuelo "Mission Planner". Fuente: Elaboración propia.

Ilustración 89 Archivo .WAYPOINTS con fecha actual. Fuente: Elaboración propia.

En este caso se repetían los mismos problemas (debido a la creación del archivo en un directorio "root") que en el caso del no, por lo que de la misma forma se actualiza la llamada al programa pruebaSI.py desde el sí lógico del programa principal.

```
import os
import time

k=time.strftime('%d-%b-%y')
h='Si'+k+'.txt'
ru='/home/pi/Desktop/CarpetaCompartidaPRUEBA/informes_mision/'+h
file=open(ru, "w+")
a=k+'.txt'
o='/home/pi/Desktop/CarpetaCompartidaPRUEBA/informes_mision/'+a
file.write("El archivo de mision coincide con la fecha actual se procede a cargar la mision en la ruta %s" %o)
```

Ilustración 90 PruebaSI.py llamado desde el sí lógico. Fuente: Elaboración propia.

Para la prueba de funcionamiento en inicio se hará:

- Carga de una misión diferente en la controladora de vuelo, para evitar que la misión estuviese precargada de pruebas anteriores.

- Generación de un archivo de misión con el nombre del día.
- Reinicio de la Raspberry.

```

permitted by applicable law.
Last login: Wed May  2 10:07:25 2018
pi@raspi_locis:~ $ sudo python RPI_to_PIX.py
Inicio de programa
Connect /dev/ttyAMA0 source_system=255
no script Registro_de_vuelo/mavinit.scr
Loaded 3 waypoints from /Registro_de_vuelo/wp/archivowaypoints.txt
Log Directory: Registro_de_vuelo/logs/2018-05-02/flight1
Telemetry log: Registro_de_vuelo/logs/2018-05-02/flight1/flight.tlog
link 1 down
link 1 OK
online system 1
ALT_HOLD> Mode ALT_HOLD
[Sent waypoint 0 : MISSION_ITEM {target_system : 1, target_component : 0, seq :
0, frame : 0, command : 16, current : 1, autocontinue : 1, param1 : 0.0, param2
: 0.0, param3 : 0.0, param4 : 0.0, x : 43.523027, y : -5.612192, z : 35.982169}
Sent waypoint 1 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 1
, frame : 3, command : 16, current : 0, autocontinue : 1, param1 : 0.0, param2 :
0.0, param3 : 0.0, param4 : 0.0, x : 43.523344, y : -5.610733, z : 40.0}
Sent waypoint 2 : MISSION_ITEM {target_system : 1, target_component : 0, seq : 2
, frame : 3, command : 181, current : 0, autocontinue : 1, param1 : 0.0, param2
: 1.0, param3 : 0.0, param4 : 0.0, x : 43.523144, y : -5.609867, z : 40.0}
Sent all 3 waypoints
APM: APM:Copter V3.5.5 (27229c83)
APM: PX4: 0384802e NuttX: 1bcae90b
APM: Frame: QUAD
APM: PX4v2 001A0029 3134510F 36313935
Got MAVLink msg: MISSION_ACK {target_system : 255, target_component : 0, type :
0}
APM: Flight plan received
L?R@E$RTL_LOIT_TIME _fence breach
BWAq=
>_RLL_PFlight battery 100 percent
pi@raspi_locis:~ $ sudo nano reboot
pi@raspi_locis:~ $ sudo rebooot █

```

Ilustración 91 Carga de una misión diferente antes de realizar la prueba y posterior reinicio de la Raspberry. Fuente: Elaboración propia.

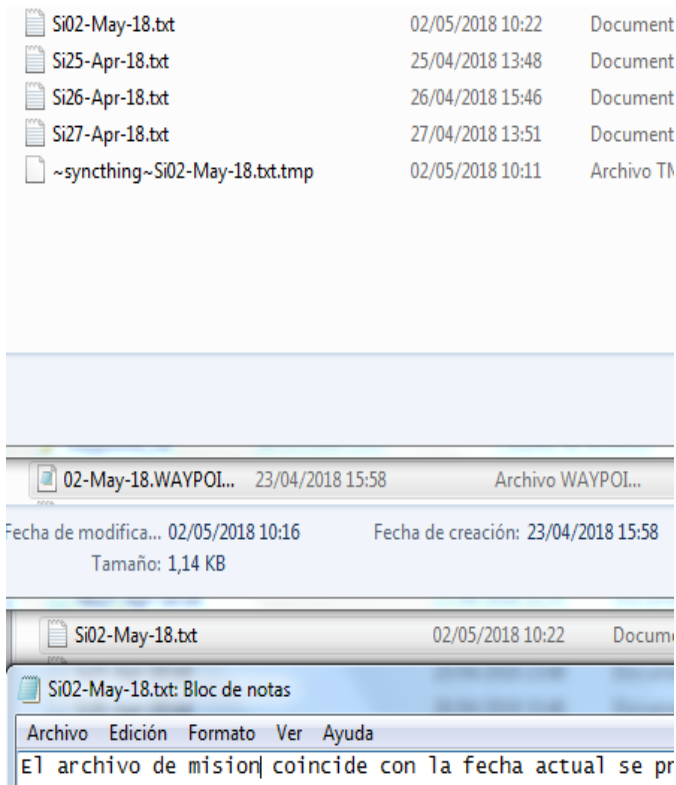


Ilustración 92 Informe generado al inicio de la Raspberry.
Fuente: Elaboración propia.

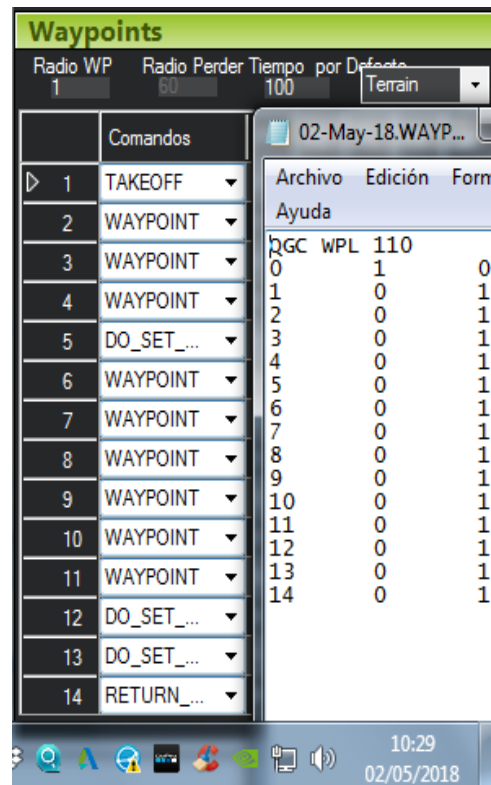


Ilustración 93 Archivo .txt de la carpeta compartida VS misión cargada en la controladora. Fuente: Elaboración propia.

De este modo podemos calificar también esta parte de la prueba como exitosa.

9. RTK

9.1.- PROBLEMÁTICA INICIAL

Debido a la falta de exactitud intrínseca de cualquier sistema GPS existía un margen de error correspondiente a un par de metros a la hora de posicionar el dron, llegando, en algunas pruebas de vuelo a acumularse “waypoint” tras “waypoint” hasta ser inaceptable, esto era en cierto modo preocupante cuando se volaba en espacios reducidos que pudiesen contar con obstáculos.

Para solucionar este problema se instala un dispositivo RTK.

9.2.- PRINCIPIO TEÓRICO

El término proviene del inglés “Real Time Kinematic” o navegación cinética satelital en tiempo real. Es una técnica muy novedosa que encuentra aplicación especialmente en cartografía y navegación marina.

Su funcionamiento se basa en el uso de medidas de fase de navegadores con señal GPS GLONNAS o Galileo, donde una única estación de referencia realiza correcciones en tiempo real, para que de este modo se pueda obtener una precisión sub-métrica.

Los receptores satelitales normales, comúnmente llamados GPS (de forma errónea pues este término designa la red de satélites estadounidense únicamente) comparan una señal aleatoria que se envía desde el satélite con una copia local generada por la misma señal.

La estación base retransmite la fase de la portadora que midió, y las unidades móviles comparan sus propias medidas de la fase con la recibida de la estación de referencia. De este modo se permite que las estaciones móviles calculen sus posiciones relativas absolutas relacionadas con las coordenadas de la estación base.

La señal del satélite tarda un tiempo predeterminado en llegar al receptor, de modo que ambas señales no se alinean de forma correcta. Generalmente la copia satelital tiene retraso en referencia a la local.

Se realiza un retraso progresivo de la señal local de modo que en algún momento ambas señales acaben alineándose. Y de este modo finalmente pueda calcularse la distancia al satélite.

Lo preciso de esta medida depende en gran parte de la capacidad electrónica del receptor para comparar las dos señales.

El caso RTK sigue los mismos preceptos pero usando el portador de satélite como su señal, no los mensajes de este. La mejora por lo tanto es alta. Tanto que permite alcanzar exactitud cent métrica en el mejor de los casos.

9.3.- PARTES DEL SISTEMA RTK

En el proyecto UAV-Inspection se usará el dispositivo RTK Reach de la casa Emlid, que consta de las siguientes partes:

Modulo Rover: Es una placa Intel Edison que se programa para funcionar como módulo en el aire (proceso en el cual se profundiza posteriormente), pudiendo equiparar su funcionamiento en solitario con el de un GPS común.

Modulo Base: Es otra placa Intel Edison que se programa (como se verá más adelante) para actuar como módulo en tierra.

Ambos módulos establecen un vínculo, en donde el módulo aire (conectado a la controladora de vuelo del dron) recibe correcciones en su posicionamiento de su gemelo situado en tierra.

Otra parte importante del sistema son las antenas, usadas por ambos módulos. Su uso radica en, de la forma que se introduce en el principio teórico, captar señales de satélite.

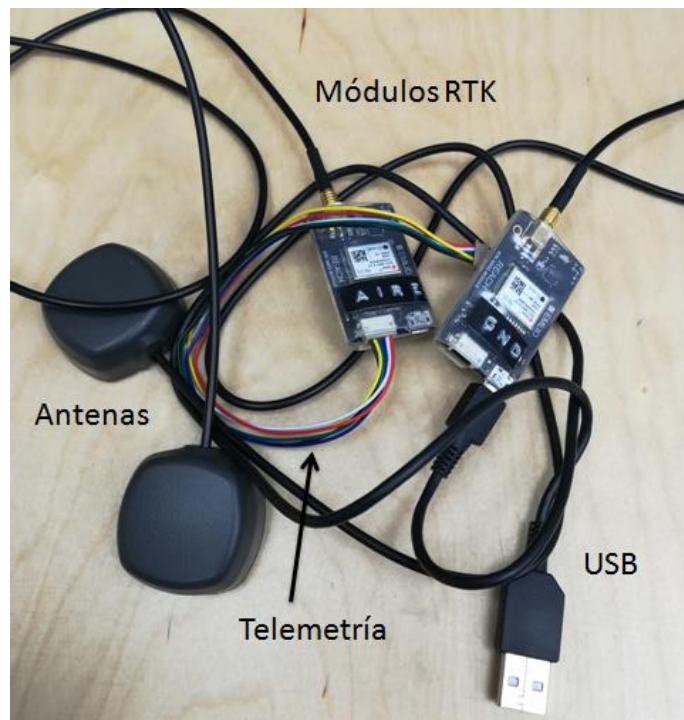


Ilustración 94 Partes del sistema RTK. Fuente: Elaboración propia.

9.4.- NOTACIÓN IMPORTANTE

A continuación se recoge notación importante referente al uso del dispositivo RTK.

Solution status

"-" means there is no information for the software to process. Either not enough time has passed or the antenna is not placed correctly.

Single means that rover has found a solution relying on it's own receiver and base corrections are not applied. Configuring positioning mode to Single will also result in this status. Precision in standalone mode is on meters-level.

Float means that base corrections are now taken into consideration and positioning is relative to base coordinates, but the integer ambiguity is not resolved. Precision in float mode is submeter-level.

Fix means that positioning is relative to the base and the integer ambiguity is resolved. Precision in standalone mode is centimeter-level.

Ilustración 95 Estado de la solución. Fuente: Documentación de Emlid,
<https://docs.emlid.com/reach/common/reachview/status/> Acceso: 01/06/2018.

9.5.- CONEXIONES FÍSICAS

Las conexiones del sistema RTK siguen una lógica sencilla, y fácilmente consultable en los documentos de la compañía (EMLID REACH).

En su página web presentan toda la documentación para una correcta implementación del dispositivo: <https://docs.emlid.com/reach/> que se podría resumir según el siguiente esquema:

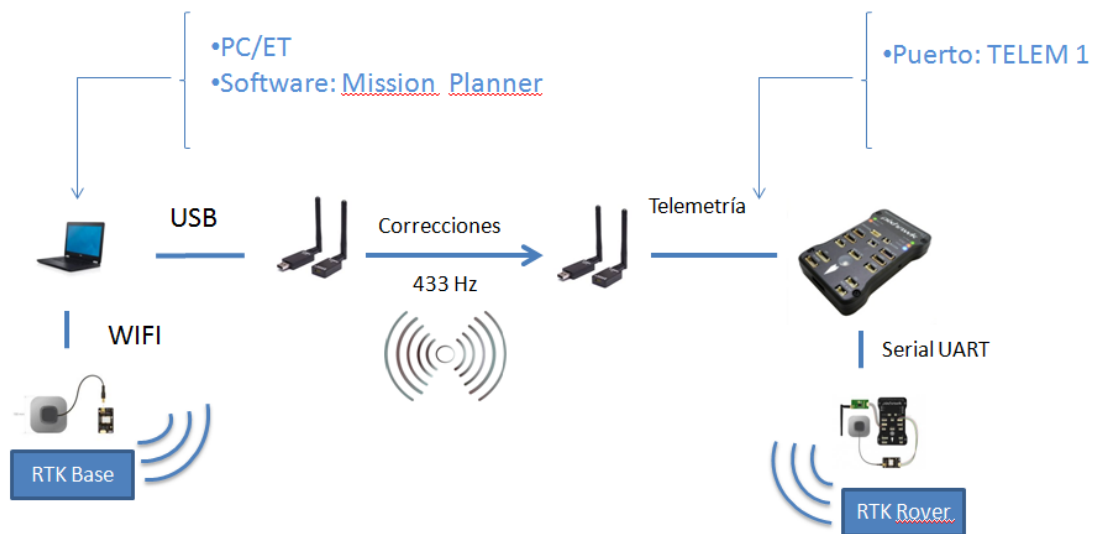


Ilustración 96 Esquema de conexionado físico RTK. Fuente: Elaboración propia.

9.6.- PROCEDIMIENTO DE APLICACIÓN E INTEGRACIÓN EN EL SISTEMA

A continuación se procede a detallar el montaje y acoplamiento del sistema RTK, para ello simplemente se seguirá el documento del fabricante que indica el procedimiento, para consultar la documentación en inglés directamente desde la página web de la compañía seguir el siguiente enlace (<https://docs.emlid.com/reach>).

9.6.1.- Encendido

El sistema se enciende conectando el cable micro-USB a USB proporcionado por el fabricante con una fuente de tensión de 5V, en este caso se usó un PC común de la oficina.

9.6.2.- Conexión con “ReachView”

“ReachView” es una interfaz web disponible para interactuar con cualquiera de los módulos del sistema RTK.

Para hacer uso de la aplicación primero se deberá alimentar alguno de los dos módulos del sistema. De esta manera, si alguno de los módulos Reach se alimenta por primera vez se creará un punto Wi-Fi Hotspot.

Tras esto:

- Abrir un buscador de conexiones Wi-Fi en por ejemplo un portátil, en este caso se usó el portátil de la oficina.
- Conectarse a una red llamada Reach: xx: xx
- La contraseña será emlidreach

9.6.3.- Ajustando la conexión Wi-Fi

Después de conectarse a la red ofrecida por el módulo Reach, abrir un navegador web.

- Escribir `http://reach.local` o `http://192.168.42.1` en la barra de direcciones para acceder al “ReachView”.

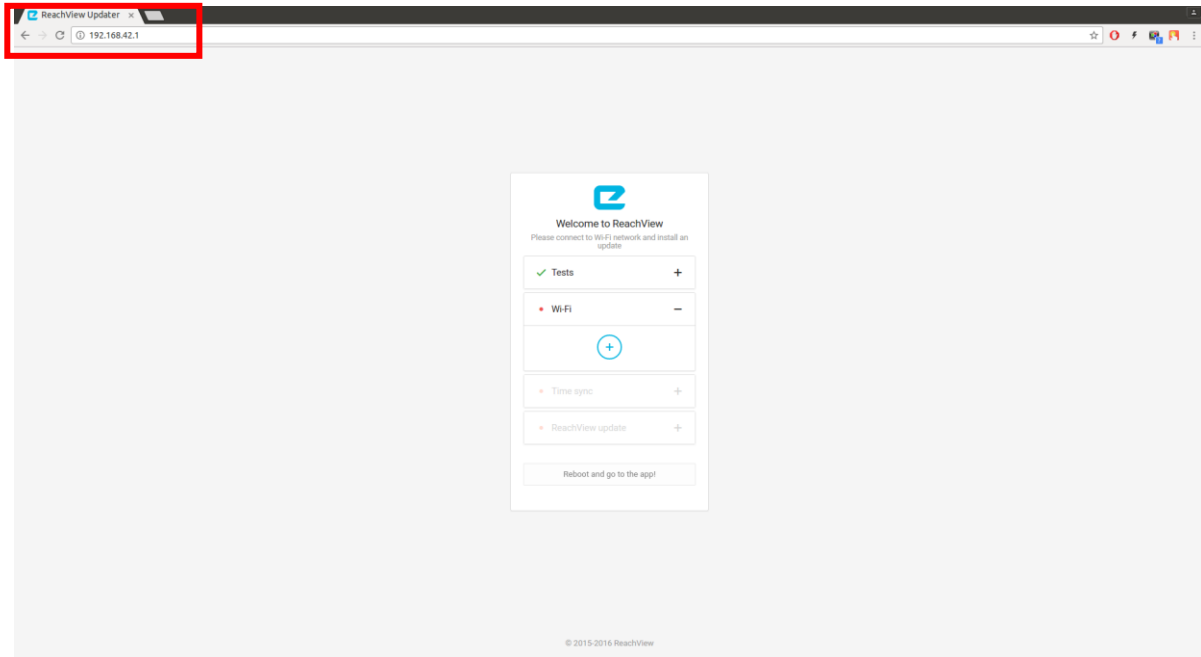


Ilustración 97 “ReachView” visto desde un navegador web. Fuente: documentos de Emlid.

- Pulsar el botón del + para conectar el módulo a una red Wi-Fi, se necesitará un nombre para la red, una contraseña y especificar el tipo de seguridad.
- Posteriormente, una vez hecho pulsar en conectar.
- Tras esto, el módulo Reach intentará conectarse a la red especificada.

9.6.4.- Acceso al dispositivo Reach RS dentro de una red Wi-Fi

Lo primero que se necesita hacer es identificar la dirección IP del dispositivo, para eso se debe usar un software de escaneo.

En este caso concreto se usó una aplicación para Android o IOS llamada Fing. La cual es capaz de detectar direcciones IP de los dispositivos conectados a la misma red Wi-Fi que el aparato que la use.

A continuación se muestra un ejemplo de uso de la citada aplicación, en donde se puede ver el módulo de base del dispositivo Reach bajo el nombre de Murata Manufacturing.

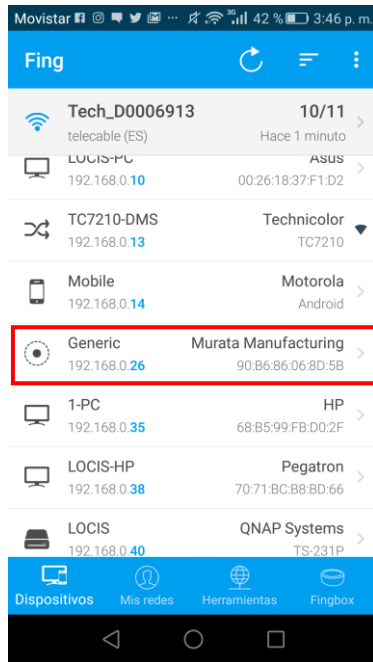


Ilustración 98 Dispositivos conectados a la red de la oficina de LOCIS. Fuente: Elaboración propia.

Una vez conocida la dirección IP del dispositivo en esa red Wi-Fi podemos acceder a él tecleándola en la barra de direcciones. Como se muestra en la imagen siguiente, en donde se ha cambiado el nombre del dispositivo a RTKtierra (módulo del sistema que se quedará en tierra, es decir la base):

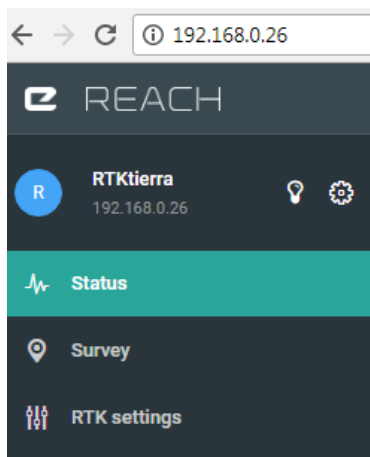


Ilustración 99 ReachView módulo base. Fuente: Elaboración propia.

9.6.5.- Ajustes de la estación base

Conectar uno de los módulos, el que ejercerá de base, para después configurarlo de la siguiente manera:

- Ir a “Base Mode” en el menú de la izquierda (se puede observar una parte del menú en la imagen anterior).
- Ajustarlo a protocolo de conexión TCP para el puerto 9000 (conexión a través de protocolo internet).

9.6.6.- Ajustes de la estación Rover

Conectar el otro módulo, que ejercerá de Rover (modulo aire), para después configurarlo de la siguiente manera:

- Ir a “correction input” en el menú de la izquierda.
- Elegir TCP como modo de corrección.
- Escoger cliente como rol.
- Escoger puerto por defecto 9000.
- Para “format” escoger RTCM3 (estándar de comunicaciones).
- “Apply” para guardar los cambios.

9.6.7.- Visionado de resultados

Para poder ver los datos y las correcciones ofrecidas por el módulo de base se debe ir a la pestaña “Status” del menú de la izquierda del módulo Rover.

9.6.8.- Resumen de interfaz ReachView

La interfaz de los módulos del sistema RTK, a la cual accedemos con la dirección IP de cada uno de ellos, consta de 9 pestañas.

De esas 9 las más importantes serán, la de “Status” o estado, que se muestra por pantalla. La de “Base Mode” o modo de la base, la de “Correction input” o corrección de entrada y la de Wi-Fi (que permite añadir nuevas redes).

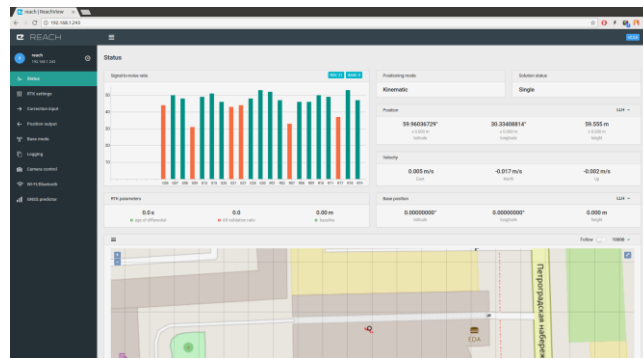


Ilustración 100 Pantalla ejemplo de la interfaz ReachView. Fuente: Documentación de Emlid.

9.6.9.- Añadir una nueva red Wi-Fi

Después de comprobar la necesidad de una red Wi-Fi para las pruebas de campo (remitirse al apartado de “pruebas de campo” para más información), se añade este punto como documentación a la hora de realizar nuevas conexiones.

Para ello será necesario establecer el modo “Hotspot” en un terminal móvil, en el caso concreto el Aquaris U plus del tutor en la empresa y seguir los siguientes pasos:

- Creación de la nueva red, proporcionándole un nombre y una contraseña.

- Conectar los módulos Reach a la nueva red, para ello pulsar en Wi-Fi dentro de ReachView
- Abrir un software, que del mismo modo que se realizó anteriormente nos permita conocer la dirección IP dentro de esta nueva red.

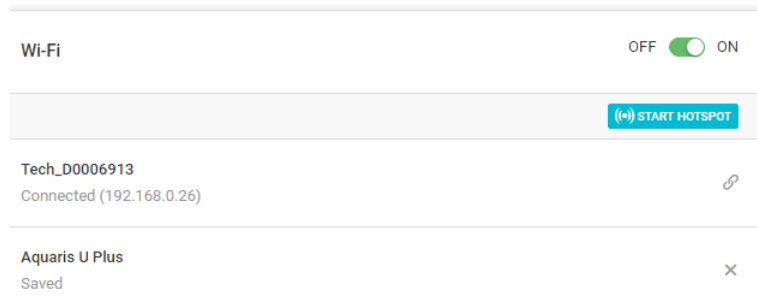


Ilustración 101 Creación de nueva conexión. Fuente: Elaboración propia.

En la imagen se puede ver la conexión existente con el Wi-Fi de la oficina, abajo se observa la conexión guardada con el terminal móvil.

9.7.- PRUEBAS DE CAMPO RTK

9.7.1.- Prueba de campo 1

9.7.1.1.- Resumen

Se realiza la primera prueba del dispositivo RTK.

9.7.1.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

9.7.1.3.- Fecha y lugar de realización

El día 15 de febrero se realizó una prueba de funcionamiento dentro de la oficina de LOCIS.

9.7.1.4.- Motivo de la prueba

Se pretende recrear el montaje y acoplamiento del sistema RTK al dron, de la misma forma que en la documentación del fabricante.

Para, de este modo disponer de un sistema que, como se explica en apartados anteriores, reduzca el error intrínseco de posicionamiento referido al GPS.

9.7.1.5.- Resultados de la prueba

Al realizar el proceso no se aprecia ningún tipo de corrección proveniente del módulo BASE al módulo ROVER, que es lo que se esperaba.

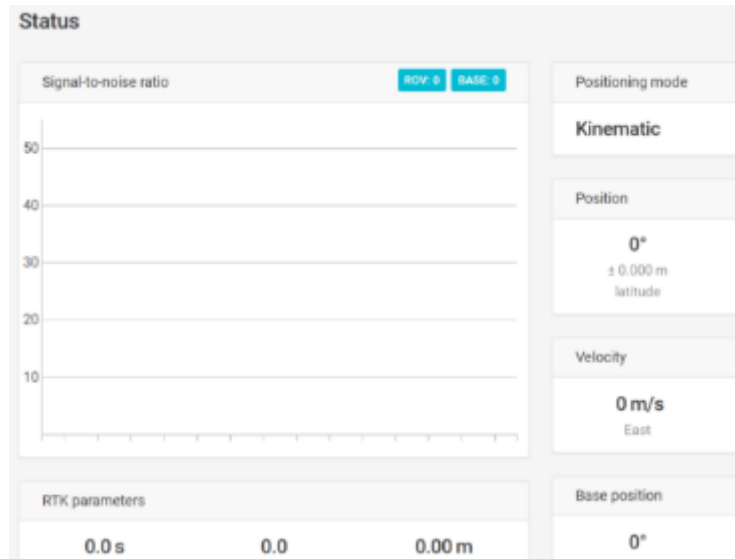


Ilustración 102 No correcciones (no recepción de comunicación satelital). Fuente: Elaboración propia.

9.7.1.6.- Conclusiones

Se decide repetir la prueba para asegurar que el procedimiento es el correcto, posteriormente, después de investigar se encuentra el motivo del error:

Como se apreció mirando el código de colores del GPS del dron y se confirmó usando un GPS de mano, es imposible encontrar satélites de posicionamiento dentro de la oficina, por lo que se decide realizar la siguiente prueba de conexionado del sistema RTK fuera de la oficina.

9.7.1.7.- Otros comentarios

9.7.2.- Prueba de campo 2

9.7.2.1.- Resumen

Se realiza la segunda prueba del dispositivo RTK.

9.7.2.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

9.7.2.3.- Fecha y lugar de realización

El día 16 de febrero se realizó una prueba de funcionamiento en el campo de pruebas de LOCIS.

9.7.2.4.- Motivo de la prueba

Dado que la primera prueba resulto ser no exitosa debido a no poder captar ningún satélite de posicionamiento dentro de la oficina se plantea la posibilidad de realizar esta segunda prueba en el exterior.

9.7.2.5.- Resultados de la prueba

Se traslada el siguiente equipo al campo de pruebas:

- Dron
- Ambos módulos del dispositivo RTK
- PC/Estación de tierra
- Radios de telemetría
- GPS de mano

Después de comprobar con el GPS de mano que se recibe señal de posicionamiento por satélite, se repite el procedimiento especificado anteriormente.

Todo el proceso ocurre según lo previsto hasta el último paso, en donde se realiza la conexión de la BASE con el software planificador de vuelo (“Mission Planner”).

En ese punto sucede un error del tipo:

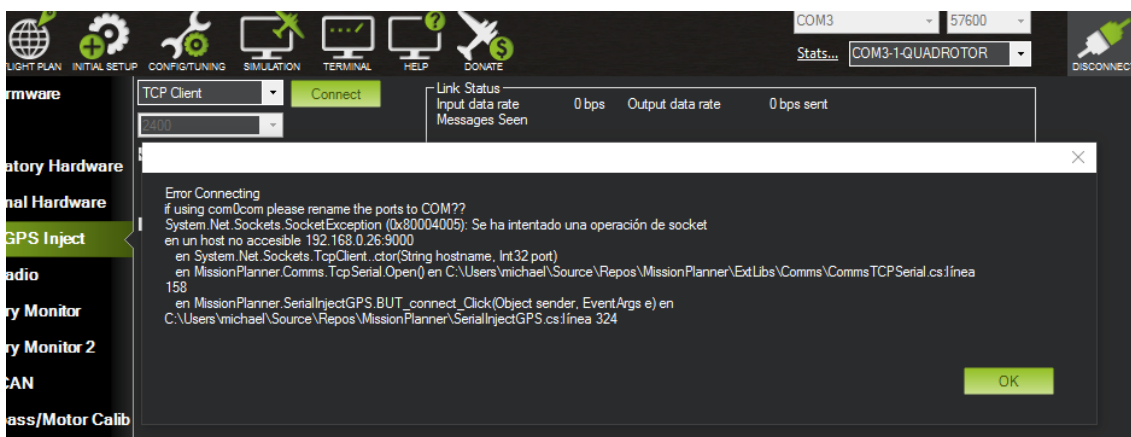


Ilustración 103 Error al intentar "GPS inject". Fuente: Servidor interno LOCIS.

Dicho error hace referencia a la no accesibilidad del socket (conexión entre dos dispositivos) del módulo BASE, representado por su dirección IP dentro de la red de la oficina de LOCIS (consultar el apéndice para más documentación).

Analizando este error se llega a la conclusión de que el problema en el campo de pruebas pasa por la imposibilidad del módulo BASE de detectar la red de la oficina LOCIS, y de por tanto conectarse con el software de planificación de vuelo.

Es decir es imposible crear una conexión TCP (especificada anteriormente) entre cliente y servidor (Rover y base) en ausencia de Wi-Fi.

9.7.2.6.- Conclusiones

A raíz de las dos pruebas anteriores es necesario realizar una tercera, para la cual debe buscarse un punto que comparta las siguientes características:

- Existencia de conexión Wi-Fi, los módulos RTK son accesibles mediante direcciones IP vinculadas con la red de la oficina de LOCIS.
- Captación de satélites de posicionamiento.

9.7.2.7.- Otros comentarios

9.7.3.- Prueba de campo 3

9.7.3.1.- Resumen

Se realiza la tercera prueba del dispositivo RTK.

9.7.3.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

9.7.3.3.- Fecha y lugar de realización

El día 18 de febrero se realizó una prueba de funcionamiento en la puerta exterior de la oficina LOCIS.

9.7.3.4.- Motivo de la prueba

Dados los resultados de las pruebas anteriores se hizo necesario buscar un lugar con las características anteriormente descritas.

Para ello se escoge el exterior de la oficina, al lado de la puerta exterior.

9.7.3.5.- Resultados de la prueba

Después de comprobar la captación de satélites con el GPS de mano y la posibilidad de conectarse al Wi-Fi de la oficina se procede a realizar otra vez el procedimiento.

Esta vez el resultado es semi-exitoso, se consigue el estado “Single” del módulo Rover (consultar el apartado referente a estado de módulos RTK dentro del apéndice para más información), pero no se consiguen correcciones del módulo BASE.

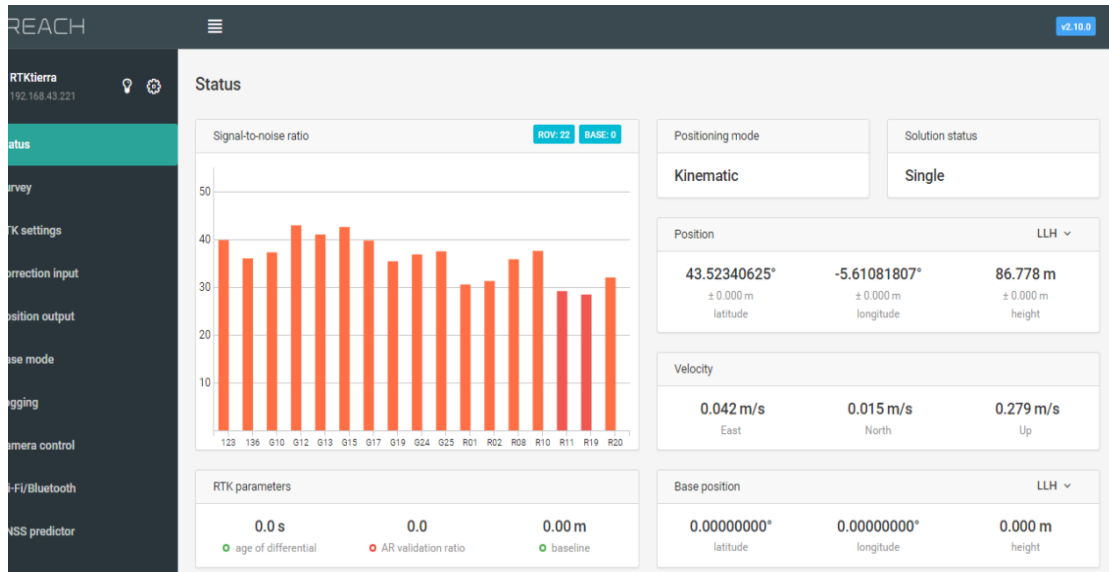


Ilustración 104 Módulo Rover del sistema RTK, en modo "Single". Fuente: Servidor interno LOCIS.

En la imagen podemos ver la posición referente a LOCIS, la velocidad, y la cantidad y el tipo de satélites que el módulo Rover es capaz de detectar.

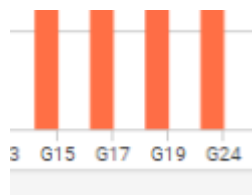


Ilustración 105 Vista de detalle al tipo de satélite. Fuente: Servidor interno LOCIS.

Asimismo también se aprecia que no se están recibiendo correcciones procedentes del módulo de BASE (se ve un cero en el conteo de satélites de BASE), que según la documentación de Emlid se representarían como barras grises al lado de las naranjas y rojas.

9.7.3.6.- Conclusiones

Tras leer la documentación del fabricante se llega a una posible causa del error, para la captación de satélites por parte de la base es importante la colocación correcta de la antena que acompaña a dicho módulo.

Asimismo, en la imagen siguiente se puede leer un mensaje de ayuda referente a la interfaz del módulo base que dice "Please connect antenna and make sure that it has unobstructed sky view" ó "conecte la antena y asegúrese que tiene una vista despejada del cielo".

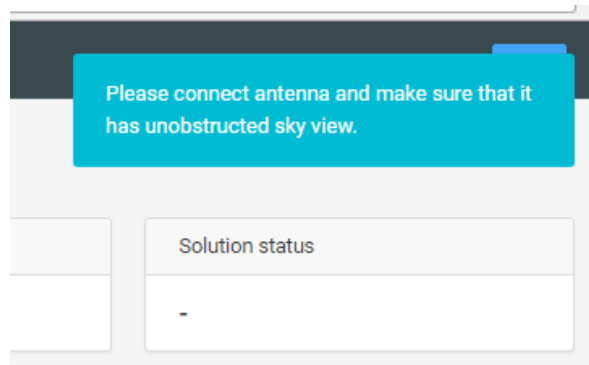


Ilustración 106 Mensaje de ReachView del módulo base. Fuente: Elaboración propia.

Por lo que se plantean pruebas sucesivas para poder llegar a al menos una solución float.

9.7.3.7.- Otros comentarios

9.7.4.- Prueba de campo 4

9.7.4.1.- Resumen

Se realiza la cuarta prueba del dispositivo RTK.

9.7.4.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

9.7.4.3.- Fecha y lugar de realización

El día 20 de febrero se realizó una prueba de funcionamiento en el campo de pruebas de LOCIS.

9.7.4.4.- Motivo de la prueba

Dados los resultados de las pruebas anteriores se hizo necesario buscar un lugar a cielo más despejado que permite una mejor colocación de la antena del módulo BASE.

Para ello se escoge una colina detrás de la oficina, dentro del campo de pruebas de LOCIS.

9.7.4.5.- Resultados de la prueba

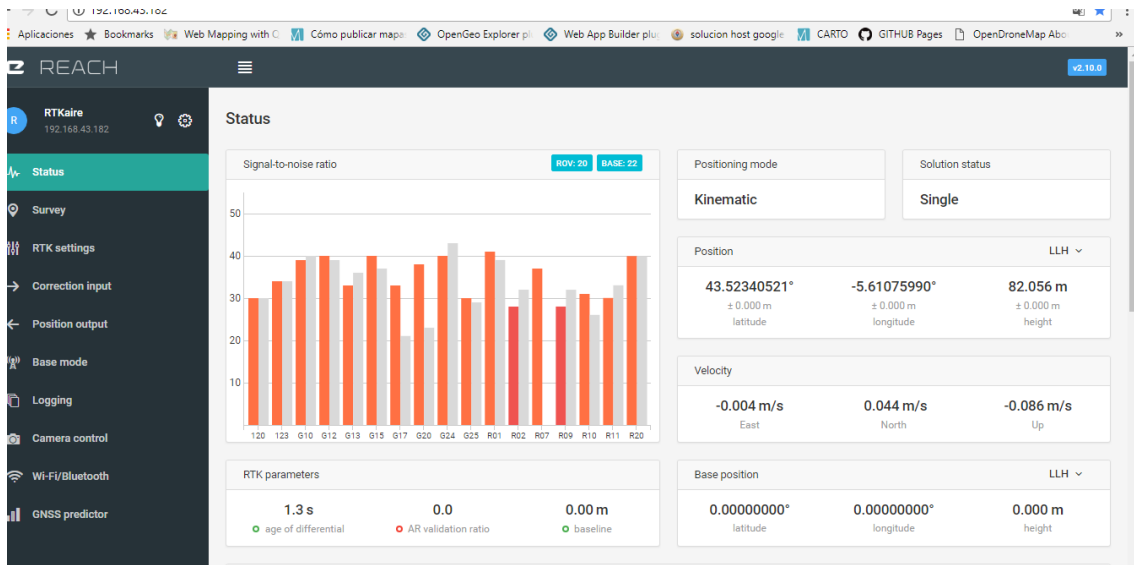


Ilustración 107 Sistema RTK con solución Single, con la BASE captando satélites. Fuente: Servidor interno LOCIS.

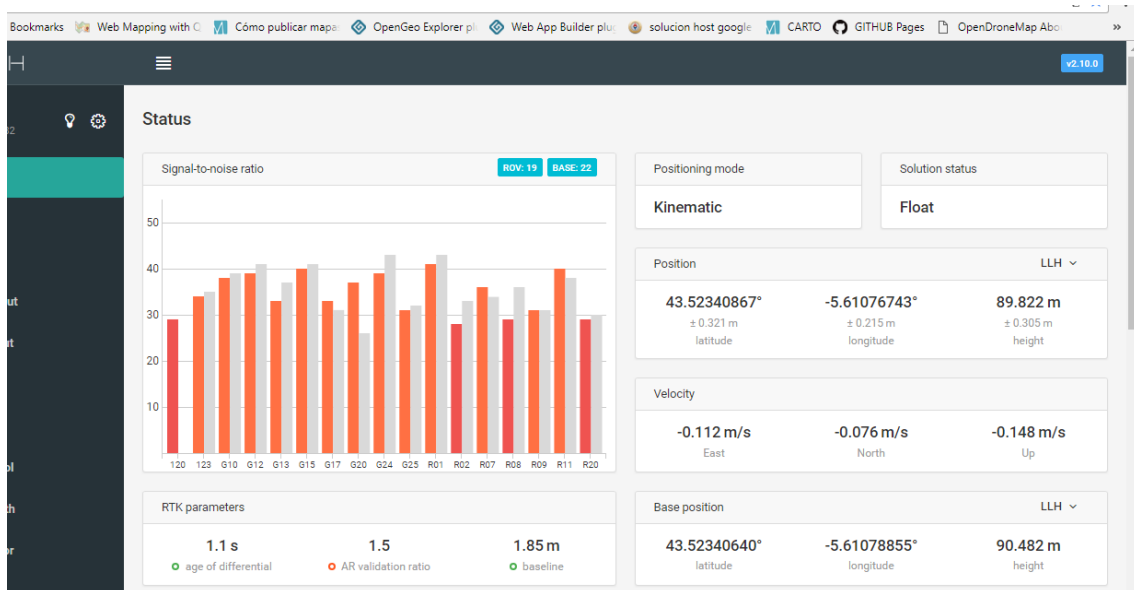


Ilustración 108 Sistema RTK con solución Float, recibiendo correcciones de la base. Fuente: Servidor interno LOCIS.

9.7.4.6.- Conclusiones

9.7.4.7.- Otros comentarios

Actualmente se pretende fabricar un aumento a la antena entregada por el fabricante con el producto.

9.8.- CONCLUSIONES DEL USO DEL DISPOSITIVO

Primero es necesario establecer una conexión mediante WIFI entre el módulo RTK que actúa como base y el PC que actúa como estación de tierra.

Se recomienda que la conexión WIFI provenga de un terminal móvil en modo Hot-spot (usando una tarifa de datos móvil para generar un punto de acceso WIFI). Este es debido a que en las misiones es más que probable que no exista conexión WIFI.

El módulo RTK es capaz de conectarse a puntos WIFI y recordarlos, por lo que una vez establecida dicha conexión será posible activar las correcciones.

Para recibir correcciones de la base es necesario contar con un planificador de vuelo con una funcionalidad de “INJECT GPS”, como por ejemplo Mission Planner.

Las correcciones se transmiten vía radio a la controladora de vuelo que a su vez se encuentra conectada al módulo Rover.

9.9.- PROCEDIMIENTO DE USO EN CAMPO

Después de las sucesivas pruebas realizadas se confecciona una manual para el sistema, del modo:

- Conexión terminal móvil en modo “Hotspot”, para ello se usa el Aquaris U plus de Guillermo (tutor del proyecto) con la contraseña locis010809. A esta red se conectarán los módulos RTK con las siguientes direcciones IP:
 - RTK modulo Rover o aire (el que vuela con el dron) 192.168.43.182
 - RTK modulo BASE (manda correcciones desde tierra) 192.168.43.221
- Encendido de módulos RTK, ambos deben estar en color verde y azul fijo (para más detalles consultar el siguiente enlace a la documentación del fabricante <https://docs.emlid.com/reach/led-status/>)
- Conexión del Rover al dron mediante el terminal serial 4/5
- Conexión de la base mediante Mission Planner, para ello usamos la funcionalidad GPS Inject seleccionando TCP client port 9000 y añadiendo la dirección del dispositivo BASE en la red correspondiente, la de la red Aquaris en este caso.
- Comprobar estado “fixed” o “float” dentro de la interfaz de los módulos proporcionada por el fabricante, y comprobar la llegada de correcciones vía Mission Planner.

10. Lidar

10.1.- INTRODUCCIÓN

Se estudia el uso del sistema Lidar (“light detection and ranging”) como instrumento de “sense and avoid” para el proyecto UAV Inspection.

Esta tecnología permite determinar la distancia desde un emisor a un objeto o superficie utilizando un haz láser pulsado. La distancia al objeto se determina midiendo el tiempo de retraso entre la emisión de pulso y su detección a través de la señal reflejada.

De este modo se pretende que el nuevo componente se integre con la navegación y sea capaz de detectar obstáculos ubicados debajo del dron, realizando de forma autónoma las órdenes pertinentes para evitar dicho obstáculo.

Para ello se especifican una serie de bloques de trabajo que se deberán completar de forma sucesiva hasta llegar a la integración con la controladora:

- Comprobación de la funcionalidad del dispositivo Lidar.
- Registro de datos de distancia en un fichero txt.
- Integración FPV (la integración FPV se realizará en un punto aparte debido a que integra información procedente del Lidar, altura, y de la controladora, latitud y longitud, con fotos de la cámara RGB)
- Integración en la navegación (PIXHAWCK). En función del tiempo que se disponga.

El dispositivo usado en el proyecto se corresponde con el modelo Lite-v3 de la marca Garmin, con las siguientes especificaciones técnicas:

- Tamaño: 20x48x40 mm
- Peso: 22g
- Resolución: 1cm
- Voltaje de funcionamiento: 4,75-5,5 V de CC; 6V máx.

Según la documentación consultada, el propio fabricante recomienda varios modos de conexión e implementación:

- Conexión mediante bus serie de datos I2C.
- Conexión mediante PWM.

Para lo cual se debe consultar la esquemática de puertos del Lidar, que se numeran de la manera siguiente:

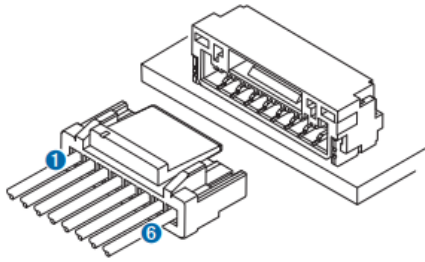


Ilustración 109 Esquema puertos de conexión de Lidar. Fuente: Staticgarmin.com Acceso: 3/5/2018.

Cumpliendo la siguiente notación:

1. 5 Vdc
2. Pull-up interno
3. Control del modo
4. SCL
5. SDA
6. Ground

El dispositivo Lidar puede conectarse, entre otros, a:

- Una “board” común ya sea Arduino o Raspberry.
- Una controladora de vuelo como por ejemplo PIXHWACK.

10.2.- CONEXIÓN CON LA RASPBERRY

Como se introdujo anteriormente el fabricante recomienda dos modos diferentes de cableado: I2C o PWM.

El estándar I2C adquiere su nombre del inglés “Inter-Integrated Circuit” o circuito inter integrado. Se corresponde con un bus serie de datos utilizado principalmente para la comunicación entre diferentes partes de un circuito, como por ejemplo entre un controlador y circuitos periféricos integrados.

El otro método hace referencia a las siglas inglesas “pulse-width modulation” (PWM) o modulación por ancho de pulsos. En este método se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

10.2.1.- Conexión hardware: Standard I2C

En este apartado se especifica el cableado necesario para comunicar la Raspberry pi con el dispositivo Lidar Lite, usando para ello el bus de datos I2C.

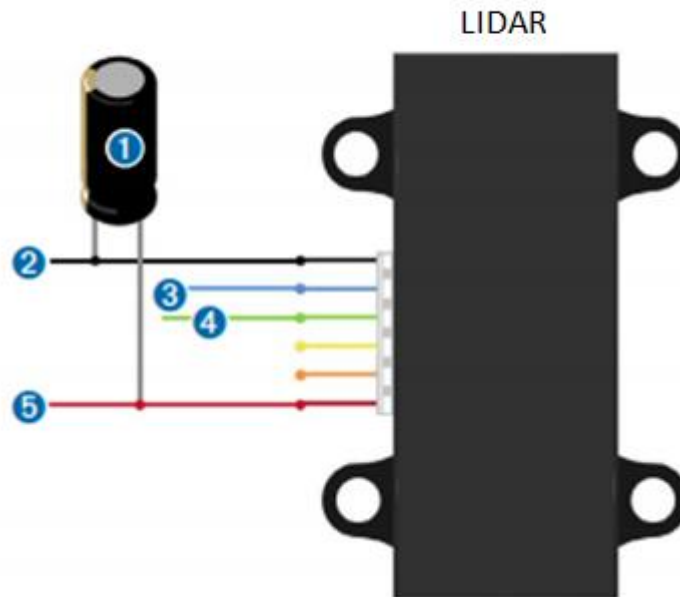


Ilustración 110 Diagrama de conexiones I2C. Fuente: <https://mobiustripblog.wordpress.com/2016/12/26/first-blog-post/> Acceso: 2/5/2018.

En donde cada número adquiere la consiguiente notación:

1. Condensador de 680 microfaradios. Nota: Los condensadores tienen polaridad.
2. Ground
3. Conexión I2C SDA “Serial Data” o línea de datos.
4. Conexión I2C SCL “Serial Clock” o línea de reloj.
5. Voltaje + (5Vdc)

De este modo el Lidar se alimenta desde uno de los pines GPIO de +5V de la Raspberry por lo que el condensador debe ubicarse entre la alimentación y uno de los terminales Ground de la Raspberry.

También deben cablearse las conexiones de la línea de datos (3) y línea de reloj (4) con sus correspondientes en la Raspberry, del modo:

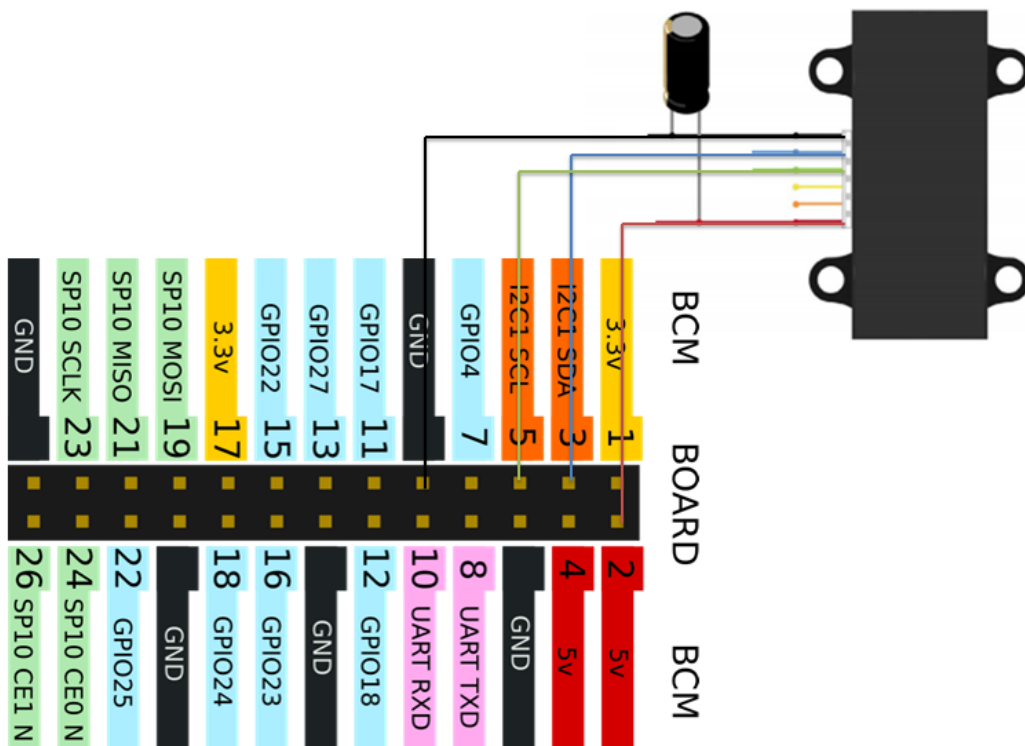


Ilustración 111 Esquema de conexiones Lidar- Raspberry. Fuente: Elaboración propia.

Nota: Es muy importante que el conector largo del condensador se derive al Vdc mientras que el conector corto al “Ground”.

10.2.2.- Software: Standard I2C

Primero se debe habilitar la conexión I2C en la Raspberry, para lo cual seguimos los siguientes pasos:

- Por consola de comandos tecleamos sudo raspi-config
- Escogemos la opción: 5 “Interfacing Options” y pulsamos “Enter”
- Escogemos la opción: P5 I2C y pulsamos “Enter”
- Pulsar sobre “Sí” para habilitar la interfaz I2C
- Pulsar “Ok” y “Finish”



Ilustración 112 Resultado de habilitar en la Raspberry I2C. Fuente: Elaboración propia.

Una vez completados todos estos pasos debemos escribir por consola:

- `ls /dev/*i2c*`, que debe retornar algo como `/dev/i2c-1`

```
pi@raspi_locis:~ $ ls /dev/*i2c*
/dev/i2c-1
```

Ilustración 113 Output correspondiente como interfaz habilitada. Fuente: Elaboración propia.

Esta comprobación nos indica si hemos habilitado de forma correcta la interfaz, posteriormente necesitamos una serie de herramientas para interactuar con el bus I2C, para ello escribiremos por consola los siguientes comandos:

- `sudo apt-get install -y i2c-tools`
- `sudo apt-get install libi2c-dev`
- `i2cdetect`: esta función debería retornar una dirección haciendo referencia al periférico (muestra todos los “slaves” en el bus especificado)

```
pi@raspi_locis:~ $ i2cdetect -y 1
   0 1 2 3 4 5 6 7 8 9 a b c
00:  -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- --
50:  -- -- -- -- -- 56 -- -- --
60:  -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- --
pi@raspi_locis:~ $
```

Ilustración 114 Comando `i2cdetect` haciendo referencia al periférico (Lidar) Fuente: Elaboración propia.

```
pi@raspi_locis:~ $ sudo apt-get install -y i2c-tools
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
i2c-tools ya está en su versión más reciente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 5
pi@raspi_locis:~ $ sudo apt-get install i2c-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
E: No se ha podido localizar el paquete i2c-dev
pi@raspi_locis:~ $ sudo apt-get install libi2c-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
libi2c-dev ya está en su versión más reciente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 5
pi@raspi_locis:~ $
```

Ilustración 115 Actualizaciones correspondientes. Fuente: Elaboración propia.

10.2.3.- Conexión hardware: PWM

En este apartado se especifica el cableado necesario para comunicar la Raspberry pi con el dispositivo Lidar Lite, usando para ello la modulación por ancho de pulsos.

A continuación se muestra por imagen el cableado necesario para establecer la comunicación:

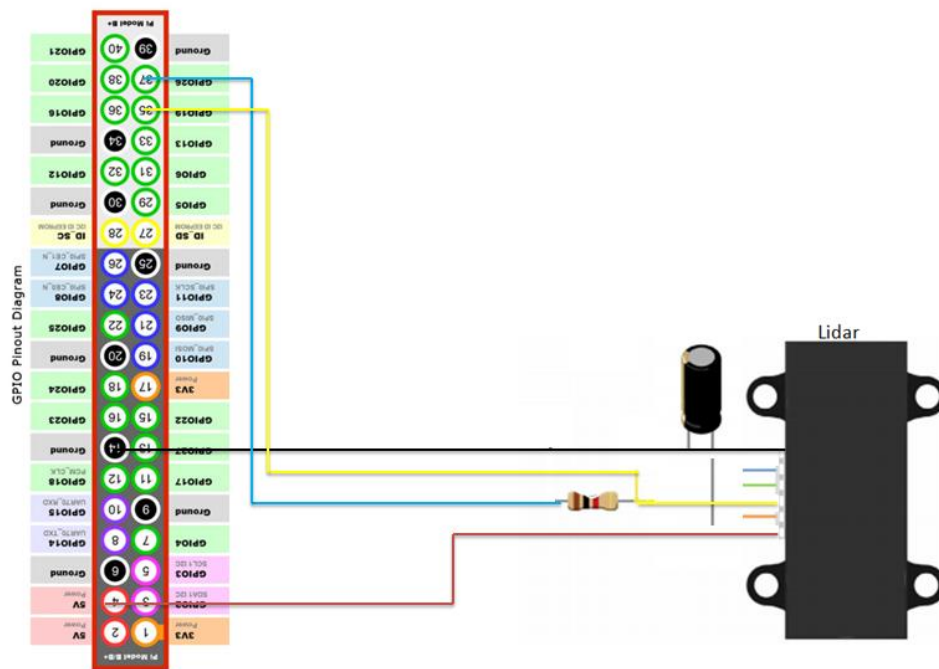


Ilustración 116 Esquema teórico de conexiones PWM Lidar y Raspberry pi. Fuente: Elaboración propia.

En donde se ven las siguientes conexiones:

- Terminal Ground del dispositivo Lidar con algunos de los terminales Ground de la Raspberry (en este caso se ilustra uno de ellos). Cableado color negro.
- Terminal de 5V del dispositivo Lidar con algunos de los terminales de 5V de la Raspberry (en este caso se ilustra uno de ellos). Cableado color rojo.
- Terminal de control de modo de Lidar Lite (cableado amarillo):
 - A input digital GPIO 19 en modo BCM, pin físico 37.
 - A resistencia de 1 Kohm
- Resistencia de 1 Kohm al output digital GPIO 26 en modo BCM, pin físico 36 (cableado azul).
- Se recomienda el uso de un condensador entre el terminal de 5V y el de Ground para evitar los picos en las corrientes de entrada cuando el dispositivo está inhabilitado.

10.2.4.- Software: PWM

Simplemente se necesita implementar un script para ejecutar desde la Raspberry, como por ejemplo el siguiente:

```
GNU nano 2.2.6          Fichero: PWM_lidar_rasp.py
import RPi.GPIO as gpio
import time
import datetime

#hay mas pines con pwm para probar
pwm_input=19
pwm_output=26

gpio.setmode(gpio.BCM)
gpio.setwarnings(False)

gpio.setup(pwm_output,gpio.OUT)#actua como trigger
gpio.setup(pwm_input,gpio.IN)#actua como monitor

gpio.output(pwm_output,gpio.LOW)#el trigger como LOW pa

def contar_microsec():
    millis = int(round(time.time()*1000000))
    return millis
while True:

    #contar cuanto tiempo el input es HIGH
    while not gpio.input(pwm_input):
        continue

    start= contar_microsec()
    while gpio.input(pwm_input):
        continue
    end=contar_microsec()
    pulseWidth= (end-start)
    if(pulseWidth != 0):
        pulseWidth= pulseWidth/10
        print pulseWidth
```

Ilustración 117 Script para toma de datos por Lidar. Fuente: Elaboración propia

Que realiza los siguientes comandos:

- Importación de las librerías necesarias.
- Define las variables y da estados de input u output:

En este caso existe un pin que se comporta como “trigger” o ejecutor (el output) y otro como monitor (el input).

- El tiempo que se cuenta en “pulseWidth” o ancho de pulso es el tiempo durante el cual el input es HIGH, que es el mismo tiempo que el haz láser tarda desde su salida desde el Lidar hasta su recepción como eco al chocar con un objeto.
- La función contar_microsec retorna el tiempo actual en microsegundos, por lo que se calculará la diferencia entre el “end” y el “start” del haz (teniendo así el tiempo que tardo la señal en ir desde el Lidar al objeto ida y vuelta). Al dividir por 10 este

tiempo se consigue la distancia en cm debido a que 10 microsegundos equivalen a un cm a la velocidad del haz láser.

10.3.- PRUEBAS DE FUNCIONAMIENTO

10.3.1.- Primera prueba (Toma de medidas con placa Arduino Uno y PWM)

Se pretende realizar la conexión del Lidar con una “board”. En este caso primero se hará, por sencillez, con una placa Arduino Uno.

La conexión hardware se realiza del mismo modo que para la Raspberry, teniendo en cuenta la reubicación de los puertos en cada una de las placas. A continuación se muestra un esquema del conexionado para la placa Arduino.

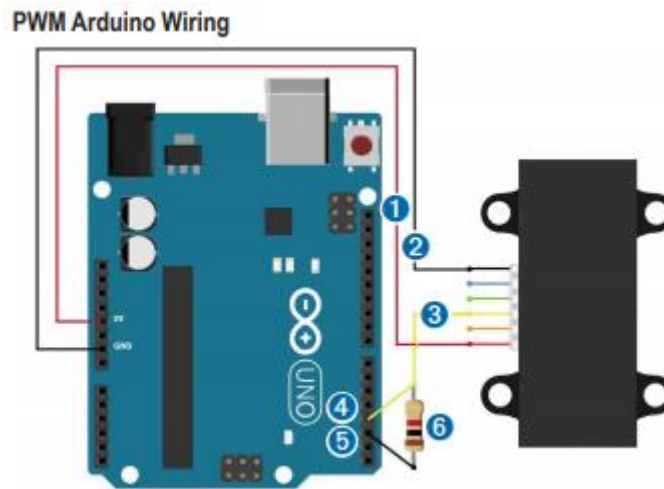


Ilustración 118 Esquema teórico de las conexiones para la placa Arduino Uno. Fuente: http://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf

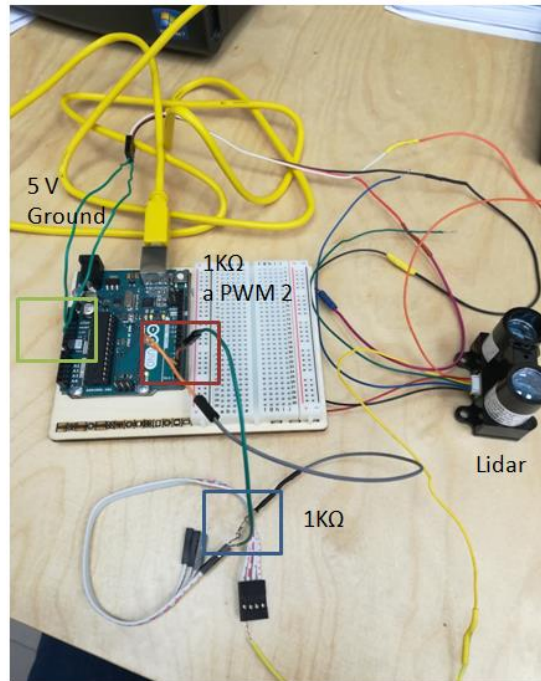


Ilustración 119 Montaje del dispositivo Lidar en la placa Arduino. Fuente: Elaboración propia.

Usando el programa “GetDistancePWM.ino” descargado del repositorio web GitHub (https://github.com/garmin/LIDARLite_v3_Arduino_Library/blob/master/examples/GetDistancePwm/GetDistancePwm.ino) se muestran las alturas en cm correspondientes a dos escenarios:

- Primero con el dispositivo Lidar encima de la mesa y sin ningún obstáculo, de forma que medirá la altura hasta el techo.
- Segundo la altura desde la mesa hasta la mano, ubicada está a cierta distancia en altura pero en el eje de medida del láser.

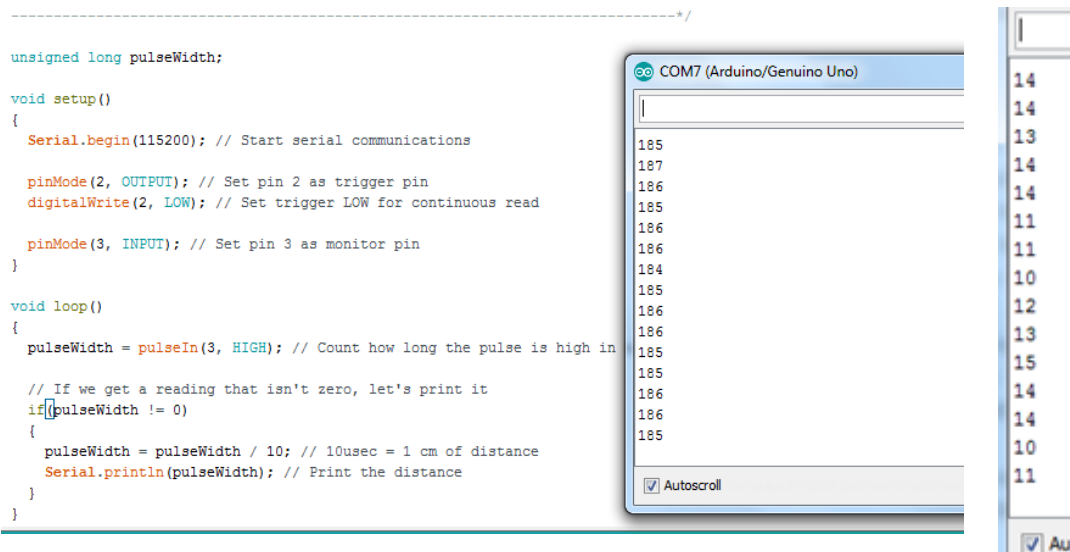


Ilustración 120 Medición de la altura al techo en cm usando el Lidar vs medición poniendo la mano.
Fuente: Elaboración propia. .

Nota: Para poder ver los resultados de las medidas por pantalla es necesario activar la opción de “monitor serie”.

Primero se empieza la comunicación entre el Arduino y el ordenador que actúa como consola a 115200 baudios.

Para luego:

- Establecer con la función PinMode como actuará cada uno de los pines.
- Dar un valor con digital write al output como LOW
- Entrar en el bucle

Dentro del bucle se computa el tiempo que el pin Input es alto, o lo que es lo mismo el tiempo desde el que se manda el láser hasta la recepción de su eco al chocar con un objeto.

A razón de haber obtenido mediciones en cada uno de los dos escenarios, se considera la prueba como exitosa, quedando a realizar comprobaciones de la exactitud en los resultados proporcionados por el dispositivo Lidar.

10.3.2.- Segunda prueba (Toma de medidas con placa Raspberry pi3 y I2C)

El montaje de la conexión (siguiendo el esquema teórico) se muestra en la siguiente imagen, la placa Arduino simplemente se usa para facilitar la conexión de los terminales 5 V y Ground de cada uno de los dispositivos con el condensador intermedio.

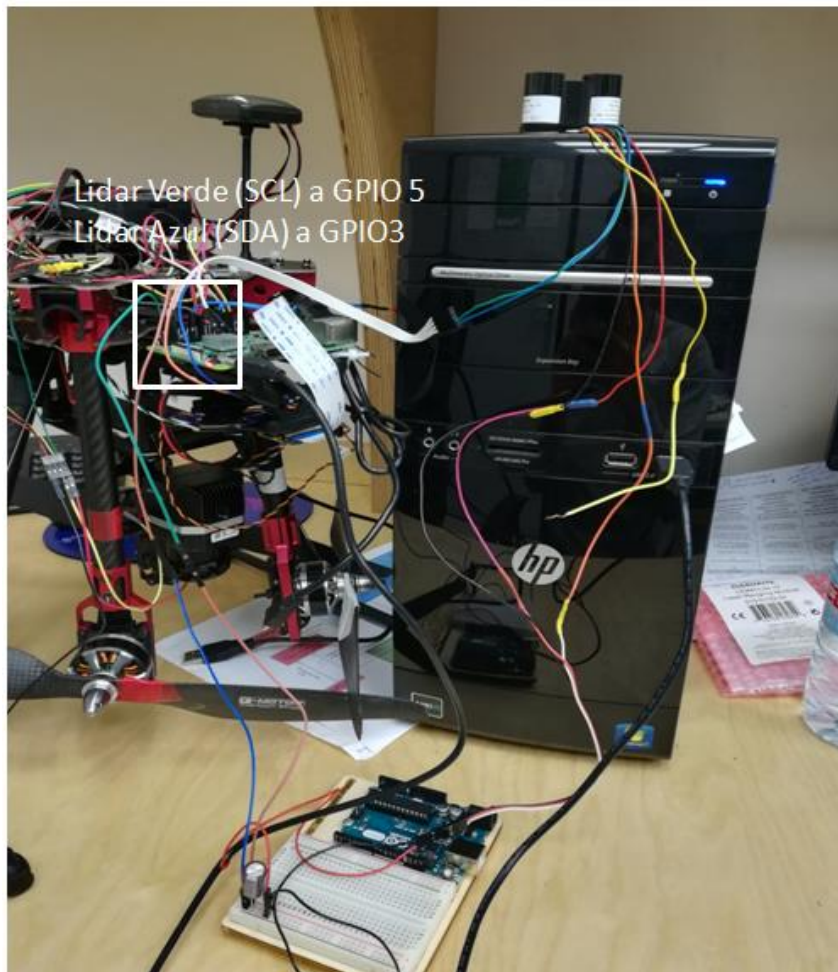


Ilustración 121 Conexión I2C a Raspberry pi. Fuente: Elaboración propia.

Para la realización de la prueba se descarga el programa `lidar_lite.py` del repositorio GitHub (https://github.com/Sanderi44/Lidar-Lite/blob/master/python/lidar_lite.py).

El programa anterior consta únicamente de una serie de definiciones escritas en lenguaje Python, las cuales han de ser “llamadas” cada vez que se quiera hacer uso de ellas.

Para ello se implementa otro programa llamado `lidar_I2C.py` (ubicado en la ruta de la carpeta compartida) cuyo código se resume en:

```
#!/usr/bin/env python
# -*- coding: cp1252 -*-
from lidar_lite import Lidar_Lite
import os
import sys
import time
import smbus

lidar = Lidar_Lite()

connected = lidar.connect(1)
ru='/home/pi/Desktop/CarpetaCompartidaPRUEBA/Lidar.txt'
f=open(ru,"w+")

if connected < -1:
    print("Not Connected")
else:
    print("Connected")
for i in range(100):
    distance = lidar.getDistance()
    f.write("La distancia es %s\n" %distance)
    print distance
```

Bus de datos de modelo B Raspberry

Ilustración 122 Código de lidar_I2C.py Fuente: Elaboración propia.

Al probar el programa anterior se produce un error del tipo:

```
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python lidar_I2C.py
Connected
Traceback (most recent call last):
  File "lidar_I2C.py", line 27, in <module>
    distance = lidar.getDistance()
  File "/home/pi/Desktop/CarpetaCompartidaPRUEBA/lidar_lite.py", line 33, in getDistance
    self.writeAndWait(self.distWriteReg, self.distWriteVal)
  File "/home/pi/Desktop/CarpetaCompartidaPRUEBA/lidar_lite.py", line 24, in writeAndWait
    self.bus.write_byte_data(self.address, register, value);
IOError: [Errno 121] Remote I/O error
```

Ilustración 123 Error al ejecutar el programa. Fuente: Elaboración propia.

Como se puede leer por pantalla se aprecia un fallo correspondiente al input/output del dispositivo Lidar, la razón de este fallo es haber confundido los cables correspondientes a SDA y SCL.

Una vez solucionado este primer inconveniente se vuelve a ejecutar el programa, esta vez imprimiéndose por pantalla los siguientes resultados:

```
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python lidar_I2C.py
Connected
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Ilustración 124 Ejecución del programa. Fuente: Elaboración propia.

Obviamente los resultados obtenidos no se consideran satisfactorios, dado que se esperaba tener unas medidas similares a las que se obtuvieron en la primera prueba usando la placa Arduino.

Después de investigar se llega a la causa del error. La nueva versión del Kernel (núcleo) de la Raspberry causa que el controlador del bus i2c intente la comunicación por medio de “repeated starts” (para más información consultar apéndices), opción que no es soportada por el dispositivo Lidar.

```
pi@raspi_locis:/dev $ uname -r
4.9.35-v7+
```

Ilustración 125 Versión actual del Kernel. Fuente: Elaboración propia.

Por lo tanto se barajan una serie de opciones para solventar este problema:

- Usar una versión más antigua de Kernel que permita desactivar a voluntad la opción de “repeated starts”.
- Usar un parche o modificación del Kernel actual que arregle esta opción.

Se prueba primero a cambiar únicamente el controlador del bus i2c a la versión anterior requerida, para ello:

```
pi@raspi_locis:~$ lsmod
Module                Size
bnep                  12051
hci_uart              20020
btbcm                  7916
bluetooth             365511
brcmfmac              222874
brcmutil              9092
cfg80211              543027
rfkill                20851
spidev                7373
snd_bcm2835           24427
snd_pcm               98501
snd_timer             23968
snd                   70032
i2c_bcm2835           7167
```

Ilustración 126 Controlador del bus i2c actual, visto con el comando “lsmod”. Fuente: Elaboración propia.

- Se descarga la versión anterior de controlador del bus desde un repositorio:



Ilustración 127 Descarga del archivo correspondiente (el link de descarga se encuentra en la bibliografía). Fuente: Elaboración propia.

- Guardar el archivo DTBO en la ruta /boot/overlays de la Raspberry:

```
-rwxr-xr-x 1 root root 886 may 4 13:16 i2c1-bcm2708.dtbo
```

Ilustración 128 Archivo en la ruta /boot/overlays. Fuente: Elaboración propia.

- Editar /boot/config.txt con sudo nano /boot/config.txt
- Al final añadir dtoverlay=i2c1-bcm2708 y dtparam=i2c1=on

```
GNU nano 2.2.6 Fichero: /boot/config.txt
# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

# NOOBS Auto-generated Settings:
hdmi_force_hotplug=1
start_x=1
gpu_mem=560
enable_uart=1
dtoverlay=w1-gpio
core_freq=250
dtoverlay=i2c1-bcm2708
dtparam=i2c1=on
```

Ilustración 129 Fichero editado en /boot/config.txt. Fuente: Elaboración propia.

- Editar el fichero /etc/modules (responsable de ejecutar diferentes módulos del Kernel al iniciar la Raspberry)

```

pi@raspi_locis:/etc
GNU nano 2.2.6                                Fichero: modules

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

snd-bcm2835
i2c-bcm2708
i2c-dev

```

Ilustración 130 Fichero editado en /etc/modules. Fuente: Elaboración propia.

- Guardar y reiniciar el sistema con sudo reboot

```

pi@raspi_locis:/boot $ sudo vcdbg log msg |& grep -v HDMI
000943.672: brfs: File read: /mfs/sd/config.txt
000944.738: brfs: File read: 1764 bytes
000980.252: brfs: File read: /mfs/sd/config.txt
001278.879: *** Restart logging
001278.905: brfs: File read: 1764 bytes
001280.669: brfs: File read: /mfs/sd/cmdline.txt
001280.726: Read command line from file 'cmdline.txt'
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p7 rootfstype=ext4 ele
eadline fsck.repair=yes rootwait
001280.992: brfs: File read: 113 bytes
001575.307: brfs: File read: /mfs/sd/kernel7.img
001575.331: Loading 'kernel7.img' to 0x8000 size 0x45dfb0
001578.964: No kernel trailer - assuming DT-capable
001578.993: brfs: File read: 4579248 bytes
001583.656: brfs: File read: /mfs/sd/bcm2710-rpi-3-b.dtb
001583.681: Loading 'bcm2710-rpi-3-b.dtb' to 0x465fb0 size 0x44d8
001689.825: brfs: File read: 17624 bytes
001693.110: brfs: File read: /mfs/sd/config.txt
001693.156: brfs: File read: 1764 bytes
001705.266: brfs: File read: /mfs/sd/overlays/pi3-miniuart-bt.dtbo
001716.832: Loaded overlay 'pi3-miniuart-bt'
001717.357: dtparam: i2c_arm=on
001726.771: dtparam: spi=on
001734.635: dtparam: audio=on
001784.028: brfs: File read: 1105 bytes
001800.036: brfs: File read: /mfs/sd/overlays/wl-gpio.dtbo
001806.695: Loaded overlay 'wl-gpio'
001844.368: brfs: File read: 1116 bytes
001852.587: brfs: File read: /mfs/sd/overlays/i2c1-bcm2708.dtbo
001861.138: Loaded overlay 'i2c1-bcm2708'
002900.385: Device tree loaded to 0x1bfeb600 (size 0x49e3)
002902.678: gpioman: gpioman_get_pin_num: pin SDCARD_CONTROL_POWER not de
004339.552: vchiq_core: vchiq_init_state: slot_zero = 0xdb580000, is_mast
004350.208: TV service:host side not connected, dropping notification 0x0
, 0x00000001, 0x00000010
020724.801: camsubs: Looking for camera 0: i2c_port = 0, led gpio = 134,
nable gpio = 133
020726.435: camsubs: Camera found OK
020729.524: gpioman: gpioman_get_pin_num: pin CAMERA_LED not defined
020737.718: brfs: File read: 886 bytes

```

Ilustración 131 Loaded overlay de forma correcta. Fuente: Elaboración propia.

El uso de los pines GPIO como fuente de voltaje puede acarrear problemas debido a que se estaría funcionando en el rango mínimo de voltajes para los cuales es apto el dispositivo Lidar.

Por lo tanto se realiza la misma prueba con una fuente de tensión externa (se usa para ello la placa Arduino como fuente de alimentación), con los siguientes cambios en el conecionado:

- 5V de tensión proveniente de la placa Arduino.
- Ground común entre el dispositivo Lidar y ambas placas.

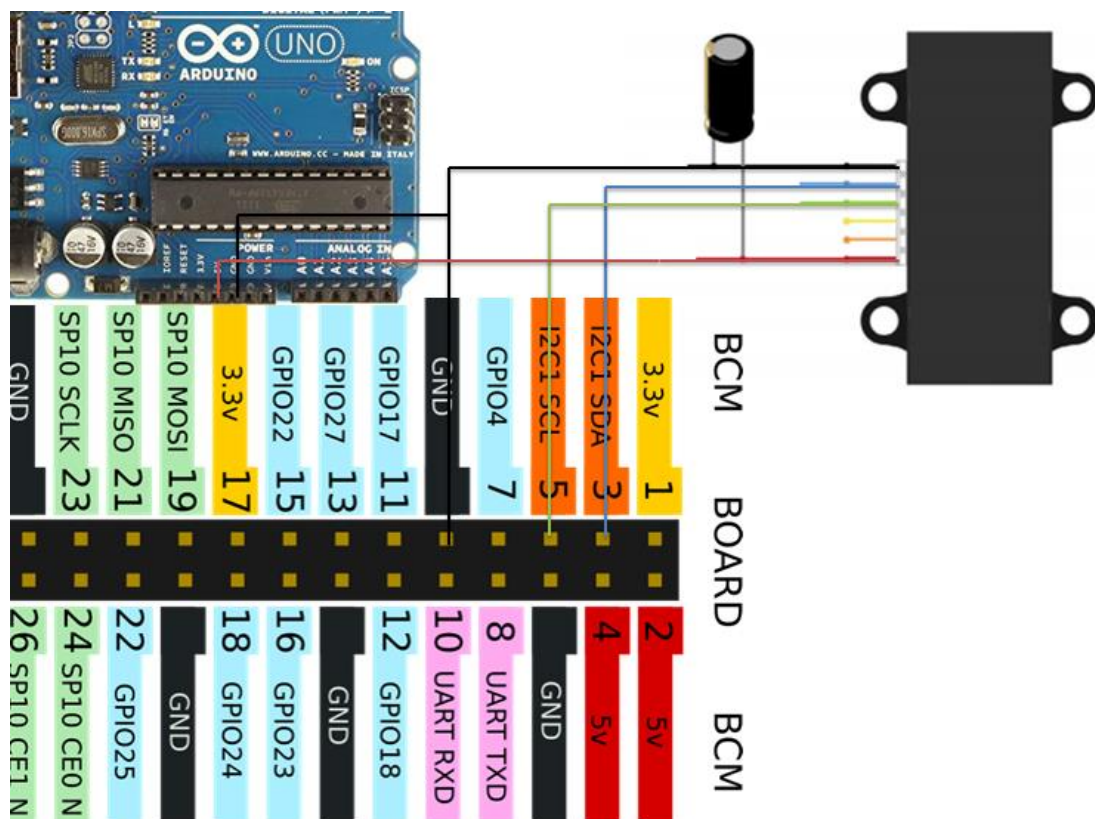


Ilustración 132 Alimentación con fuente externa. Fuente: Elaboración propia.

De la misma manera que cuando se alimentaba el Lidar con la Raspberry se producen lecturas de 0 cm, de modo que se repite el problema.

Al no haber funcionado con solamente cambiar la versión del controlador del bus, ni usando una fuente de tensión externa se probará cambiando la versión de todo el Kernel.

Para volver a versiones posteriores de Kernel en cualquier aplicación Linux se deberán seguir los siguientes pasos:

- Escribimos en la consola de comandos: `sudo apt-get install rpi-update`
- Consultar el repositorio del firmware de la Raspberry para descargar una versión anterior, consultando en foros se recomienda la versión 4.4.38.

- Se abre el repositorio en cuestión y en la barra de direcciones se copia la secuencia numérica correspondiente a la versión del Kernel que nos interese:

<https://github.com/Hexxeh/rpi-firmware/tree/af9cb14d5053f89857225bd18d1df59a089c171e>

Ilustración 133 Versión correcta. Fuente: GitHub.

- Posteriormente tecleamos por consola:
 - `sudo rpi-update <secuencia numérica de la versión del Kernel>`

```
pi@raspi_locis:~ $ sudo rpi-update af9cb14d5053f89857225bd18d1df59a089c171e
```

Ilustración 134 Comando por consola para "Down grade" del "Kernel". Fuente: Elaboración propia.

```
pi@raspi_locis:~ $ sudo rpi-update af9cb14d5053f89857225bd18d1df59a089c171e
*** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
*** Performing self-update
*** Relaunching after update
*** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
This update bumps to rpi-4.4.y linux tree
Be aware there could be compatibility issues with some drivers
Discussion here:
https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=144087
#####
*** Downloading specific firmware revision (this will take a few minutes)
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  168      0  168    0     0    247      0  --:--:-- --:--:-- --:--:--    247
100 38.3M    0 38.3M    0     0   258k      0  --:--:--  0:02:32 --:--:-- 58203
100 51.8M    0 51.8M    0     0   163k      0  --:--:--  0:05:24 --:--:-- 299k
*** Updating firmware
*** Updating kernel modules
*** depmod 4.4.38-v7+
*** depmod 4.4.38+
*** Updating VideoCore libraries
*** Using HardFP libraries
*** Updating SDK
*** Running ldconfig
*** Storing current firmware revision
*** Deleting downloaded files
*** Syncing changes to disk
*** If no errors appeared, your firmware was successfully updated to af9cb14d50
53f89857225bd18d1df59a089c171e
*** A reboot is needed to activate the new firmware
pi@raspi_locis:~ $
pi@raspi_locis:~ $ sudo reboot
```

Ilustración 135 Down grade de la versión del Kernel exitosa. Fuente: Elaboración propia.

```
pi@raspi_locis:~ $ uname -r
4.4.38-v7+
```

Ilustración 136 Nueva versión del Kernel. Fuente: Elaboración propia.

Se siguen produciendo outputs de 0 cm de distancia, debido a esto y a no encontrar una solución al problema se opta por intentar una conexión usando PWM.

10.3.3.- Tercera prueba toma de medidas (Raspberry y PWM)

Primero se realiza la prueba usando la placa Arduino como fuente de alimentación externa, por las mismas razones que se daban anteriormente en el conexionado I2C.

Los esquemas de conexión real y teórico se muestran a continuación en las siguientes imágenes:

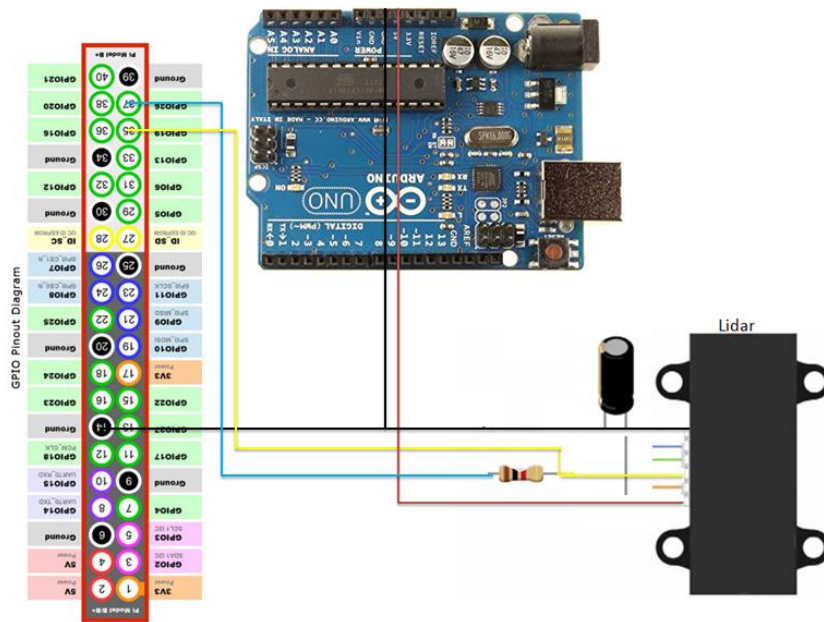


Ilustración 137 Esquema teórico de conexiones PWM Lidar y Raspberry pi (Usando la placa Arduino como fuente de alimentación externa). Fuente: Elaboración propia.

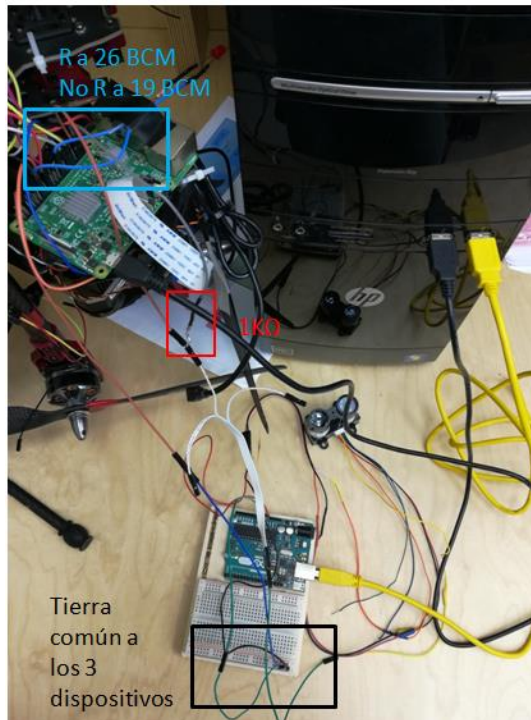


Ilustración 138 Esquema de conexión real de Lidar y Raspberry pi (Usando la placa Arduino como fuente de alimentación externa). Fuente: Elaboración propia.

Posteriormente se intenta recrear una versión en lenguaje Python del código Arduino que permitía tomar las medidas de distancia en la primera prueba:

```

GNU nano 2.2.6      Archivo: PWM_lidar_rasp.py
import RPi.GPIO as gpio
import time
import datetime

#hay mas pines con pwm para probar
pwm_input=19
pwm_output=26

gpio.setmode(gpio.BCM)
gpio.setwarnings(False)

gpio.setup(pwm_output,gpio.OUT)#actu
gpio.setup(pwm_input,gpio.IN)#actu
gpio.output(pwm_output,gpio.LOW)#e

def contar_microsec():
    millis = int(round(time.time()*1000000))
    return millis
while True:

    #contar cuanto tiempo el input es HIGH
    while not gpio.input(pwm_input):
        continue

    start= contar_microsec()
    while gpio.input(pwm_input):
        continue
    end=contar_microsec()
    pulseWidth= (end-start)
    if(pulseWidth != 0):
        pulseWidth= pulseWidth/10
        print pulseWidth

```

```

pi@raspi_locis: ~/Desktop/CarpetaCompartidaPRUEBA
181
181
180
181
181
183
181
181
181
^CTraceback (most recent call last):
  File "PWM_lidar_rasp.py", line 28, in <module>
    while gpio.input(pwm_input):
KeyboardInterrupt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python PWM_lidar_rasp
182
182
183
183
181
182
182
182
183
183
183
182
182
183

```

```

pi@
13
16
12
11
17
15
12
13
12
13
10
12
15
14
12
14

```

Ilustración 139 Script y repetición de la prueba con Arduino Fuente: Elaboración propia.

Este script realiza la siguiente sucesión de comandos:

- Importación de librerías necesarias.
- Inicialización de variables, designación de pines GPIO en sus diferentes modos:

De igual manera que en la versión .ino tenemos un pin_input, usado como monitor, y un pin_output, usado como ejecutor.

Un bucle infinito se encarga de dar las medidas de distancia haciendo la conversión de microsegundos a cm. (dividiendo entre 10 si tenemos en cuenta la velocidad del haz láser).

Una vez comprobado el funcionamiento mediante la consola de la Raspberry, se pretende usar también la alimentación proveniente de esta, para ello se usa el esquema de conexión teórico definido en el apartado de hardware.

En este caso el esquema real del montaje quedaría del siguiente modo:

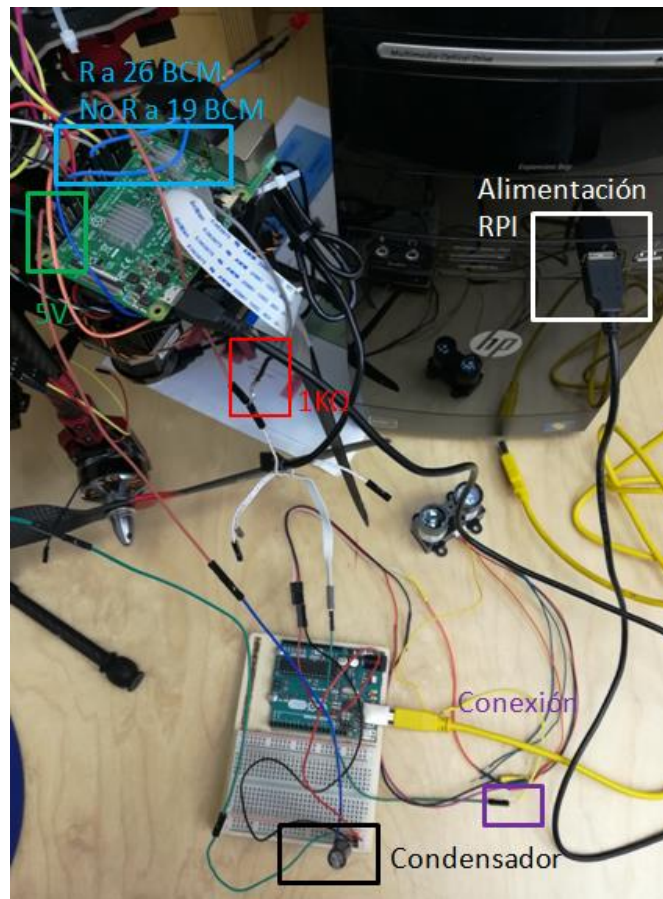


Ilustración 140 Montaje real cableado Lidar PWM con Raspberry. Fuente: Elaboración propia.

Ejecutando el mismo script que en el caso de la fuente de alimentación externa se consiguen las mismas medidas.

10.3.4.- Exportar medidas tomadas por el Lidar a un fichero .txt

Para poder acceder cómodamente a las alturas registradas después de cada misión se exportan a un fichero ubicado dentro de la carpeta compartida, actualizando para ello el script de la siguiente manera:

```
import RPi.GPIO as gpio
import time
import datetime

#hay mas pines con pwm para probar
pwm_input=19
pwm_output=26

gpio.setmode(gpio.BCM)
gpio.setwarnings(False)

gpio.setup(pwm_output,gpio.OUT)#actua como trigger
gpio.setup(pwm_input,gpio.IN)#actua como monitor

gpio.output(pwm_output,gpio.LOW)#el trigger como LOW para lectura continua
try:
    fecha=time.strftime("%H-%M-%S")
    ru='/home/pi/Desktop/CarpetaCompartidaPRUEBA/informes_Lidar/'+fecha+'.txt'
    f=open(ru,'w+')

    lidar_list=[]

    def contar_microsec():
        millis = int(round(time.time()*1000000))
        return millis
    while True:
        #contar cuanto tiempo el input es HIGH
        while not gpio.input(pwm_input):
            continue
        start= contar_microsec()
        while gpio.input(pwm_input):
            continue
        end=contar_microsec()
        pulseWidth= (end-start)
        if(pulseWidth != 0):
            pulseWidth= pulseWidth/10
            print pulseWidth
            lidar_list.append(pulseWidth)
            f.write("%s\t" %pulseWidth)
            f.write("\n")
except KeyboardInterrupt:
    print ("Saliendo de programa a peticion")
```

Ilustración 141 Exportación de datos de altura a la carpeta compartida. Fuente: Elaboración propia.

En donde primero se crea un archivo nombrado en base a la hora de la misión, en donde se pueden leer las diferentes alturas registradas en esta.

Se entiende entonces que este script debe ejecutarse durante la misión de forma automática.

Para la salida de los informes generados es importante saber que en el sistema operativo Linux (el que usa la Raspberry) el retorno de carro para nueva línea se especifica como \n,

pero en Windows (sistema operativo en el que se vería la información al sincronizar la Raspberry con la estación de tierra) el retorno de carro se especifica como `\r\n`.

Por lo tanto para la correcta visualización de los datos es necesario realizar ese cambio.

```
181 10-48-15
181 10-48-15
181 10-48-15
181 10-48-15
181 10-48-15
181 10-48-15
181 10-48-15
181 10-48-15
181 10-48-15
181 10-48-15
```

Ilustración 142 Output en forma de .txt para el Lidar, siguiendo el esquema altura registra en una marca temporal específica. Fuente: Elaboración propia.



PWM_lidar_rasp.py

10.1 Script funcionamiento Lidar. Fuente: Elaboración propia.

10.3.5.- Exactitud en las mediciones del dispositivo Lidar.

Se pretende estimar la exactitud de las medidas tomadas con el Lidar de forma experimental, teóricamente el dispositivo ofrece unos errores (según su documentación técnica del siguiente orden):

Performance

Specification	Measurement
Range (70% reflective target)	40 m (131 ft)
Resolution	+/- 1 cm (0.4 in.)
Accuracy < 5 m	±2.5 cm (1 in.) typical*
Accuracy ≥ 5 m	±10 cm (3.9 in.) typical Mean ±1% of distance maximum Ripple ±1% of distance maximum
Update rate (70% Reflective Target)	270 Hz typical 650 Hz fast mode** >1000 Hz short range only
Repetition rate	~50 Hz default 500 Hz max

Ilustración 143 Actuación del dispositivo. Fuente: <https://static.garmin.com/pumac/LIDAR Lite v3 Operation Manual and Technical Specifications.pdf> Acceso: 8/5/2018.

Como es lógico, en el entorno de la oficina solo se podrá probar el comportamiento del dispositivo en medidas a menos de cinco metros, siendo necesario el acoplamiento al dron antes de poder realizar pruebas a más de esta distancia.

11. Integración FPV

11.1.- INTRODUCCIÓN

Se pretende realizar la integración de los datos GPS (latitud y longitud) procedentes de la controladora de vuelo y los datos de altura procedentes del Lidar con datos de imagen en formato video procedentes de la cámara RGB.

Para ello se usará un dispositivo llamado OSD (“On Screen Display” o visualización en pantalla) que integra los datos antes mencionados en el video.

Una vez conseguido dicho output (video con datos) se enviará por medio de una antena transmisora (Tx) hacia una antena receptora (Rx) conectada a una pantalla o visor de alguna clase, siguiendo un esquema del siguiente tipo:

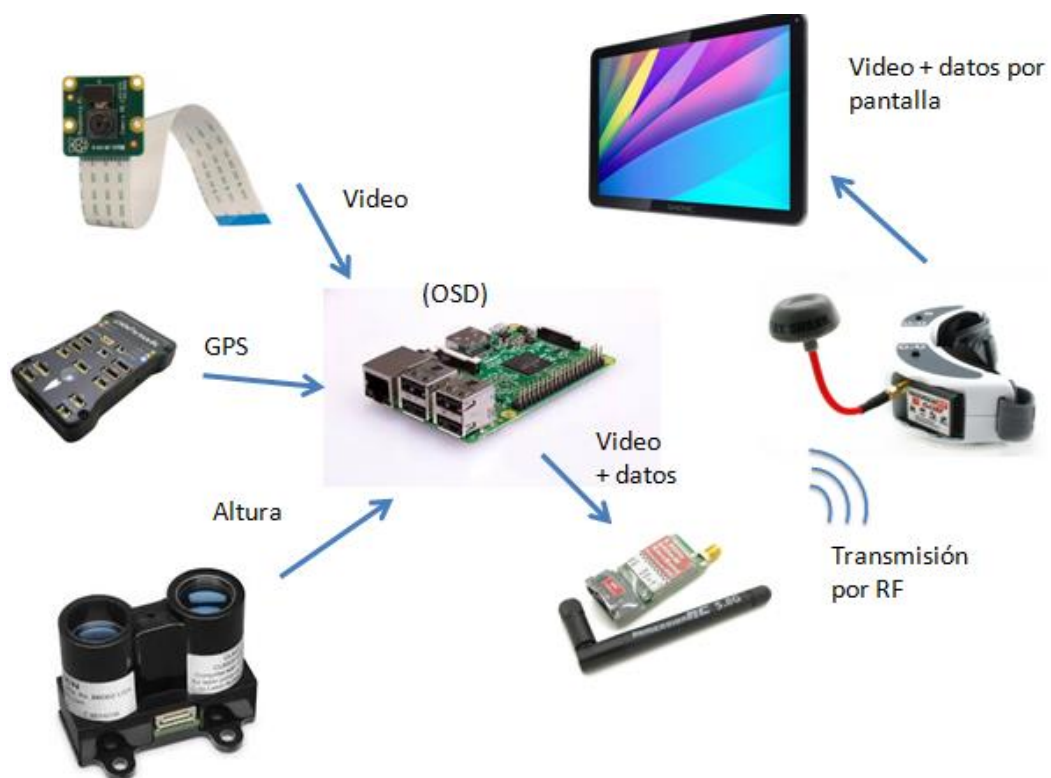


Ilustración 144 Esquema de conexiones. Fuente: Elaboración propia.

Para evitar poner más dispositivos en el dron y disminuir su peso se pretende realizar un programa python que se ejecute desde la Raspberry que emule la función del OSD (prescindiendo del uso de éste).

Los pasos por seguir para la consecución del output son los siguientes:

- Integración de datos en video usando un .Py que emule la función del OSD

Para luego realizar la transmisión del output de dos maneras:

- Opción A: Usar la Raspberry de forma que emule el funcionamiento del dispositivo Tx, necesitándose solo una antena para la transmisión de información al dispositivo Rx.
- Opción B: Usar un dispositivo Tx comercial (con su antena) para realizar la transmisión de información al Rx.

11.2.- DISPOSITIVOS REQUERIDOS

11.2.1.- Transmisor (Tx)

El dispositivo se trata de un Raceband de RC Immersion:



Ilustración 145 Tx de Rc Innovations. Fuente: <https://rc-innovations.es/ImmersionRC-transmisor-FPV-5.8Ghz-600mw-race-band>



Ilustración 146 Tx de Rc Innovations (parte trasera). Fuente: Elaboración propia.

11.2.1.1.- Conexiones del dispositivo

En este dispositivo encontramos las siguientes conexiones:

- Alimentación: Se compone de GND y Batt (se puede alimentar directamente con batería Li-Po 4s).
 - Usa un conector JST con cable



Ilustración 147 Conector JST con cable. Fuente: RcInnovations Acceso:17/5/2018.

- FPV (First Person View): Se compone de AUDR (audio “right” u oído derecho), AUDL (“left”), VID (video), GND, 5V Out (regulador interno para alimentar cámaras de 5V).
 - Usa una conexión serial TTL con cable
- Conector de antena SMA hembra

Se pretende realizar la alimentación desde la propia Raspberry, para evitar el acoplamiento de otra batería en el dron.

11.2.1.2.- Especificaciones técnicas importantes

- Input de video: 1Vpp, 75 Ohm. Se trata de un tipo de video denominado “composite” o compuesto.

La Raspberry es capaz de producir output de este tipo.

11.2.1.3.- Canales de frecuencias

La frecuencia de transmisión para señales de audio y video se centra en la banda de los 5.8 GHz, especificando las siguientes frecuencias:



Ilustración 148 Tabla de frecuencias del dispositivo. Fuente: Elaboración propia.

Por lo que, para que un emisor y un receptor puedan comunicarse deben estar en la misma frecuencia.

11.2.1.4.- Uso dentro del proyecto

Se pretende, como primera línea de trabajo, acoplar la antena directamente a la Raspberry para la transmisión de la información.

En segunda opción se plantea la integración del dispositivo “Raceband Tx” en el proyecto.

11.2.2.- OSD (“on screen display”)

En la oficina se cuenta con un dispositivo OSD denominado MAVLINK-OSD de la marca CRIUS.

Este pequeño dispositivo permite acoplar datos de entrada (desde la controladora de vuelo por ejemplo) a un video, para ello cuenta con los siguientes puertos:

11.2.2.1.- Puertos de entrada

- Entrada de datos: Se corresponden con los puertos que encontramos en el lado izquierdo en la imagen siguiente, por ellos se produce la entrada de los datos que se incrustan en el video. Son DTR, TX, RX, GROUND.
- Alimentación: Se corresponde con los símbolos Batt (+ de una batería) y Ground.
- Entrada de video: Imágenes sobre las cuales se produce el acoplamiento de datos.

11.2.2.2.- Puertos de salida

- Salida de video con datos (Video Out), conectado al canal de transmisión (en este caso la antena “Raceband Tx”).

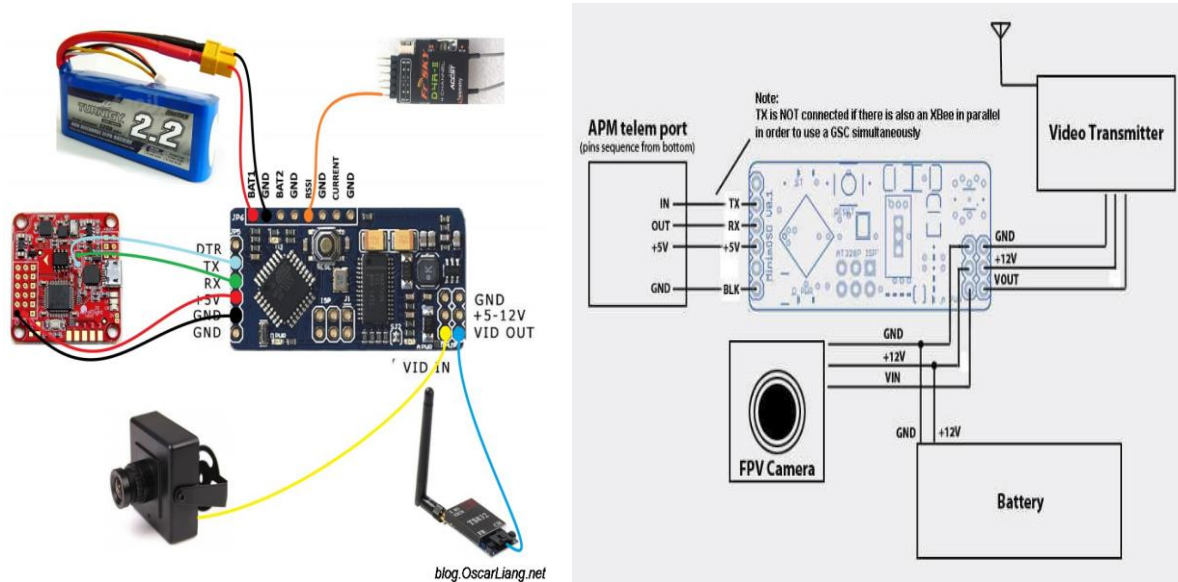


Ilustración 149 Conexiones dispositivo OSD Fuente: <https://oscarliang.com/fpv-guide/> Fuente: <http://ardupilot.org/copter/docs/common-minim-osd-quick-installation-guide.html>

11.2.2.3.- Uso dentro del proyecto

Se pretende que este dispositivo no forme parte del proyecto, por lo que será necesario desarrollar un script que emule su comportamiento.

11.3.- CAPTURA DE VIDEO Y “STREAM” VÍA WI-FI CON RASPI-CAM.

Como primera aproximación al FPV se pretende realizar la captación de video, ya sea por medio de una grabación como en formato “live” (“en vivo”) para posteriormente realizar su envío al PC por medio de una conexión Wi-Fi. De este modo:

Primero se debe habilitar el uso de la cámara, para ello:

- Se introduce el siguiente comando por consola: `sudo raspi-config`

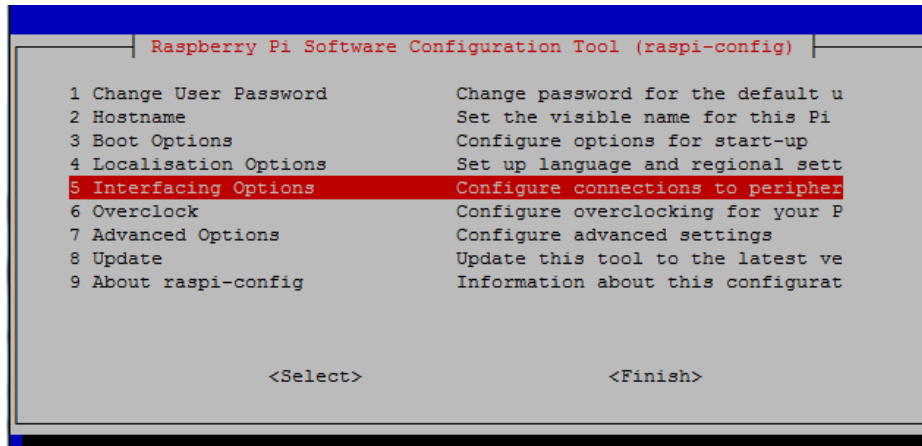


Ilustración 150 sudo raspi-config. Fuente: Elaboración propia.

- Se siguen los pasos de la interfaz gráfica

11.4.- CAPTURA DE VIDEO CON GSTREAMER

Se instala GStreamer 1.0 y libraspberrypi-dev, con el comando por teclado:

- `sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-bad1.0-dev gstreamer1.0-tools libraspberrypi-dev`

Instalar el módulo `gst-rpicamsrc`, un envoltorio de GStreamer que proporciona capturas de video desde la cámara Rpi, usando para ello un zip descargado del repositorio GitHub (<https://github.com/thaytan/gst-rpicamsrc>) con la siguiente sucesión de comandos:

- `sudo apt-get install zip unzip`
- `sudo unzip -d /home/pi ./gst-rpicamsrc-master.zip` (con el .zip descargado en el mismo directorio)
- Por consola y en el directorio en donde se guardaron los archivos descargados:
`sudo ./autogen.sh --prefix=/usr --libdir=/usr/lib/arm-linux-gnueabi/hf/make`
- `sudo make install`
- `make`

Haciendo ahora: `gst-launch-1.0 rpicamsrc bitrate=1000000 ! filesink location=test.h264` (en el directorio `/home/pi/rpicam`) se genera un archivo de video con nombre `test.h264`.



Ilustración 151 Archivo de video generado. Fuente: Elaboración propia.

11.5.- “STREAM”

Para iniciar live stream desde la Raspberry teclear por consola:

```
pi@raspi_locis:~/rpicam $ raspivid -t 999999 -o - | nc -l -p 5001
```

Ilustración 152 Comando por consola para iniciar “live”. Fuente: Elaboración propia.

El comando “nc” lanza la herramienta de red Unix netcat que lee y envía datos a través de conexiones network usando protocolos TCP ó UCP.

-t especifica la longitud del video en milisegundos.

-o guarda un video en el “path” elegido (no escribir opción implica el “path” actual).

5001 hace referencia al puerto de escucha.

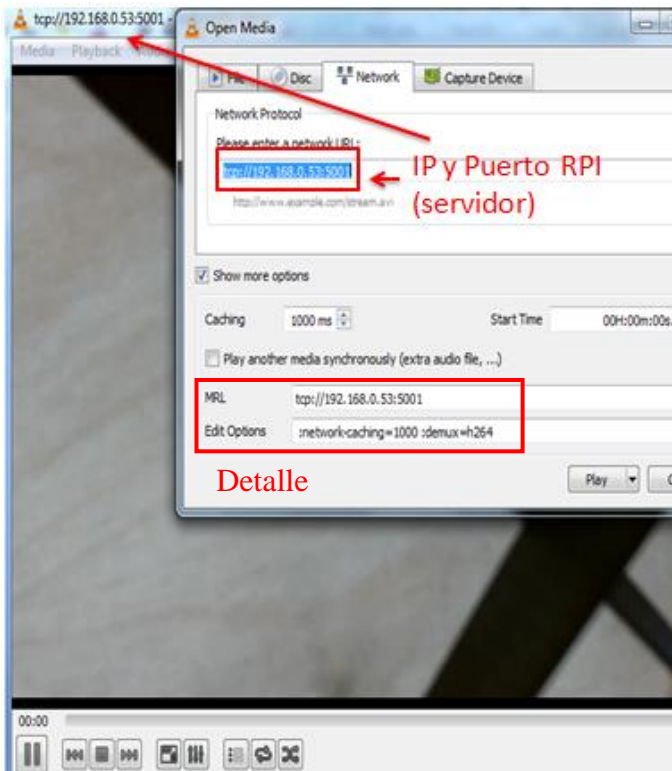


Ilustración 153 Cambios a realizar en reproductor VLC.
Fuente: Elaboración propia.

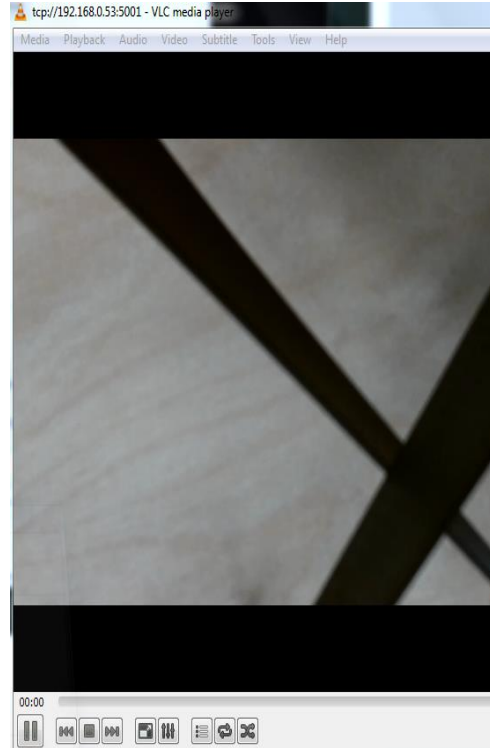


Ilustración 154 Output como "live" video.
Fuente: Elaboración propia.

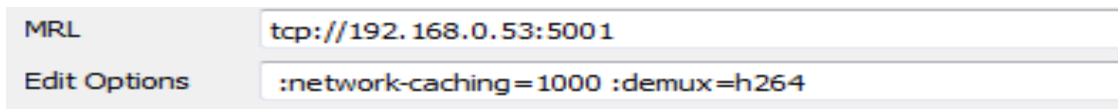


Ilustración 155 Vista detalle "edit option". Fuente: Elaboración propia.

11.6.- TRANSMISIÓN DE VIDEO DESDE LA RASPBERRY PI MEDIANTE EL PUERTO JACK

11.6.1.- Señal de video a transmitir: "Composite" video o video compuesto

Es una señal de video comúnmente usada en la transmisión de audio doméstico, los conectores usados en el transporte de datos para este tipo de línea son cables coaxiales de 75 Ohm de impedancia.

Por lo que coincide con las especificaciones técnicas en cuanto a input de video del "Rasband Tx" por radiofrecuencia.

Las formas de conseguir un output en forma de video desde la Raspberry pi son las siguientes:

- Usando el puerto "JACK" (video del tipo "composite"), forma que se usará.
- Usando el puerto HDMI (audio y video digital)

- Usando los pines GPIO

11.6.2.- Introducción

Se pretende conseguir una salida de video desde el puerto Jack de la Raspberry pi 3 para poder enviarlo por radiofrecuencia.

Para ello primero se localizará el puerto correspondiente en la Raspberry y dentro de este el canal dedicado al video.

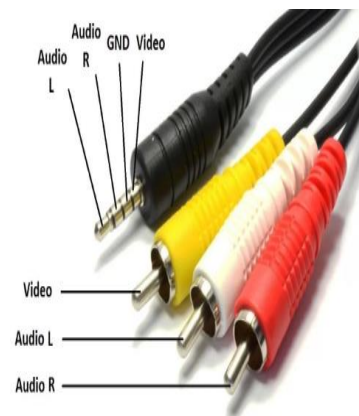
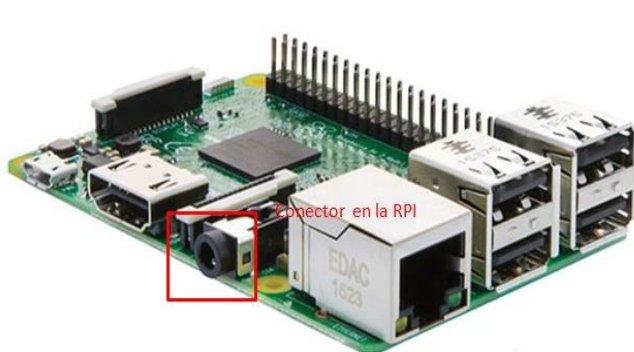


Ilustración 156 Conector Jack en la Raspberry pi. Fuente: https://es.gearbest.com/raspberry-pi/pp_488334.html

Ilustración 157 Diagrama de un cable destinado a un puerto Jack de 3.5mm y un cable adaptador RCA. Fuente: <https://descubriendolaorangeipi.wordpress.com/2017/01/10/salidas-de-audio-y-video-hdmi-y-rca/> Acceso: 21/05/18.

Device	Sleeve	Ring 2	Ring 1	Tip
	4	3	2	1
Model B+	Video	Ground	Right	Left

Ilustración 158 Diagrama puerto Jack TRRS (tip, ring, ring, sleeve) en Raspberry pi 3. Fuente: <https://www.raspberrypi-spy.co.uk/2014/07/raspberry-pi-model-b-3-5mm-audiovideo-jack/> .Acceso: 21/5/2018.

Por lo tanto, la solución pasa por aislar el cable dedicado al video para poder realizar su transmisión.

Para ello, otra opción es buscar como el conector Jack TRRS está mapeado dentro de la “board”.

El “test point” referente a video es el PP 24 en la parte de atrás del modelo B de la Raspberry pi 3, según la siguiente imagen:

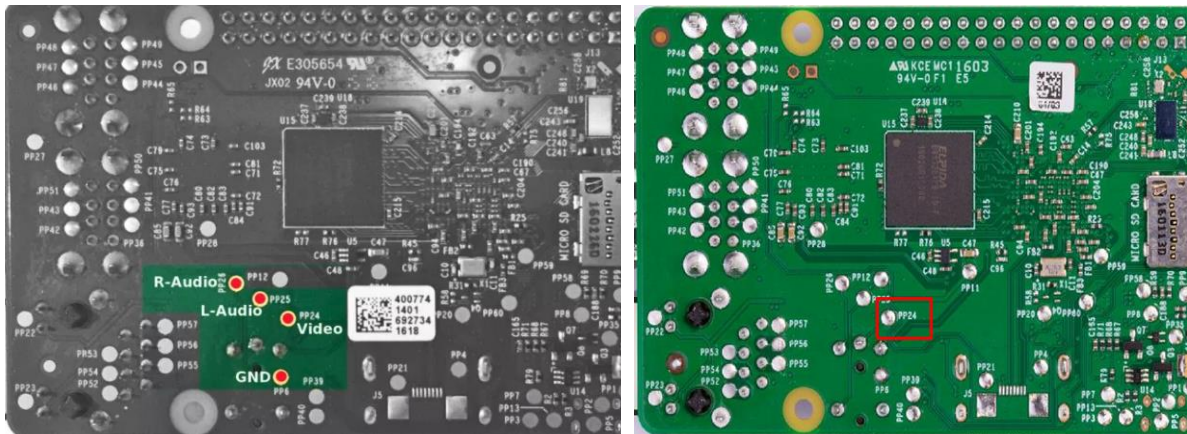


Ilustración 159 test points Raspberry. Fuente: <http://andidinata.com/2017/02/hack-it-no-av-cable-no-problem/> . Acceso: 21/5/2018.

Por lo que se puede realizar la salida de video usando:

- El “sleeve” del puerto Jack TRRS y un GPIO como Ground (el Ground debe provenir del mismo sitio que la señal de video).
- El PP 24 y un GPIO como Ground (el Ground debe provenir del mismo sitio que la señal de video).

11.6.3.- Software

Además de la realización de las conexiones físicas (hardware) es necesario editar el fichero config.txt de la Raspberry, para evitar que el dispositivo funcione por defecto con salida de video como HDMI (para ello comentamos todas las líneas referentes a este tipo de salida). A la par que habilitamos:

- sdtv_mode:

sdtv_mode	result
0	Normal NTSC
1	Japanese version of NTSC – no pedestal
2	Normal PAL
3	Brazilian version of PAL – 525/60 rather than 625/50, different subcarrier

Ilustración 160 Modo sdtv. Fuente: <https://www.raspberrypi.org/documentation/configuration/config-txt/video.md>

- sdtv_aspect:

sdtv_aspect	result
1	4:3
2	14:9
3	16:9

Ilustración 161 Aspecto de sdtv Fuente: <https://www.raspberrypi.org/documentation/configuration/config-txt/video.md>

```

GNU nano 2.2.6                               Fichero: /boot/config.txt
# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
#hdmi_drive=2
█
# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4

# uncomment for composite PAL europe
sdtv_mode=0
sdtv_aspect=1

```

Ilustración 162 sdtv en /boot/config.txt. Fuente: Elaboración propia.

11.6.4.- Primera prueba: Emisión de video desde la Raspberry a una pantalla

Se pretende enviar señal de video procedente de la cámara de la Raspberry a una pantalla externa, usando para ello un cable JACK a RCA (desde la Raspberry a la pantalla), según la siguiente conexión:

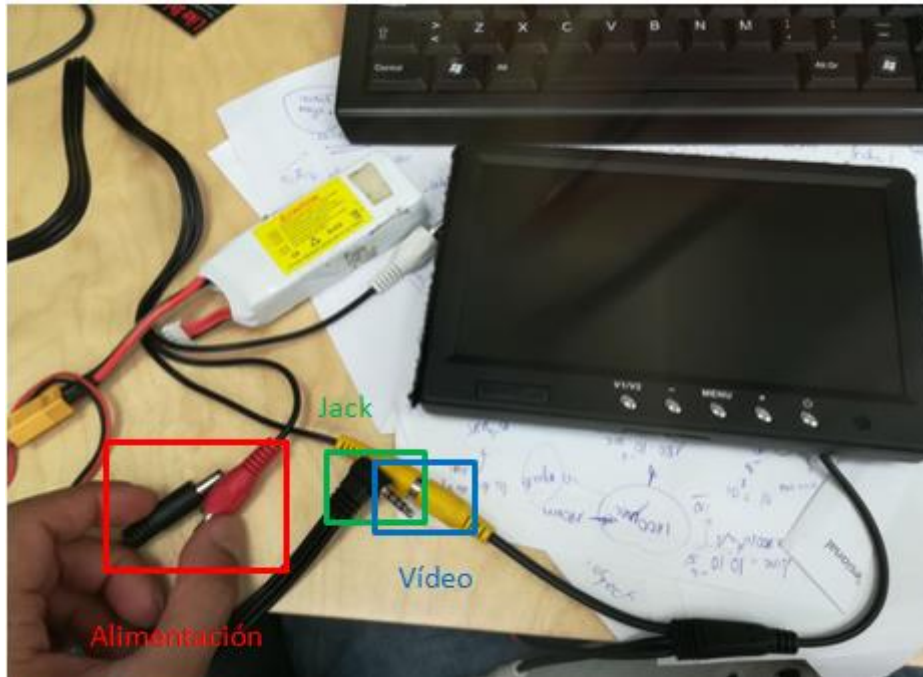


Ilustración 163 Montaje de la pantalla. Fuente: Elaboración propia.

Al conectar la alimentación se puede ver tanto el “desktop” de la Raspberry como la imagen procedente de la cámara, pero ambas con una pésima calidad de imagen.

Usando el mismo comando por consola que para el “stream” vía internet se consigue recibir imagen procedente de la “Rpicam”:



Ilustración 164 Desktop de la Raspberry pi desde la pantalla externa. Fuente: Elaboración propia.



Ilustración 165 Imagen de video procedente de la “Rpicam”. Fuente: Elaboración propia.

Por el momento se desconoce el porqué de la mala calidad de video pero se sospecha que pueda tratarse de un conflicto entre los diferentes estándares entre el Jack TRSS hembra de la Raspberry pi y el macho del adaptador RCA de la pantalla.

Device	Sleeve	Ring 2	Ring 1	Tip	OK?
	4	3	2	1	
Model B+	Video	Ground	Right	Left	✓
Apple	Video	Ground	Right	Left	✓
Zune	Video	Ground	Right	Left	✓
Camcorders	Right	Ground	Video	Left	⚠
MP3 Players	Ground	Video	Right	Left	✗

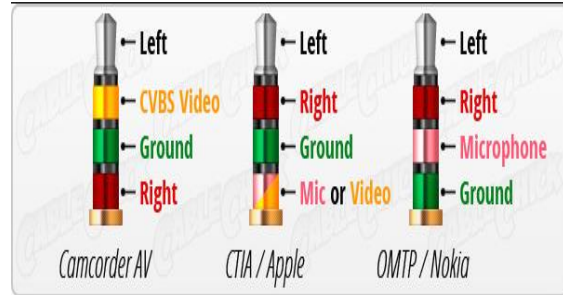


Ilustración 166 Diferentes estándares de Jack TRSS (en función del diferente fabricante). Fuente: <https://www.cablechick.com.au/blog/understanding-trrs-and-audio-jacks/> Acceso: 21/5/2018.

Otra posible causa puede deberse a una señal de tierra intermitente.

Se pretende probar una conexión punto a punto cableando, para ello:

11.6.5.- Segunda prueba: Funcionamiento de la emisora de radiofrecuencia.

Primero se prueba el funcionamiento correcto de la emisora de radiofrecuencia, con una cámara.

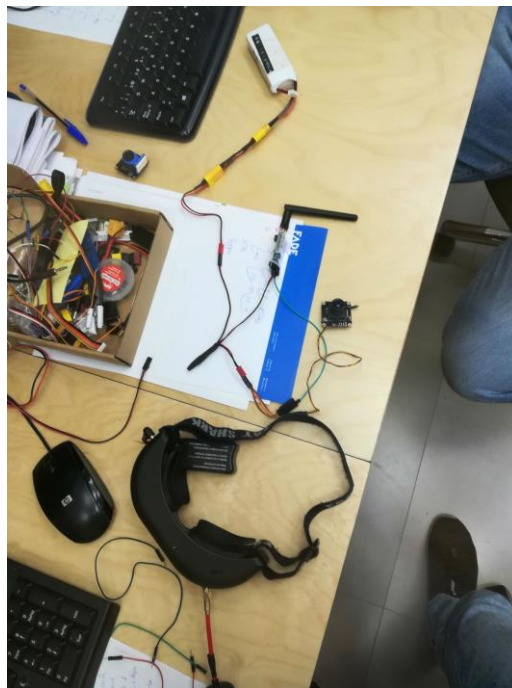


Ilustración 167 Prueba de funcionamiento de la emisora de radiofrecuencia. Fuente: Elaboración propia.

Por las gafas se logra ver imagen en tiempo real procedente de la cámara. Por lo que se sacan las siguientes conclusiones:

- La Tx y la Rx (emisión y transmisión por radiofrecuencia) están en la misma frecuencia y por lo tanto son capaces de comunicarse.
- El dispositivo Tx funciona de forma correcta.

11.6.6.- Transmisión de video desde Rpicam vía radiofrecuencia.

Se pretende transmitir la imagen de video procedente de la Rpicam a la pantalla por medio de radiofrecuencia, siguiendo un esquema del tipo:

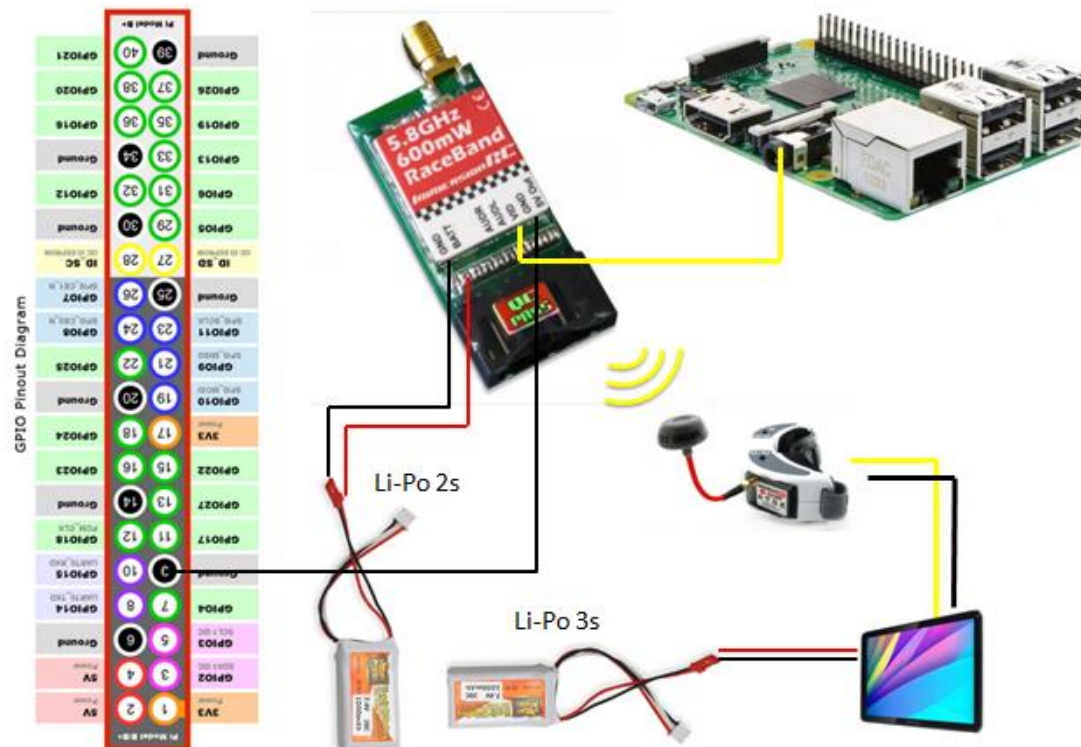


Ilustración 168 Esquema teórico de conexiones. Fuente: Elaboración propia.

En donde:

- Las líneas rojas hacen referencia a alimentación.
- Las líneas negras hacen referencia a Ground.
- Las líneas amarillas hacen referencia a señal de video.

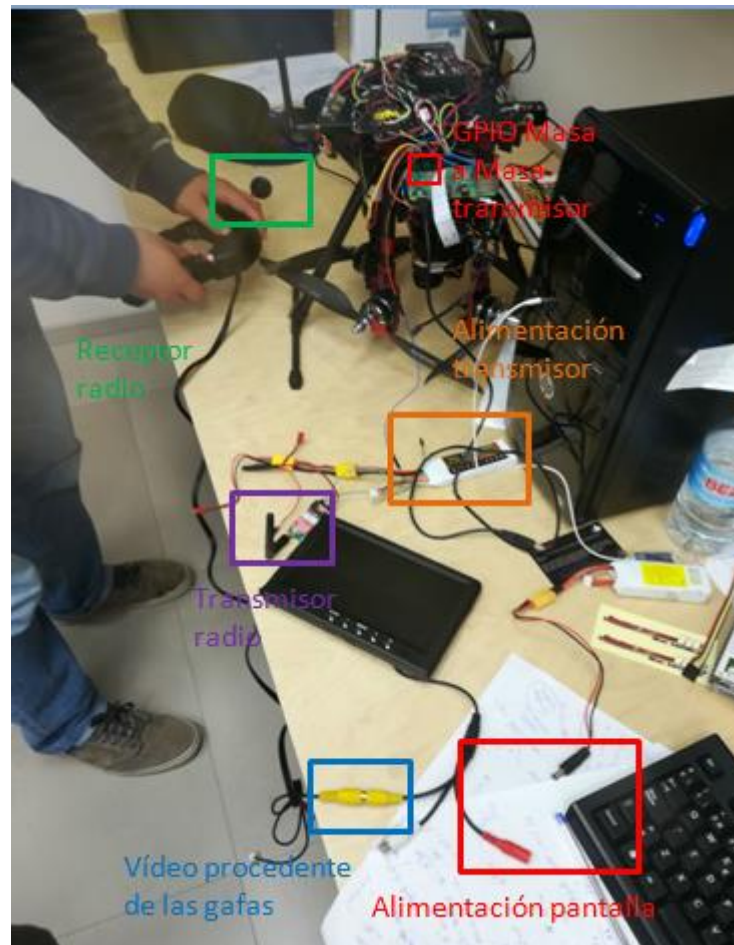


Ilustración 169 Montaje experimental. Fuente: Elaboración propia.

Una vez realizado el montaje:

- Se alimenta la Raspberry por medio de un USB desde la torre del ordenador.
- Se alimenta la Tx desde una batería externa Li-Po 2s (8V) de la marca RC.
- Se alimenta la pantalla desde una batería externa Li-Po 3s (12V) de la marca RC.
- Se alimentan las gafas desde su propia batería (suministrada por el fabricante).



Ilustración 170 Contacto entre el pin dedicado a video en el Jack TRRS de la Raspberry y la Tx. Fuente: Elaboración propia.

Una vez se ejecuta la orden desde la consola de la Raspberry pi y se produce contacto con el pin destinado a video del conector Jack TRRS, se produce la transmisión de imagen:

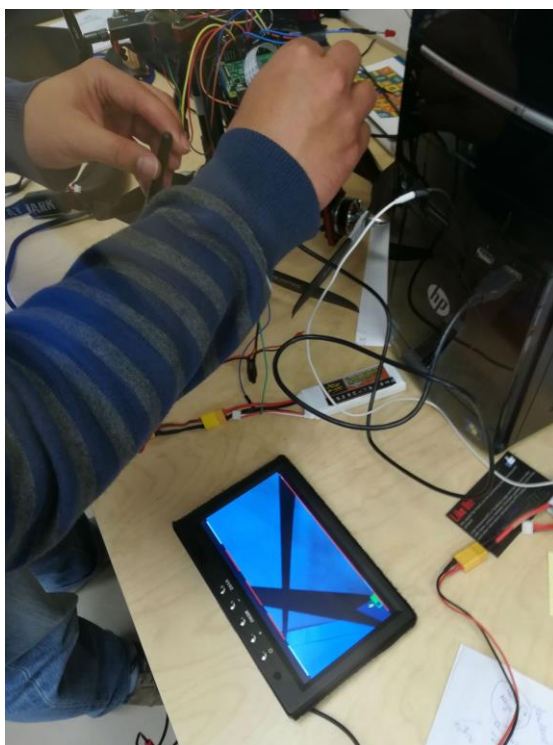


Ilustración 171 Visión por pantalla de la Rpicam. Fuente: Elaboración propia.

Por razones de sencillez se decide usar y soldar la salida de video marcada en la siguiente imagen.

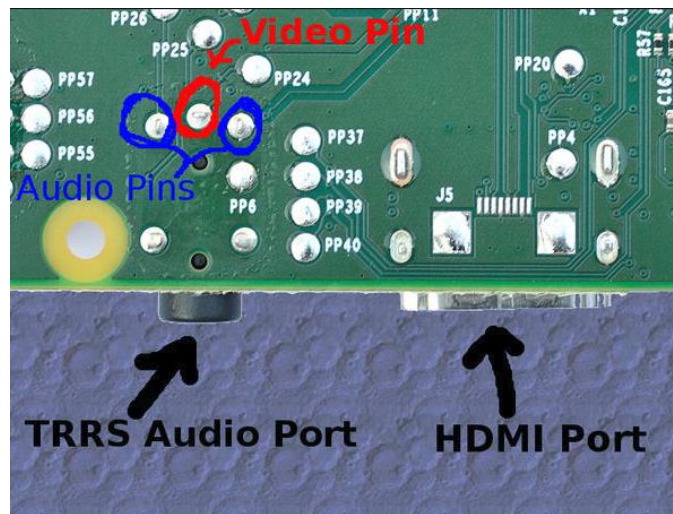


Ilustración 172 Salidas de video vía "test pads" PP24 o video pin marcado. Fuente: Elaboración propia.

11.7.- AÑADIDO DE DATOS AL VIDEO (EMULACIÓN POR MEDIO DE LA RASPBERRY DEL DISPOSITIVO OSD)

Se pretende añadir los datos al video "live" antes de transmitirse por radiofrecuencia a la pantalla, para que de este modo se produzca un output de video + datos.

Para ello se ejecuta un script Python como el siguiente:

```
from picamera import PiCamera
import datetime as dt
import time
camera = PiCamera()
camera.resolution = (1280, 720)
camera.start_preview()
camera.annotate_text = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
camera.start_recording('videopi.h264')
start = dt.datetime.now()
while True:
    camera.annotate_text = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    camera.wait_recording(0.2)
camera.stop_recording()
camera.stop_preview()
```

Ilustración 173 Script Python para añadir fecha al "live" video. Fuente: Elaboración propia.

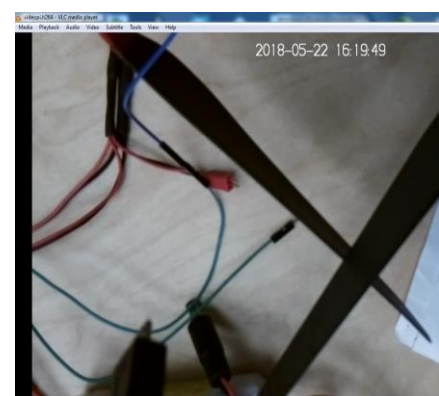


Ilustración 174 Timestamp video Rpicam. Fuente: Elaboración propia.

Se actualiza con la posición GPS desde la controladora (se obtiene de la misma forma que para el apartado de geotiquetado de fotos RGB) usando para ello el siguiente script:

```
try:
from dronekit import connect
import dronekit
lat=1
lon=1
vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)
def listener_gps_raw(self,name,posgps):#funcion oyente para GPS
#print ("lat:%s\tlon:%s\tsat.number:%s\tten:%s" %(posgps.lat
global lat,lon#,alt
lat=posgps.lat
lon=posgps.lon
# lat=int(lat)
lat=1.0*lat #para convertir de int a float
lat=lat/10000000#esta multiplicado por 10 elevado a 7
lat=str(lat)
# lon=int(lon)
lon=1.0*lon #para convertir de int a float
lon=lon/10000000#esta multiplicado por 10 elevado a 7
lon=str(lon)
alt=posgps.alt
#video con text overlay
from picamera import PiCamera
import datetime as dt
import time
camera = PiCamera()
camera.resolution = (1280, 720)
camera.start_preview()
camera.annotate_text = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
camera.start_recording('videopi.h264')
start = dt.datetime.now()
while True:
camera.annotate_text = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
time.sleep(0.1)#para darle tiempo al listener
camera.annotate_text = "lat=%s lon=%s" %(lat,lon)
camera.wait_recording(0.2)
camera.stop_recording()
camera.stop_preview()
except KeyboardInterrupt:
print("Saliendo a peticion")
```

Ilustración 175 Script para añadir fecha y GPS al video “live”. Fuente: Elaboración propia.

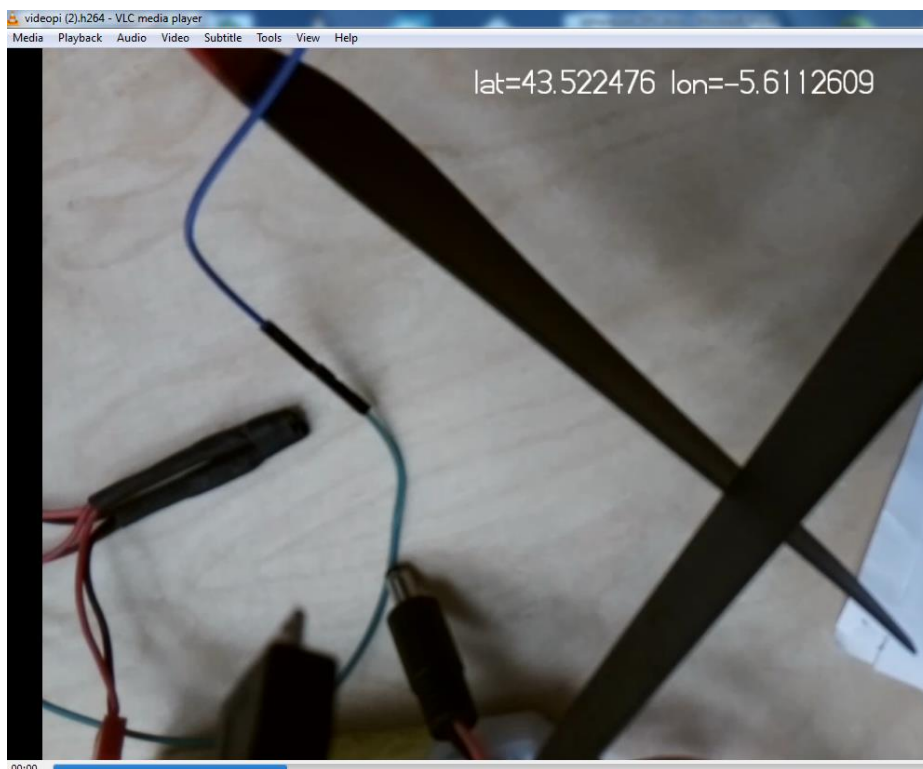


Ilustración 176 GPS "overlay" en video Rpicam. Fuente: Elaboración propia.

En el video se puede ver como el “Timestamp” se solapan con el GPS sustituyendo uno a otro (sin mostrarse ambos por pantalla), por lo que se optará por llamar una sola vez a la función `annotate_text` con todas las variables necesarias.

Del modo siguiente:

```
tiempo = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')#cambiar
vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
time.sleep(0.1)#para darle tiempo al listener
camera.annotate text = "lat=%s lon=%s time=%s" %(lat,lon,tiempo)
```



Ilustración 177 Actualización en el script. Fuente: Elaboración propia.

Ilustración 178 "Overlay" GPS y Timestamp. Fuente: Elaboración propia.

Para añadir el dato de la altura procedente del dispositivo Lidar se pretende leer el último dato que este arroje a un fichero .txt para posteriormente incrustarlo en el video.

Para ello se debe ejecutar el script del Lidar de forma automática, desde otro terminal. Para ello se ejecutará usando la función screen (de la que se hará uso con la ayuda de una serie de utilidades).

```
pi@raspi_locis:~ $ sudo pip install git+http://github.com/Christophe31/screenuti
ls.git
Collecting git+http://github.com/Christophe31/screenutils.git
  Cloning http://github.com/Christophe31/screenutils.git to /tmp/pip-req-build-v
8MchY
Building wheels for collected packages: screenutils
  Running setup.py bdist_wheel for screenutils ... done
  Stored in directory: /tmp/pip-ephem-wheel-cache-dgJWli/wheels/e2/c3/06/76c83ae
93ab9f77c7b5a0b74ed997c59ec1df9a7eff6c41404
Successfully built screenutils
Installing collected packages: screenutils
Successfully installed screenutils-0.0.1.6.2
```

Ilustración 179 Instalación de utilidades para multiplexación "screen". Fuente: Elaboración propia.

```
sudo chmod +s /usr/bin/screen
```

Ilustración 180 Cambio de derechos de la multiplexadora GNU screen. Fuente: Elaboración propia.

Con la última actualización del script se consigue incrustar la posición GPS proveniente de la controladora y la altura desde el dispositivo Lidar:

```
try:
    from screenutils import list_screens, Screen
    import time
    s=Screen("lid",True)
    print s
    s.send_commands('sudo python /home/pi/Desktop/CarpetaCompartidaPRUEBA/PWM_lidar_rasp.py')
    def get_alt():
        fileHandle = open ( 'informes_Lidar/lidar.txt','r' )
        lineList = fileHandle.readlines()
        fileHandle.close()
        return lineList[len(lineList)-1]#la ultima es vacia
    time.sleep(1)
    from dronekit import connect
    import dronekit
    lat=1
    lon=1
```

Ilustración 181 launchvid.py . Fuente: Elaboración propia.

```

def listener_gps_raw(self,name,posgps):#funcion oyente para GPS
    #print ("lat:%s\tlon:%s\tosat.number:%s\ten:%s" %(posgps.lat,
    global lat,lon#,alt
    lat=posgps.lat
    lon=posgps.lon
    # lat=int(lat)
    lat=1.0*lat #para convertir de int a float
    lat=lat/10000000#esta multiplicado por 10 elevado a 7
    lat=str(lat)

    # lon=int(lon)
    lon=1.0*lon #para convertir de int a float
    lon=lon/10000000#esta multiplicado por 10 elevado a 7
    lon=str(lon)
    alt=posgps.alt
def get_alt():
    fileHandle = open ( 'informes_Lidar/lidar.txt',"x" )
    #ojo si se cambia de sitio poner path abs
    lineList = fileHandle.readlines()
    fileHandle.close()
    return lineList[len(lineList)-1]

vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)

from picamera import PiCamera
import datetime as dt
import time
camera = PiCamera()
camera.resolution = (1280, 720)
camera.start_preview()
camera.start_recording('videopi.h264')
while True:
    tiempo = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')#cambiar nombre lib
    vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
    time.sleep(0.1)#para darle tiempo al listener
    alt=get_alt()
    camera.annotate_text = "lat=%s lon=%s time=%s alt=%s" %(lat,lon,tiempo,alt)
    print "lat=%s lon=%s time=%s alt=%s" %(lat,lon,tiempo,alt)
    camera.wait_recording(0.2)
    camera.stop_recording()
    camera.stop_preview()
except KeyboardInterrupt:
    print("Saliendo a peticion")
    s.kill()

```

El script se compone de los siguientes comandos:

- Multiplexación de terminal con la función screen, en donde se ejecuta el script dedicado a captar las alturas del Lidar (se hace referencia a la sesión con la variable “s”).
- Las alturas del Lidar se vuelcan en un fichero.txt desde donde se leen posteriormente desde la función get_alt(): para incluirlas como información de video por pantalla.
- Los datos GPS se obtienen desde la controladora por medio de un “listener” como se explica en el apartado referente a información GNSS (geo-tag).
- Las funciones referentes a la cámara provienen del módulo Picamera (importado al inicio del script).

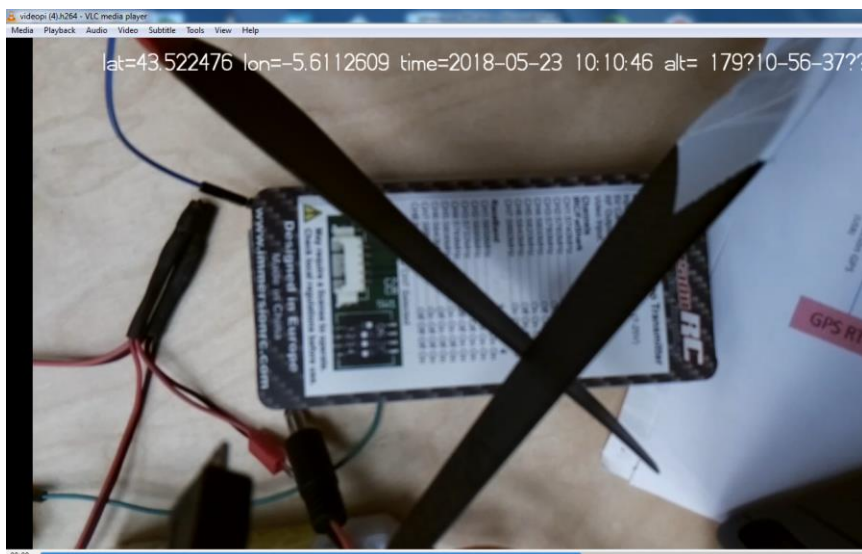


Ilustración 182 "Overlay" GPS Timestamp y altura. Fuente: Elaboración propia.

11.8.- CORRECCIÓN DE ERRORES

- Lectura cortada:

A veces las líneas correspondientes a datos de altura se leían cortadas, por esto es mejor leer el antepenúltimo elemento:

```
10-57-40\r\n', '185\t10-57-40\r\n', '185\t10-57-40\r\n', '184\t10-57-40\r\n\
84\t10-57-40\r\n', '174\t10-57-40\r\n', '184\t10-57-40\r\n', '162\t10-'
el corte es11
la linea a cortar es 184          10-57-40
```

Ilustración 183 Ejemplo de lectura cortada. Fuente: Elaboración propia.

- “List index out of range”:

A veces se produce un error de este tipo:

```
Traceback (most recent call last):
  File "utils.py", line 28, in <module>
    alt=get_alt()
  File "utils.py", line 19, in get_alt
    corte=len(lineList[len(lineList)-2])-lineList[len(lineList)-2].find("\t")
IndexError: list index out of range
```

Ilustración 184 “list index out of range”. Fuente: Elaboración propia.

Es debido a que al leer el archivo .txt que contiene las alturas el archivo está recién creado (y no contiene nada). De este modo `len(lineList)=0` y `lineList(0-2)` lo cual, al ser un número menor que cero queda fuera del rango del vector.

Para solucionar este problema debemos ir al script dedicado al Lidar, que cada vez que toma un valor de altura sobrescribe el fichero entero (al usar la opción `f.write(w)`).

Esto solo da problemas cuando coincide la operación de creación con la de cálculo de longitud del .txt.

Para corregir este error se debe cambiar el modo de escritura “w” por “a” (append) para continuar con los datos del .txt sin crear uno nuevo.

- 'r' – Read mode which is used when the file is only being read
- 'w' – Write mode which is used to edit and write new information to the file (any existing files with the same name will be erased when this mode is activated)
- 'a' – Appending mode, which is used to add new data to the end of the file; that is new information is automatically amended to the end
- 'r+' – Special read and write mode, which is used to handle both actions when working with a file

Ilustración 185 Modo de apertura de fichero. Fuente: <http://www.pythonforbeginners.com/files/reading-and-writing-files-in-python>

Además se añade un tiempo de retardo antes de leer el fichero en el bucle para así asegurar la existencia de valores.

Para facilitar la visualización se opta por solo mostrar los datos de altura sin la fecha de toma de esta (actualizando para ello la función) como se puede comprobar en el video siguiente:

```
def get_alt():
    global fileHandle
    time.sleep(2)
    lineList = fileHandle.readlines()
    time.sleep(0.1)
    corte=len(lineList[len(lineList)-2])-lineList[len(lineList)-2].find("\n")
    return lineList[len(lineList)-2][:-corte]#la ultima es vacia
```

Ilustración 186 Actualización de la función. Fuente: Elaboración propia.

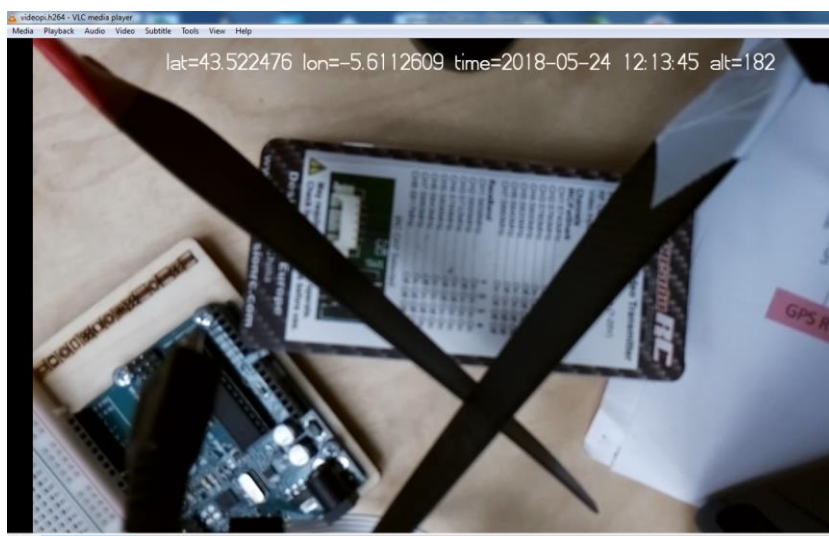


Ilustración 187 “Rpvideo” con corrección de errores. Fuente: Elaboración propia.

- Cliente único:

La cámara de la Raspberry solo es capaz de gestionar un cliente a la vez, de modo que no será capaz de recibir la orden de tomar una imagen cuando está grabando video (emitirá un error de recursos insuficientes).

```
picamera.exc.PiCameraMMALError: Camera component couldn't be enabled: Out of resources (othe
```

Ilustración 188 Ejemplo de error de cliente único. Fuente:
<https://raspberrypi.stackexchange.com/questions/26829/picamera-not-working>

Para solventar este error se para el video durante una fracción de segundo (imperceptible al ojo humano) pero en apariencia suficiente para dar la orden de toma de foto:

```
ruta='/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test%s.jpg'\nprint ruta\ncamera.wait_recording(0.3)\ncamera.capture(ruta,use_video_port=True)\ncamera.wait_recording(0.3)
```

Ilustración 189 Disparo de foto mientras se produce la grabación. Fuente: Elaboración propia.

Debido a esto en el video se pueden ver saltos de algunos segundos (aproximadamente 3).



launchvid.py

11.1 Actualización de programa. Fuente: Elaboración propia.

```
lat=43.522476 lon=-5.6112609 time=2018-05-28 10:59:23 alt=183\n/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test158.jpg\nse saco foto True\nlat=43.522476 lon=-5.6112609 time=2018-05-28 10:59:25 alt=182\n/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test159.jpg\nse saco foto True\nlat=43.522476 lon=-5.6112609 time=2018-05-28 10:59:26 alt=180\n/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test160.jpg\nse saco foto True\nlat=43.522476 lon=-5.6112609 time=2018-05-28 10:59:28 alt=182\n/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test161.jpg\nse saco foto True\nlat=43.522476 lon=-5.6112609 time=2018-05-28 10:59:30 alt=183\n/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test162.jpg\nse saco foto True\n█
```

Ilustración 190 Toma de fotos durante la emisión de video. Fuente: Elaboración propia.

11.9.- INTEGRACIÓN DE LAUNCHVID.PY EN EL DRON

Se pretende integrar el sistema FPV dentro del programa principal (automatico.py), para ello:

11.9.1.- Integración hardware:

La batería que alimenta la radio es demasiado grande y pesada para poder usarla en un vuelo, por lo que será necesario buscar métodos alternativos.



Ilustración 191 Batería que alimenta la radio. Fuente: <https://www.banggood.com/es>

11.9.1.1.- Alimentación de la radio vía los GPIO de la Raspberry:

Según la documentación de la radio Tx se requieren 600mW para alimentarla, de modo que sería necesaria una salida de más de 100mW por el GPIO de 5V de la Raspberry o más de 200mA por un GPIO de 3,3V.

Es posible alimentar la radio Tx conectándola al pin de salida de 5V y a alguno de los terminales GROUND disponibles en la board. (la prueba se realizó con la cámara FLIR desconectada).

Debido a no quedar GPIO GROUND disponibles se usarán los “test pads” para conectarse a la masa, los cuales según la documentación referente a la Raspberry son los puntos PP3, 4, 5 Y 6.

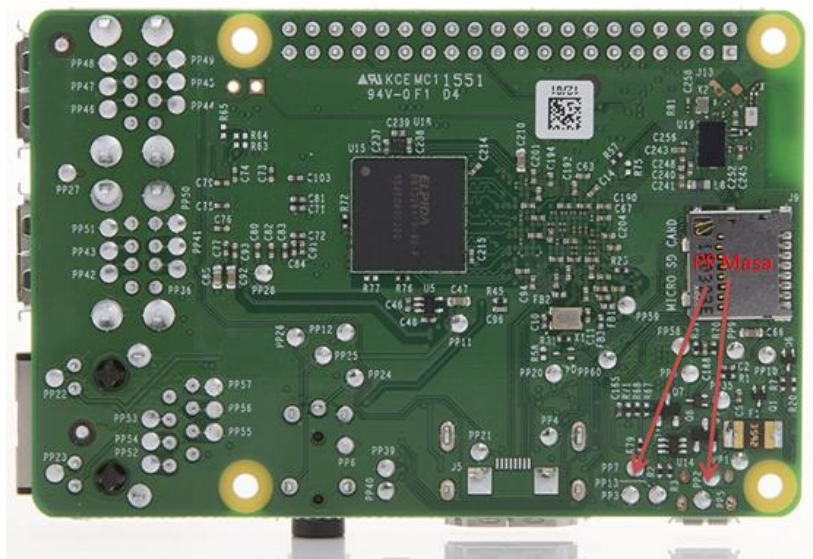


Ilustración 192 Algunos de los “test pads” referentes a masa. Fuente: Elaboración propia.

Asimismo, y aunque en principio no se estima necesario, se pueden disponer de salidas de 5V en los PP1 y PP2.

11.9.2.- Integración Software:

Las opciones de integración son:

- Directamente en automatico.py rehaciendo el programa
- Como terminal multiplexado llamado desde automatico.py

11.9.2.1.- Terminal multiplexado

Se pretende automatizar el proceso de ejecución de inicio de video, usando para ello una “detached screen” llamada desde automatico.py

Al ejecutar el script Python screenin.py, se genera una multiplexación en donde se ejecuta launchvid.py (por eso se pueden ver dos “desattached screens”).

Algunos comandos importantes para el uso de la funcionalidad screen son:

- Para parar un script se recurrirá al comando `sudo screen -S “nombre” -X “comando”`
- Ver la ejecución de un script (“re-attach”) se usa el comando `sudo screen -x nombre`.
- Los nombres los puede dar el usuario desde el script con `s=screen(nombre)`, habiendo importado antes la librería correspondiente.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python screenin.py
<Screen 'video'>
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo screen -S video -X quit
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo screen -ls
There is a screen on:
      2290.lid          (29/05/18 09:44:03)      (Detached)
1 Socket in /var/run/screen/S-root.

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo screen -s lid -X quit
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo screen -ls
No Sockets found in /var/run/screen/S-root.

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ █

```

Ilustración 193 Ejecución de programa Python para “multiplexar” el lanzamiento de video. Fuente: Elaboración propia.

```

s=Screen("video",True)
print s
s.send_commands('sudo python /home/pi/Desktop/CarpetaCompartidaPRUEBA/launchvid.py')

```

Ilustración 194 Comando que ejecuta la multiplexación de vid. Fuente: Elaboración propia.

Como se observa en las imágenes anteriores en la pantalla video donde se ejecuta “launchvid.py” se genera otra multiplexación con s.send_commands llamada lid en donde se ejecuta el script referente a medidas Lidar.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo screen -ls
There are screens on:
      1318.lid          (29/05/18 09:32:53)      (Detached)
      1290.video       (29/05/18 09:32:51)      (Detached)
2 Sockets in /var/run/screen/S-root.

```

Ilustración 195 Terminales multiplexadas. Fuente: Elaboración propia.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo screen -x video
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:39 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test63.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:41 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test64.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:43 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test65.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:45 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test66.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:47 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test67.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:49 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test68.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:51 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test69.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:53 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test70.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:55 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test71.jpg
se saco foto True
lat=43.522476 lon=-5.6112609 time=2018-05-29 09:35:57 alt=178
/home/pi/Desktop/CarpetaCompartidaPRUEBA/picameratest/test72.jpg
se saco foto True

```

Ilustración 196 "re-attach" a una terminal multiplexada. Fuente: Elaboración propia.

El problema está ahora en que no es posible conectarse desde dos procesos diferentes (dos scripts Python) al mismo vehículo (usando el mismo modo, serial en este caso), por lo que será necesario buscar otra forma de pasar datos GPS entre procesos (ya que no es posible escucharlos desde la controladora de forma simultánea).

```

vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)

```

Ilustración 197 Conexión con la controladora mediante puerto serial ttyAMA0. Fuente: Elaboración propia.

Para ello se volcarán los datos GPS desde launchvid.py a un .txt, desde donde se leerán en automatico.py (esto es debido a que el FPV debe funcionar en toda la misión, en una franja temporal mayor que el bucle infinito de señal de foto de automatico.py).

De este modo solo sería necesaria la conexión por ttyAMA0 de la controladora con uno de los procesos (script).

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo screen -S lid -X quit
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ ls -l GPS.txt
-rw-r--r-- 1 root root 700 may 29 12:30 GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo chmod 755 GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ ls -l GPS.txt
-rwxr-xr-x 1 root root 700 may 29 12:30 GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo chmod 777 GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ ls -l GPS.txt
-rwxrwxrwx 1 root root 700 may 29 12:30 GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ █

```

Ilustración 198 Otorgamos privilegios sobre el archivo .txt (usado para almacenar datos) a todos los usuarios. Fuente: Elaboración propia.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ ls -l informes_Lidar/lidar.txt
-rw-r--r-- 1 pi pi 10166272 may 29 13:08 informes_Lidar/lidar.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ ls -l GPS.txt
-rwxrwxrwx 1 root root 700 may 29 12:30 GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ chown pi: GPS.txt
chown: cambiando el propietario de «GPS.txt»: Operación no permitida
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo chown pi: GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ ls -l GPS.txt
-rwxrwxrwx 1 pi pi 4796 may 29 13:10 GPS.txt
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ █

```

Ilustración 199 Cambio de dueño del archivo. Fuente: Elaboración propia.

Una vez realizado el cambio de privilegios del archivo permitimos que launchvid.py pueda volcar los datos de GPS sobre el archivo especificado (sin problemas de compatibilidad).

```

lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609
lat43.522476-lon-5.6112609

```

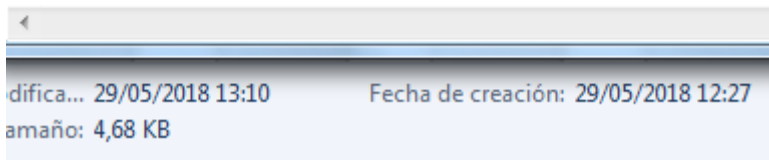


Ilustración 200 Volcado de datos en GPS.txt Fuente: Elaboración propia.

Ahora los datos de GPS se leerán desde automatico.py para así poder geo-etiquetar las fotos térmicas y RGB.

Surgen problemas de compatibilidad con la cámara al usarla en dos terminales multiplexados, la pantalla principal que saca fotos ocupa la cámara (al igual que sucede con la conexión serial tampoco es posible el uso de la cámara por dos procesos diferentes):


```

[remote detached from 3588.lid]
[3588.lid detached.]

<Screen 'lid'>
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
13
la foto es /home/pi/Desktop/Carpet
.522476-lon-5.6112609.jpg y es Tru
14
la foto es /home/pi/Desktop/Carpet
.522476-lon-5.6112609.jpg y es Tru
15
la foto es /home/pi/Desktop/Carpet
.522476-lon-5.6112609.jpg y es Tru
16
la foto es /home/pi/Desktop/Carpet
.522476-lon-5.6112609.jpg y es Tru
17
la foto es /home/pi/Desktop/Carpet
.522476-lon-5.6112609.jpg y es Tru
18

```

Ilustración 201 Script sacando fotos continuamente a la derecha (terminal video), con script de video multiplexado a la izquierda (terminal principal). Fuente: Elaboración propia.

```

<Screen 'lid'>
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
mmal: mmal_vc_port_enable: failed to enable port vc.null_sink:in:0(OPQV): ENOSPC
mmal: mmal_port_enable: failed to enable connected port (vc.null_sink:in:0(OPQV))0x1b0fcc0
mmal: mmal_connection_enable: output port couldn't be enabled
Traceback (most recent call last):
  File "/home/pi/Desktop/CarpetCompPartidaPRUEBA/launchvid.py", line 63, in <module>
    camera = PiCamera()
  File "/usr/lib/python2.7/dist-packages/picamera/camera.py", line 433, in __init__
    self._init_preview()
  File "/usr/lib/python2.7/dist-packages/picamera/camera.py", line 513, in _init_preview
    self, self._camera.outputs[self.CAMERA_PREVIEW_PORT])
  File "/usr/lib/python2.7/dist-packages/picamera/renderers.py", line 558, in __init__
    self.renderer.inputs[0].connect(source).enable()
  File "/usr/lib/python2.7/dist-packages/picamera/mmalobj.py", line 2212, in enable
    prefix="Failed to enable connection")
  File "/usr/lib/python2.7/dist-packages/picamera/exc.py", line 184, in mmal_check
    raise PiCameraMMALError(status, prefix)
picamera.exc.PiCameraMMALError: Failed to enable connection: Out of resources

```

Ilustración 202 Terminal multiplexado visto con sudo screen -x video: El script de video no puede usar la cámara ocupada en tomar fotos. Fuente: Elaboración propia.

El script de video es incapaz de conectar con la cámara en cuanto se sacan más de 30 fotos (cuando se excede el “timeout”) y reporta un error de “out of resources” (la cámara está siendo usada por otro proceso).

De este modo se opta por intentar la segunda opción (reescribir automatico.py).

11.9.2.2.- Reescribiendo el código en automatico.py

A continuación se pretende reescribir el código del script general para añadir el video.



automatico.py

Ilustración 203 Actualización de automatico.py Fuente: Elaboración propia.

```

#importacion de librerias y modulos
import RPi.GPIO as gpio
import time
import sys
import os
from picamera import PiCamera #as camera
#import RPI_to_PIX
from dronekit import connect
import dronekit
from gi.repository import GExiv2
from math import *
from screenutils import list_screens, Screen
import datetime as dt

#inicializacion de pines GPIO

gpio.setmode(gpio.BCM) #designacion de pines GPIO
buzzer_pin=27
gpio.setup(buzzer_pin, gpio.OUT)
gpio.output(buzzer_pin, gpio.LOW)

gpio.setup(05, gpio.IN, pull_up_down=gpio.PUD_UP)
gpio.setup(06, gpio.IN, pull_up_down=gpio.PUD_UP)

gpio.setup(24, gpio.OUT)

```

Ilustración 204 Actualización de automatico.py Fuente: Elaboración propia.

```

gpio.setup(24, gpio.OUT)
disparoVUE= gpio.PWM(24 , 50)
disparoVUE.start(5)
camera=PiCamera()
time.sleep(5)
lat=1
lon=1
alt=""
try:
    #inicializacion de variable contador
    i=0
    #vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)#conexion con controladora
    #camera=PiCamera()
    s=Screen("lid",True)
    print s
    s.send_commands('sudo python /home/pi/Desktop/CarpetaCompartidaPRUEBA/FWM_lidar_rasp.py')
    time.sleep(2)
    vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)
    width=1280
    height=720
    camera.resolution=(width, height)
    def beep (): #en la primera prueba que se hizo beep era free de args
        global buzzer_pin #probar esta linea, no creo que importe
        gpio.output(buzzer_pin,gpio.HIGH)
        time.sleep(2)

```

Ilustración 205 Actualización de automatico.py Fuente: Elaboración propia

```

time.sleep(2)
def capturaVUE ():
    disparoVUE.ChangeDutyCycle(10)
    time.sleep(0.5)
    disparoVUE.ChangeDutyCycle(5)
def altt():
    network = []
    for line in open('informes_Lidar/lidar.txt',"r").readlines():
        network.append(line)
    corte=len(network[len(network)-2])-network[len(network)-2].find("\t")
    return network[len(network)-2][:-corte]
def gps():
    network = []
    for line in open('GPS.txt',"r").readlines():
        network.append(line)
    corte=len(network[len(network)-2])-network[len(network)-2].find("\r\n")
    network[len(network)-2]=network[len(network)-2][:-corte]
    network[len(network)-2]=network[len(network)-2].replace('?', '-')
    return network[len(network)-2]
def listener_gps_raw(self,name,posgps):#funcion oyente para GPS
    #print ("lat:%s\tlon:%s\ttsat.number:%s\ten:%s" %(posgps.lat,posgps.lon,
    global lat,lon#,alt
    lat=posgps.lat
    lon=posgps.lon
        # lat=int(lat)
    lat=1.0*lat #para convertir de int a float
    lat=lat/10000000#esta multiplicado por 10 elevado a 7
    lat=str(lat)

        # lon=int(lon)
    lon=1.0*lon #para convertir de int a float
    lon=lon/10000000#esta multiplicado por 10 elevado a 7
    lon=str(lon)
    alt=posgps.alt

```

Ilustración 206 Actualización de automatico.py Fuente: Elaboración propia

```

def interrupt(channel): #definicion del programa de interrupcion
os.system('sudo python /home/pi/Desktop/CarpetaCompartidaPRUEBA/localizacion_fotos.py')
os.system("find /home/pi/Desktop/CAPTURASDEVUELO -type f -name '*.jpg' -exec cp -rf {} /home/pi/Desktop/CarpetaCompartidaPRUEBA \;")
os.system("find /media/pi/9016-4EF81 -type f -name '*.jpg' -exec cp -rf {} /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR \;") #copi
os.system("find /media/pi/9016-4EF8 -type f -name '*.jpg' -exec cp -rf {} /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR \;") #copi
os.system("find /media/pi/9016-4EF82 -type f -name '*.jpg' -exec cp -rf {} /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR \;") #copi
os.system("find /media/pi/9016-4EF81 -type f -name '*.jpg' -exec rm -rf {} \;") #borra las fotos de la SD de la FLIR
os.system("find /media/pi/9016-4EF8 -type f -name '*.jpg' -exec rm -rf {} \;") #borra las fotos de la SD de la FLIR
os.system("find /media/pi/9016-4EF82 -type f -name '*.jpg' -exec rm -rf {} \;") #borra las fotos de la SD de la FLIR

```

Ilustración 207 Actualización de automatico.py Fuente: Elaboración propia

```

tiempo_i=[]
tiempo_antes_com=[]
tiempo_despues_com=[]
lati=[]
longi=[]
#camera.resolution = (1280, 720)
camera.start_preview()
camera.start_recording('videopi.h264')
#chequeo de la orden de fin de mision
gpio.add_event_detect(05,gpio.FALLING,callback = interrupt) #interrupcion por d
#inicio de bucle infinito de chequeo de señal de disparo
while True:

    tiempo = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S') #cambiar nombre lib
    vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
    time.sleep(0.1)#para darle tiempo al listener
    alt=altt()
    camera.annotate_text = "lat=%s lon=%s time=%s alt=%s" % (lat,lon,tiempo,alt)
    status=gpio.input(06) #chequeo del pin GPIO 06
    if status == False: #si se recibe un puslo se pasa a toma de imagenes
        i=i+1 #actualizacion de contador
        #gps=gps()
        vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
        time.sleep(0.1)#para darle tiempo al listener
        tiempo = dt.datetime.now().strftime('%Y-%m-%d-%H-%M-%S') #llamada al ti

```

Ilustración 208 Actualización de automatico.py Fuente: Elaboración propia

```

if status == False: #si se recibe un puslo se pasa a toma de imagenes
    i=i+1 #actualizacion de contador
    #gps=gps()
    vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
    time.sleep(0.1)#para darle tiempo al listener
    tiempo = dt.datetime.now().strftime('%Y-%m-%d-%H-%M-%S') #llamada al tiempo actual en for
    tiempo_i.append(tiempo)
    #image="/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB/imagenRGB%s %s.jpg"%(tiempo,gps)
    image="/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB/imagenRGB%s_lat%slon%s.jpg"%(tiempo,l
    camera.annotate_text = "lat=%s lon=%s time=%s alt=%s" % (lat,lon,tiempo,alt)
    camera.wait_recording(0.2)
    camera.capture(image,use_video_port=False)
    camera.wait_recording(0.2)#captura de imagen con RaspiCAM, etiquetado con la hora del di
    #metadata = GExiv2.Metadata(image)
    #metadata["Xmp.dc.title"] = "lat:%s\tlon:%s\talt%s\ten %s"%(lat,lon,alt,tiempo)#altera lo
    #metadata.save_file()
    #lat=str(lat)#igual este comandodebe ir antes de image
    camera.annotate_text = "lat=%s lon=%s time=%s alt=%s" % (lat,lon,tiempo,alt)
    image=str(image)
    lati.append(lat)
    longi.append(lon)
    command="sudo exiftool -GPSLongitude="+lon+" -GPSLatitude="+lat+" "+image
    tiempo_antes_com.append(tiempo)
    os.system(command)#check si sabe donde esta el exiftool
    tiempo_despues_com.append(tiempo)
    capturaVUE() #captura de imagen con cámara FLIR VUE PRO, la imagen se guarda en un dir

```

Ilustración 209 Actualización de automatico.py Fuente: Elaboración propia

```

except KeyboardInterrupt:
    print("saliendo del programa a petición")
    print tiempo_i
    print tiempo_antes_com
    print tiempo_despues_com
    print lati
    print longi
    s.kill()
#except:
    #print("ocurrió error o excepción")
finally:
    gpio.cleanup() #limpia todos los puertos

"""prueba el main y si no permite interrupcion por
aviso de error y limpieza de puertos"""

#-----Fin de Programa-----

```

Ilustración 210 Actualización de automatico.py Fuente: Elaboración propia

En este caso es un único proceso (script) el que en el mismo horizonte temporal hace uso tanto de la conexión serial de la controladora como del recurso cámara, por lo que no se producen errores.

Asimismo, se prueba a usar otro puerto de la cámara diferente al del video para la toma de fotos.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python automatico.py
[remote detached from 7981.lid]
[7981.lid detached.]

<Screen 'lid'>
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
    1 image files updated
    1 image files updated
    1 image files updated
    1 image files updated
    1 image files updated
    1 image files updated
    1 image files updated
^Csaliendo del programa a petición
['2018-05-29-16-07-00', '2018-05-29-16-07-08', '2018-05-29-16-07-16', '2018-05-29-16-07-23', '2018-05-29-16-07-30', '2018-05-29-16-07-41']
['2018-05-29-16-07-00', '2018-05-29-16-07-08', '2018-05-29-16-07-16', '2018-05-29-16-07-23', '2018-05-29-16-07-30', '2018-05-29-16-07-41']
['2018-05-29-16-07-00', '2018-05-29-16-07-08', '2018-05-29-16-07-16', '2018-05-29-16-07-23', '2018-05-29-16-07-30', '2018-05-29-16-07-41']
['43.522476', '43.522476', '43.522476', '43.522476', '43.522476', '43.522476']
['-5.6112609', '-5.6112609', '-5.6112609', '-5.6112609', '-5.6112609', '-5.6112609']

```

Ilustración 211 Ejecución de automatico.py Fuente: Elaboración propia.

En la imagen anterior se ven las marcas temporales de cada una de las fotos RGB tomadas con su información GPS.

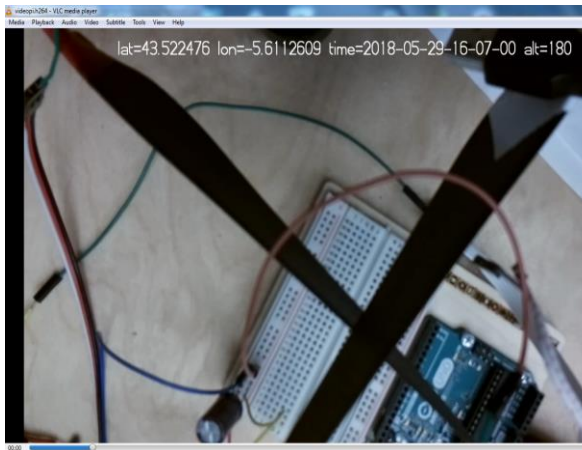


Ilustración 212 Video generado durante la ejecución de automatico.py (principio de video).
Fuente: Elaboración propia.



Ilustración 213 Video generado durante la ejecución de automatico.py (final de video). Fuente: Elaboración propia.

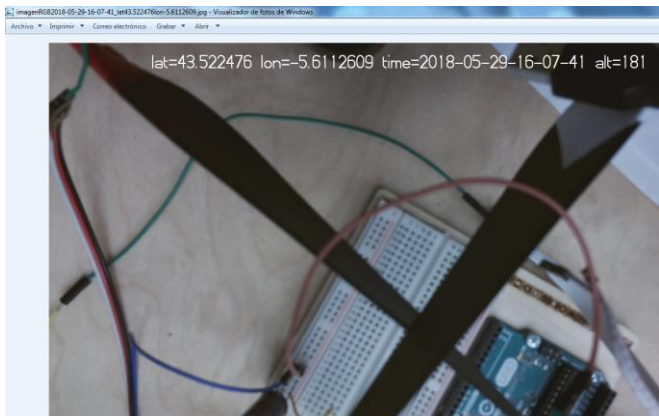


Ilustración 214 Foto tomada durante la ejecución de automatico.py Fuente: Elaboración propia.

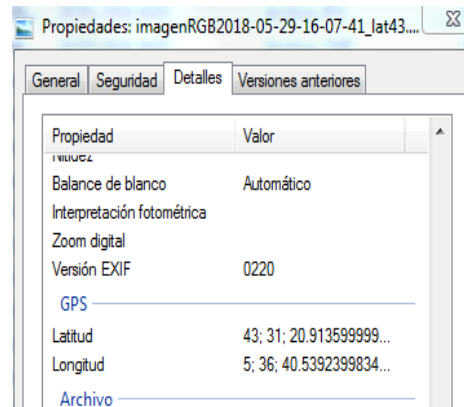


Ilustración 215 Metadatos de una de las fotos tomadas durante la ejecución de automatico.py Fuente: Elaboración propia.

12. Datos GNSS

Se pretende obtener datos de interés provenientes de la controladora de vuelo que puedan ser visualizados por el usuario a través de la Raspberry pi.

Entre otros se pretende conocer la localización GPS (latitud y longitud) a cada instante de tiempo, para de este modo géo-referenciar las fotos RGB en cada una de las misiones.

Para ello se usará un programa python llamado dronekit.py, el cual se descargará en la Raspberry usando el comando por consola:

- `pip install dronekit`

Primero será necesario conectarse desde la Raspberry a la controladora, siguiendo una secuencia de pasos:

- Conexión hardware; de modo idéntico a como se realiza en el apartado relativo al envío de misión:

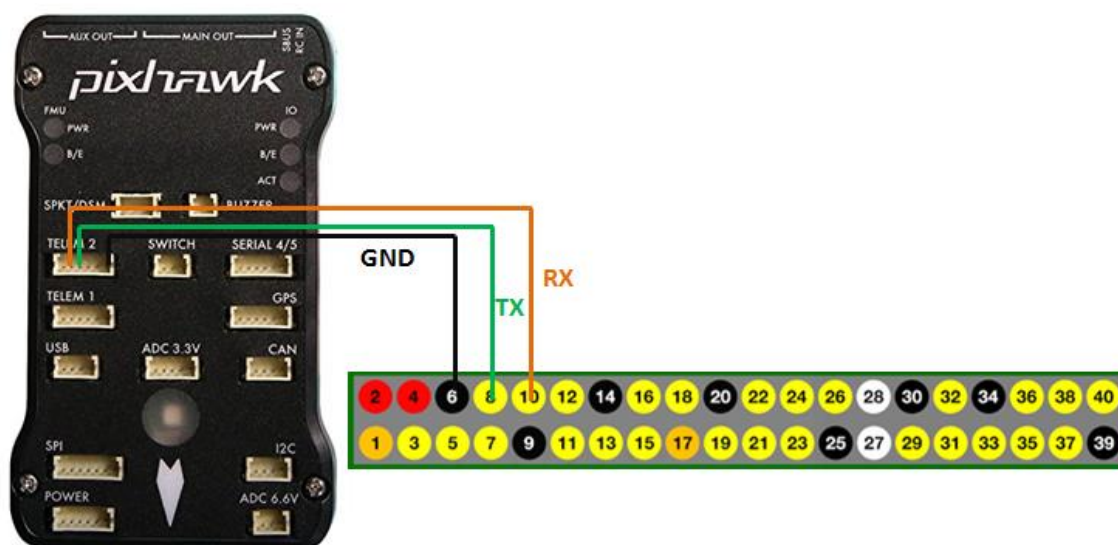


Ilustración 216 Conexión hardware necesaria entre al controladora y la Raspberry pi. Fuente: Elaboración propia.

- Elaboración del software necesario:

Realizando para ello un script que primero realizará la conexión propiamente dicha, siguiendo el siguiente ejemplo:

Linux computer connected to the vehicle via Serial port
(RaspberryPi example)

/dev/ttyAMA0 (also set baud=57600)

Ilustración 217 Conectarse a vehículo con dronekit. Fuente:
http://python.dronekit.io/guide/connecting_vehicle.html

```
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python dronekit_test.py
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
conectado a: <dronekit.Vehicle object at 0x753fdcf0>
```

Ilustración 218 Conexión satisfactoria. Fuente: Elaboración propia.

Y, posteriormente llamará a las “defs” (por medio de “listeners”) que se consideren necesarias dentro del “main” de dronekit.py. De forma similar al siguiente ejemplo:

```
class Rangefinder(object):
    """
    Rangefinder readings.

    An object of this type is returned by :py:attr:`Vehicle.rangefinder`.

    :param distance: Distance (metres). ``None`` if the vehicle doesn't have a
    :param voltage: Voltage (volts). ``None`` if the vehicle doesn't have a
    """

    def __init__(self, distance, voltage):
        self.distance = distance
        self.voltage = voltage

    def __str__(self):
        return "Rangefinder: distance={}, voltage={}".format(self.distance, self.voltage)
```

```
#Create a message listener using the decorator.
@vehicle.on_message('RANGEFINDER')
def listener(self, name, message):
    print message
```

Ilustración 219 Listener para "Rangefinder" tras importar dronekit.py como librería. Fuente:
http://python.dronekit.io/guide/mavlink_messages.html

Ilustración 220 Class “Rangefinder” dentro de dronekit.py.

Fuente:https://github.com/dronekit/dronekit-python/blob/master/dronekit/__init__.py

De este modo cada vez que la controladora envíe un comando determinado se imprimirá un mensaje, pudiendo “escuchar” las comunicaciones de la controladora.

12.1.- PRIMERA PRUEBA: CAPTACIÓN DE TODOS LOS MENSAJES PROVENIENTES DE LA CONTROLADORA DE VUELO.

En un primer acercamiento se opta por la opción de leer todos los mensajes provenientes de la controladora, al no saber la referenciación propia de cada tipo de mensaje.

De este modo se elabora el siguiente código:

Que al ejecutarse en la Raspberry permite ver por pantalla todos los mensajes MavLink que transmite la controladora de vuelo, del modo siguiente:

```
import dronekit
from dronekit import connect
import time
import os
import sys

try:
    #Linux connected to the vehicle via serial port
    vehicle = connect('/dev/ttyAMA0',baud=57600)
    print ("conectado a: %s" %vehicle)

    #callback para captar todos los mensajes
    @vehicle.on_message('*')
    def listener(self, name, message):
        print 'message: %s' %message
```

Ilustración 221 dronekit_test.py. Fuente:
Elaboración propia.

```
param_type : 9, param_count : 773, param_index : 256)
message: PARAM_VALUE {param_id : ATC_RAT_PIT_IMAX, param_value : 0
: 9, param_count : 773, param_index : 257)
message: PARAM_VALUE {param_id : ATC_RAT_PIT_FILT, param_value : 2
e : 9, param_count : 773, param_index : 258}
message: PARAM_VALUE {param_id : ATC_RAT_PIT_FF, param_value : 0.0
9, param_count : 773, param_index : 259}
message: RAW_IMU {time_usec : 1401892433, xacc : -38, yacc : -7, z
yro : 5, ygyro : -2, zgyro : 4, xmag : -27, ymag : 52, zmag : 308}
message: SCALED_IMU2 {time_boot_ms : 1401892, xacc : -37, yacc : -
8, xgyro : -69, ygyro : -23, zgyro : -7, xmag : -145, ymag : 116,
message: SCALED_PRESSURE {time_boot_ms : 1401892, press_abs : 1020
ss_diff : -0.0920312479138, temperature : 4024}
message: SYS_STATUS {onboard_control_sensors_present : 56687663, on
_sensors_enabled : 39894063, onboard_control_sensors_health : 5668
34, voltage_battery : 12220, current_battery : 62, battery_remaini
rate_comm : 0, errors_comm : 0, errors_count1 : 0, errors_count2 :
nt3 : 0, errors_count4 : 0}
message: POWER_STATUS {Vcc : 5161, Vservo : 43, flags : 1}
message: MEMINFO {brkval : 0, freemem : 43264}
message: MISSION_CURRENT {seq : 0}
message: GPS_RAW_INT {time_usec : 0, fix_type : 1, lat : 0, lon : 0
0, eph : 9999, epv : 9999, vel : 0, cog : 0, satellites_visible : 0}
message: NAV_CONTROLLER_OUTPUT {nav_roll : 0.000316879741149, nav_p
69842941e-05, nav_bearing : -112, target_bearing : 0, wp_dist : 0,
.0, aspd_error : 0.0, xtrack_error : 0.0}
message: GLOBAL_POSITION_INT {time_boot_ms : 1401892, lat : 0, lon
0, relative_alt : 750, vx : 1, vy : -1, vz : 0, hdg : 24758}
message: SERVO_OUTPUT_RAW {time_usec : 1401892890, port : 0, servo
ervo2_raw : 982, servo3_raw : 982, servo4_raw : 982, servo5_raw : 0
: 0, servo7_raw : 0, servo8_raw : 0}
message: RC_CHANNELS_RAW {time_boot_ms : 1401892, port : 0, chan1_
_raw : 0, chan3_raw : 0, chan4_raw : 0, chan5_raw : 0, chan6_raw :
: 0, chan8_raw : 0, rssi : 0}
message: RC_CHANNELS {time_boot_ms : 1401892, chancount : 0, chan1
2_raw : 0, Chan3_raw : 0, Chan4_raw : 0, chan5_raw : 0, chan6_raw
: 0, chan8_raw : 0, chan9_raw : 0, chan10_raw : 0, chan11_raw : 0
0, chan13_raw : 0, chan14_raw : 0, chan15_raw : 0, chan16_raw : 0
0, chan18_raw : 0, rssi : 0}
message: ATTITUDE {time_boot_ms : 1401893, roll : 0.00471319118515
0145776942372, yaw : -1.96227133274, rollspeed : 0.00513484422117,
-0.00262367795222, yawspeed : 0.00498730316758}
```

Ilustración 222 Output de dronekit_test. Fuente:
Elaboración propia.

De toda esta información se deben seleccionar algunos de los datos más importantes, como por ejemplo:

```
message: GLOBAL_POSITION_INT {time_boot_ms : 1404893, lat : 0, lon : 0, alt : 780, relative_alt : 780, vx : 1, vy : -1, vz : 0, hdg : 24755}
```

Ilustración 223 GLOBAL_POSITION_INT con dronekit_test.py. Fuente: Elaboración propia.

```
message: GPS_RAW_INT {time_usec : 0, fix_type : 1, lat : 0, lon : 0, alt : -17000, eph : 9999, epv : 9999, vel : 0, cog : 0, satellites_visible : 0}
```

Ilustración 224 GPS_RAW_INT con dronekit_test.py. Fuente: Elaboración propia.

```
message: ATTITUDE {time_boot_ms : 1402893, roll : 0.00469557149336, pitch : -0.00154174864292, yaw : -1.96233260632, rollspeed : 0.00429003220052, pitchspeed : -0.00293366750702, yawspeed : 0.00554769160226}
```

Ilustración 225 ATTITUDE con dronekit_test.py. Fuente: Elaboración propia.

```
message: SYSTEM_TIME {time_unix_usec : 315966180705000, time_boot_ms : 1398893}
```

Ilustración 226 SYSTEM_TIME con dronekit_test.py. Fuente: Elaboración propia.

```
message: RAW_IMU {time_usec : 1404892904, xacc : -39, yacc : -8, zacc : -997, xgyro : 5, ygyro : -2, zgyro : 5, xmag : -28, ymag : 52, zmag : 309}
```

Ilustración 227 RAW_IMU con dronekit_test.py. Fuente: Elaboración propia.

```
message: MISSION_CURRENT {seq : 0}
```

Ilustración 228 Fuente: Elaboración propia.

12.2.- OBTENCIÓN DE SÓLO UNO O VARIOS TIPOS DE MENSAJES

Para ver solo uno o varios tipos de mensajes se deben crear diferentes “callback”, como se muestra en la siguiente imagen:

```

import dronekit
from dronekit import connect
import time

try:
    #Linux connected to the vehicle via serial port RPI example
    vehicle = connect('/dev/ttyAMA0',baud=57600)
    print ("conectado a: %s" %vehicle)
    """
    #callback para captar todos los mensajes
    @vehicle.on_message('*')
    def listener(self, name, message):
        print 'message: %s' %message
    """
    def get_time():
        date=time.strftime("%H-%M-%S")
        return date
    def wait():
        time.sleep(1)
        return
    """#ruta para abrir un fichero
    def open_file(a):
        ruta='/home/pi/Desktop/CarpetaCompartidaPRUEBA/datos_controladora/'+a
        os.system('mkdir(ruta)')
        ru=ruta+'.txt'
        f=open(ru,"w")#apertura fichero
    """
    |
    def receivedPos(self,name,posMsg):#funcion
        print posMsg
    def listener_gps_raw(self,name,posgps):#funcion
        print ("lat:%s\tlon:%s\tlat.number:%s\tten:%s" %(posgps.lat,posgps.lon,posgps.satellites_visible,get_time()))
        wait()
    def listener_attitu(self,name,att):
        print att.roll
    def listener_wp(self,name,wp):
        print wp
    while True:
        #esto es un callback a una funcion(cuando se detecte el comando GLOBAL)
        #vehicle.add_message_listener('GLOBAL_POSITION_INT',receivedPos)
        vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
        #vehicle.add_message_listener('ATTITUDE',listener_attitu)
        vehicle.add_message_listener('WAYPOINT_COUNT',listener_wp)
        #print "Param: %s" % vehicle.parameters['THR_MIN']
        #while not
except KeyboardInterrupt:
    print("Saliendo a petición del usuario")

```

Ilustración 229 Dronekit_test.py Fuente: Elaboración propia.

Un “callback” como por ejemplo:

- Vehicle.add_message_listener(‘GPS_RAW_INT’,listener_gps_raw)

Esta línea que se repite en un bucle infinito (While True) ejecuta la definición especificada (en este caso listener_gps_raw) cada vez que capta el mensaje Mavlink GPS_RAW_INT de la controladora.

Es decir se “escuchan” los datos que la controladora transfiere a los dispositivos conectados a ella y se muestran por pantalla. (cada “callback”, como el comentado anteriormente, hace objetivo a un mensaje diferente que se imprime por pantalla a través de una “def” ó “listener”)

GPS_RAW_INT (#24)

The global position, as returned by the Global Positioning System (GPS). This is NOT the global position estimate of the system, but rather a RAW sensor value. See message GLOBAL_POSITION for the global position estimate.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot)
fix_type	uint8_t	See the GPS_FIX_TYPE enum.
lat	int32_t	Latitude (WGS84, EGM96 ellipsoid), in degrees * 1E7
lon	int32_t	Longitude (WGS84, EGM96 ellipsoid), in degrees * 1E7
alt	int32_t	Altitude (AMSL, NOT WGS84), in meters * 1000 (positive for up). Note that virtually all GPS modules provide the AMSL altitude in addition to the WGS84 altitude.
eph	uint16_t	GPS HDOP horizontal dilution of position (unitless). If unknown, set to: UINT16_MAX
epv	uint16_t	GPS VDOP vertical dilution of position (unitless). If unknown, set to: UINT16_MAX
vel	uint16_t	GPS ground speed (m/s * 100). If unknown, set to: UINT16_MAX
cog	uint16_t	Course over ground (NOT heading, but direction of movement) in degrees * 100, 0.0..359.99 degrees. If unknown, set to: UINT16_MAX
satellites_visible	uint8_t	Number of satellites visible. If unknown, set to 255
alt_ellipsoid	int32_t	Altitude (above WGS84, EGM96 ellipsoid), in meters * 1000 (positive for up).
h_acc	uint32_t	Position uncertainty in meters * 1000 (positive for up).
v_acc	uint32_t	Altitude uncertainty in meters * 1000 (positive for up).
vel_acc	uint32_t	Speed uncertainty in meters * 1000 (positive for up).
hdg_acc	uint32_t	Heading / track uncertainty in degrees * 1e5.

Ilustración 230 Ejemplo de mensaje Mavlink. Fuente: <http://mavlink.org/messages/common>

```
pi@raspi-locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python dronekit_test.py
>>> ARM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
conectado a: <dronekit.Vehicle object at 0x754a5c50>
GLOBAL_POSITION_INT (time_boot_ms : 3076324, lat : 0, lon : 0, alt : 1540, relative_alt : 1540, vx : 0, vy : 0, vz : 1, hdg : 24645)
GLOBAL_POSITION_INT (time_boot_ms : 3077325, lat : 0, lon : 0, alt : 1560, relative_alt : 1560, vx : 0, vy : 0, vz : 1, hdg : 24644)
GLOBAL_POSITION_INT (time_boot_ms : 3078325, lat : 0, lon : 0, alt : 1590, relative_alt : 1590, vx : 0, vy : 0, vz : 1, hdg : 24645)
GLOBAL_POSITION_INT (time_boot_ms : 3079326, lat : 0, lon : 0, alt : 1580, relative_alt : 1580, vx : 0, vy : 0, vz : 1, hdg : 24645)
GLOBAL_POSITION_INT (time_boot_ms : 3080326, lat : 0, lon : 0, alt : 1560, relative_alt : 1560, vx : 0, vy : 0, vz : 0, hdg : 24644)
GLOBAL_POSITION_INT (time_boot_ms : 3081326, lat : 0, lon : 0, alt : 1550, relative_alt : 1550, vx : 0, vy : 0, vz : 0, hdg : 24645)
GLOBAL_POSITION_INT (time_boot_ms : 3082326, lat : 0, lon : 0, alt : 1540, relative_alt : 1540, vx : 0, vy : 0, vz : 0, hdg : 24644)
GLOBAL_POSITION_INT (time_boot_ms : 3083326, lat : 0, lon : 0, alt : 1530, relative_alt : 1530, vx : 0, vy : 0, vz : 0, hdg : 24643)
GLOBAL_POSITION_INT (time_boot_ms : 3084327, lat : 0, lon : 0, alt : 1510, relative_alt : 1510, vx : 0, vy : 0, vz : 0, hdg : 24641)
```

Ilustración 231 Output dronekit_test.py para un solo oyente. Fuente: Elaboración propia.

En el caso de que solo se quisiera obtener un único valor y no durante toda la misión se seguirá el siguiente ejemplo:

```
# Get Vehicle Home Location - will be `None` until first set by autopilot
while not vehicle.home_location:
    cmds = vehicle.commands
    cmds.download()
    cmds.wait_ready()
    if not vehicle.home_location:
        print " Waiting for home location ..."

# We have a home location.
print "\n Home location: %s" % vehicle.home_location
```

Ilustración 232 Código ejemplo para solo obtener un valor. Fuente: http://python.dronekit.io/guide/vehicle_state_and_parameters.html

12.3.- OBTENCIÓN DE UN ÚNICO DATO DENTRO DE UN TIPO DE MENSAJE

Puede ser que un único tipo de mensaje MAVLINK, como por ejemplo ATTITUDE ó GPS_RAW_INT, aporte cierta información que no necesitemos procesar, para ello se

cuenta con la opción de mostrar un único dato ó atributo dentro de una clase ó tipo de mensaje.

Así, por ejemplo, se puede sacar el roll (un tipo de rotación de la aeronave) que es un dato (atributo) dentro de la clase ATTITUDE:

ATTITUDE (#30)

The attitude in the aeronautical frame (right-handed, Z-down, X-front, Y-right).

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot)
roll	float	Roll angle (rad, -pi..+pi)
pitch	float	Pitch angle (rad, -pi..+pi)
yaw	float	Yaw angle (rad, -pi..+pi)
rollspeed	float	Roll angular speed (rad/s)
pitchspeed	float	Pitch angular speed (rad/s)
yawspeed	float	Yaw angular speed (rad/s)

Ilustración 233 Parámetros de aeronave en comando Mavlink (roll dentro de ATTITUDE). Fuente: <http://mavlink.org/messages/common>

```
def listener_attitu(self,name,att):
    print att.roll
def listener_wp(self,name,wp):
    print wp

while True:
    #esto es un callback a una funcion(cuando se detecte el comando
    #vehicle.add_message_listener('GLOBAL_POSITION_INT',receivedPos)
    #vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
    vehicle.add_message_listener('ATTITUDE',listener_attitu)
    vehicle.add_message_listener('WAYPOINT_COUNT',listener_wp)
    #print "Param: %s" % vehicle.parameters['THR_MIN']
    #while not
```

```
pi@raspi_locis:~/Desktop/CarpetacompartidaPRUEBA $ sudo python dronekit_
>>> ARM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
conectado a: <dronekit.Vehicle object at 0x75473d50>
>>> Link timeout, no heartbeat in last 5 seconds
>>> ...link restored.
0.00200571957976
0.00197664368898
0.00207009026781
0.00214117805125
```

Ilustración 234 Código para obtener el "roll" de la aeronave. Fuente: Elaboración propia.

Ilustración 235 Output roll. Fuente: Elaboración propia.

De este modo se intentará aislar los datos más relevantes, como por ejemplo la posición del dron (definida como latitud y longitud) determinada por GPS satelital.

12.4.- SEGUNDA PRUEBA: OBTENCIÓN DE POSICIONAMIENTO GPS.

```
pi@raspi_locis:~/Desktop/CarpetasCompartidasPRUEBA $ sudo python dronekit_test.py
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
conectado a: <dronekit.Vehicle object at 0x75470dd0>
lat:0 lon:0 sat.number:0 en:12-55-06
lat:0 lon:0 sat.number:0 en:12-55-38
lat:0 lon:0 sat.number:0 en:12-55-43
lat:0 lon:0 sat.number:0 en:12-55-44
lat:0 lon:0 sat.number:0 en:12-55-47
lat:0 lon:0 sat.number:0 en:12-55-49
lat:0 lon:0 sat.number:0 en:12-55-51
lat:0 lon:0 sat.number:0 en:12-56-26
lat:0 lon:0 sat.number:0 en:12-56-28
lat:0 lon:0 sat.number:0 en:12-56-31
```

Ilustración 236 Output longitud, latitud y número de satélites dentro de la oficina. Fuente: Elaboración propia.

Dentro de la oficina nunca se consiguieron captar satélites, ya fuese con el GPS del dron o con el GPS de mano (para más información consultar el apartado referente a Lidar), por lo que para continuar esta prueba es necesario trasladarse al exterior.

Para ello se mueve el dron a la puerta de la oficina, de modo que la Raspberry es capaz de:

- Seguir conectada al wifi de la oficina (manteniendo su dirección IP)
- Captar un reducido número de satélites.

La interfaz de la consola se logra mediante una conexión ssh con un móvil, de la forma que se ve en la imagen:

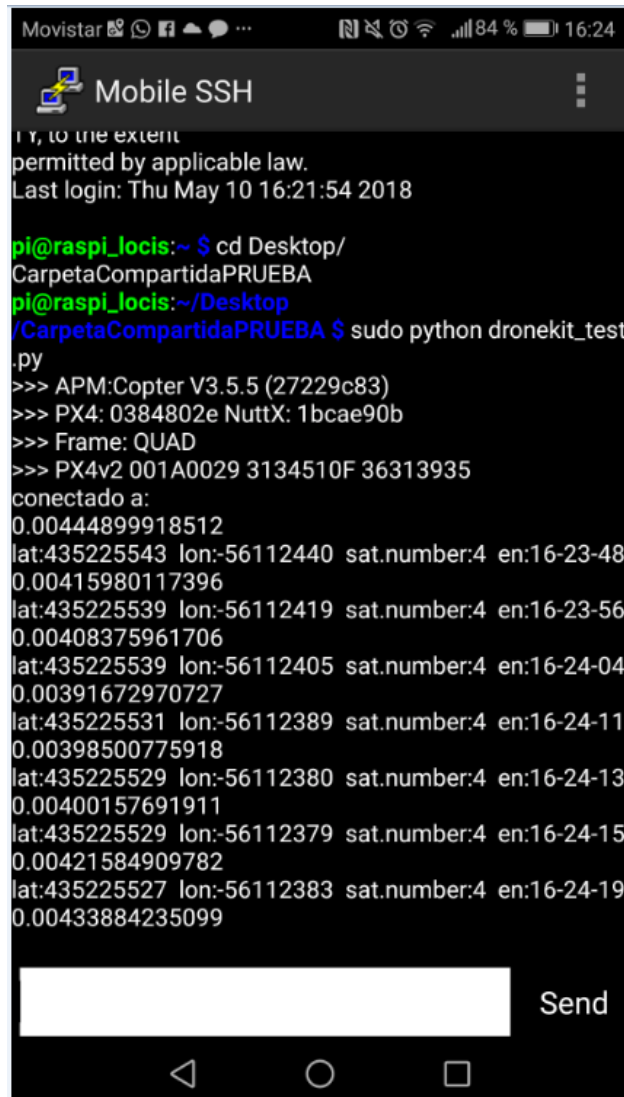


Ilustración 237 Repetición de la prueba fuera de la oficina. Fuente: Elaboración propia.

Como se puede ver en la imagen siguiente los datos de longitud y latitud vienen expresados como potencia de 7:

GLOBAL_POSITION_INT (#33)

The filtered global position (e.g. fused GPS and accelerometers). The position is in GPS-frame (right-handed, Z-up). It is sufficient.

Field Name	Type	
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot)
lat	int32_t	Latitude, expressed as degrees * 1E7
lon	int32_t	Longitude, expressed as degrees * 1E7
alt	int32_t	Altitude in meters, expressed as * 1000 (in AMSL as well)
relative_alt	int32_t	Altitude above ground in meters, expressed as * 1000
vx	int16_t	Ground X Speed (Latitude, positive north), expressed as m/s * 100
vy	int16_t	Ground Y Speed (Longitude, positive east), expressed as m/s * 100
vz	int16_t	Ground Z Speed (Altitude, positive down), expressed as m/s * 100
hdg	uint16_t	Vehicle heading (yaw angle) in degrees * 100

Ilustración 238 mensaje Mavlink. Fuente: <http://mavlink.org/messages/common>

Por lo que la posición sería lat: 43,52 y la long:-56,11 aproximadamente (datos que coinciden con la ubicación del parque tecnológico según “google maps”).

De este modo se puede concluir la prueba como exitosa.

12.5.- OUTPUT DE DATOS DE LA CONTROLADORA COMO .TXT

Por último se pretende generar ficheros que contengan datos de importancia de la controladora, como por ejemplo la geo-localización.

Para de este modo poder disponer de información GPS con referenciación horaria, del modo siguiente:

```
import dronekit
from dronekit import connect
import time
import os
import sys

try:
    #Linux conected to the vehicle via serial port RPI example
    vehicle = connect('/dev/ttyAMA0',baud=57600)
    print ("conectado a: %s" %vehicle)

    """
    #callback para captar todos los mensajes
    @vehicle.on_message('*')
    def listener(self, name, message):
        print 'message: %s' %message
    """

    def get_time():
        date=time.strftime("%H-%M-%S")
        return date

    def wait():
        time.sleep(1)
        return

    def open_file(a):
        ruta="/home/pi/Desktop/CarpetaCompartidaPRUEBA/datos_controladora/"
        dirpath=os.path.join(ruta,a)
        print dirpath
        existencia= os.path.exists(dirpath)
        print existencia
        if existencia==False:
            os.mkdir(dirpath)
            return
        ru=ruta+''+a+''+get_time()+'.txt'
        f=open(ru,"w")#apertura fichero
        return f
    g=open_file("gps_raw")#no se pueden abrir en un bucle por que se pisan
    r=open_file("roll")
```

Ilustración 239 Versión actualizada de dronekit_test.py. Fuente: Elaboración propia.


```

#para abrir file poner open_file("test")
def receivedPos(self,name,posMsg):#funcion
    print posMsg
def listener_gps_raw(self,name,posgps):#funcion
    print ("lat:%s\tlon:%s\tlat.number:%s\tten:%s" %(posgps.lat,posgps.lon,posgps.satellites_visible,get_time()))
    global g
    g.write("lat:%s\tlon:%s\tlat.number:%s\tten:%s\r\n" %(posgps.lat,posgps.lon,posgps.satellites_visible,get_time()))
    wait()
def listener_attitu(self,name,att):
    print att.roll
    global r
    r.write("roll:%s\tten:%s\r\n" %(att.roll,get_time()))
def listener_wp(self,name,wp):
    print wp
while True:
    #esto es un callback a una funcion(cuando se detecte el comando GLOBAL)
    #vehicle.add_message_listener('GLOBAL_POSITION_INT',receivedPos)
    vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
    vehicle.add_message_listener('ATTITUDE',listener_attitu)
    vehicle.add_message_listener('WAYPOINT_COUNT',listener_wp)
except KeyboardInterrupt:
    print("Saliendo a peticion del usuario")

```

Ilustración 240 Versión actualizada de dronekit_test.py. Fuente: Elaboración propia.

The screenshot shows a terminal window on the left and a Notepad application on the right. The terminal displays the execution of dronekit_test.py, showing the drone's connection status and the output of the listener functions. The Notepad application shows a log of GPS data with columns for latitude, longitude, satellite count, and time.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python dronekit_test.py
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
conectado a: <dronekit.Vehicle object at 0x754eae50>
/home/pi/Desktop/CarpetaCompartidaPRUEBA/datos_controladora/gps_raw
True
/home/pi/Desktop/CarpetaCompartidaPRUEBA/datos_controladora/roll
True
lat:435225389 lon:-56112333 sat.number:0 en:14-58-06
0.0035898941569
lat:435225389 lon:-56112333 sat.number:0 en:14-58-13
0.0036829826422
lat:435225389 lon:-56112333 sat.number:0 en:14-58-19
0.0037981919013
lat:435225389 lon:-56112333 sat.number:0 en:14-58-20

```

Archivo	Edición	Formato	Ver	Ayuda
lat:435225389	lon:-56112333	sat.number:0	en:14-58-06	
lat:435225389	lon:-56112333	sat.number:0	en:14-58-13	
lat:435225389	lon:-56112333	sat.number:0	en:14-58-19	
lat:435225389	lon:-56112333	sat.number:0	en:14-58-20	
lat:435225389	lon:-56112333	sat.number:0	en:14-58-22	
lat:435225389	lon:-56112333	sat.number:0	en:14-58-23	
lat:435225389	lon:-56112333	sat.number:0	en:14-58-24	

Ilustración 241 Ejecución del programa dentro de la oficina. Fuente: Elaboración propia.

Ilustración 242 Información de la controladora con referenciación horaria, visto desde un .txt en la estación de tierra. Fuente: Elaboración propia.



dronekit_test.py

12.1 Dronekit_test.py

12.6.- OUTPUT COMO METADATOS DE IMAGEN

Los metadatos corresponden a datos que referencian a otros datos. Por ejemplo pulsando en “propiedades” de un archivo .jpg podemos acceder a los metadatos de la citada fotografía.

Se pretende sobre-escribir los metadatos de las imágenes RGB para añadir información referente a la posición de la imagen, para ello se usará un script como el que se muestra a continuación:

```

from gi.repository import GExiv2

# Obtener metadatos actuales de la imagen
a=40
b=1
image = "/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB/diatoms%s%s.jpg"%(a,b)
metadata = GExiv2.Metadata(image)
#manipulacion de metadatos
metadata["Xmp.dc.title"] = "Jorge%s%s"%(a,b)
metadata.save_file()
# Imprimir todos los metadatos
for key in metadata:
    print("{}: {}".format(key, metadata[key]))

```

Ilustración 243 Script correspondiente a metadatos.py, para manipulación de metadatos. Fuente: Elaboración propia.

Dicho script primero importa una librería que necesita primero una serie de instalaciones por consola, de la forma que se muestra en las imágenes siguientes:

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo apt-get install python-gi
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  libexiv2-13 libexiv2-2
Paquetes sugeridos:
  exiv2
Se instalarán los siguientes paquetes NUEVOS:
  gir1.2-gexiv2-0.10 libexiv2-13 libgexiv2-2
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 5 no actual
Se necesita descargar 678 kB de archivos.
Se utilizarán 2.515 kB de espacio de disco adicional después de esta
¿Desea continuar? [S/n] s
Des:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main libexi
0.24-4.1 [628 kB]
Des:2 http://mirrordirector.raspbian.org/raspbian/ jessie/main libgex
0.10.2-2 [39,9 kB]
Des:3 http://mirrordirector.raspbian.org/raspbian/ jessie/main gir1.2
0 armhf 0.10.2-2 [10,1 kB]
Descargados 678 kB en 6s (98,9 kB/s)
Seleccionando el paquete libexiv2-13:armhf previamente no seleccionad
(Leyendo la base de datos ... 138342 ficheros o directorios instalad
te.)
Preparando para desempaquetar ../libexiv2-13_0.24-4.1_armhf.deb ...
Desempaquetando libexiv2-13:armhf (0.24-4.1) ...
Seleccionando el paquete libgexiv2-2:armhf previamente no seleccionad
Preparando para desempaquetar ../libgexiv2-2_0.10.2-2_armhf.deb ...
Desempaquetando libgexiv2-2:armhf (0.10.2-2) ...
Seleccionando el paquete gir1.2-gexiv2-0.10:armhf previamente no sele
Preparando para desempaquetar ../gir1.2-gexiv2-0.10_0.10.2-2_armhf.d
Desempaquetando gir1.2-gexiv2-0.10:armhf (0.10.2-2) ...
Configurando libexiv2-13:armhf (0.24-4.1) ...
Configurando libgexiv2-2:armhf (0.10.2-2) ...
Configurando gir1.2-gexiv2-0.10:armhf (0.10.2-2) ...
Procesando disparadores para libc-bin (2.19-18+deb8u10) ...
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $

```

Ilustración 244 Instalación correspondiente para poder importar “girepository”. Fuente: Elaboración propia.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo apt-get install python-gi-cairo
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
python-gi ya está en su versión más reciente.
fijado python-gi como instalado manualmente.
python3-gi ya está en su versión más reciente.
fijado python3-gi como instalado manualmente.
gir1.2-gtk-3.0 ya está en su versión más reciente.
fijado gir1.2-gtk-3.0 como instalado manualmente.
Se instalarán los siguientes paquetes extras:
  python3-cairo
Se instalarán los siguientes paquetes NUEVOS:
  python-gi-cairo python3-cairo python3-gi-cairo
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 5 no
Se necesita descargar 647 kB de archivos.
Se utilizarán 805 kB de espacio de disco adicional después de
¿Desea continuar? [S/n] s
Des:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main
rmhf 3.14.0-1 [310 kB]
Des:2 http://mirrordirector.raspbian.org/raspbian/ jessie/main
hf 1.10.0+dfsg-4 [26,7 kB]
Des:3 http://mirrordirector.raspbian.org/raspbian/ jessie/main
armhf 3.14.0-1 [310 kB]
Descargados 647 kB en 2s (227 kB/s)
Seleccionando el paquete python-gi-cairo previamente no selecc
(Leyendo la base de datos ... 138361 ficheros o directorios inf
te.)
Preparando para desempaquetar ../python-gi-cairo_3.14.0-1_arm
Desempaquetando python-gi-cairo (3.14.0-1) ...
Seleccionando el paquete python3-cairo previamente no seleccionad
Preparando para desempaquetar ../python3-cairo_1.10.0+dfsg-4_
Desempaquetando python3-cairo (1.10.0+dfsg-4) ...
Seleccionando el paquete python3-gi-cairo previamente no selecc
Preparando para desempaquetar ../python3-gi-cairo_3.14.0-1_ar
Desempaquetando python3-gi-cairo (3.14.0-1) ...
Configurando python-gi-cairo (3.14.0-1) ...
Configurando python3-cairo (1.10.0+dfsg-4) ...
Configurando python3-gi-cairo (3.14.0-1) ...
Procesando disparadores para libc-bin (2.19-18+deb8u10) ...
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $

```

Ilustración 245 Instalación correspondiente para poder importar “girepository”. Fuente: Elaboración propia.

Posteriormente se cambia el título correspondiente a los metadatos de una imagen dada por un texto de prueba, produciéndose el output deseado:

```

pi@raspi_locis:~/Desktop/CarpetasCompartidasPRUEBA $ sudo python metada
Exif.Image.DateTime: 2018:05:11 13:23:42
Exif.Image.ExifTag: 192
Exif.Image.ImageLength: 1080
Exif.Image.ImageWidth: 1920
Exif.Image.Make: RaspberryPi
Exif.Image.Model: RP_imx219
Exif.Image.ResolutionUnit: 2
Exif.Image.XResolution: 72/1
Exif.Image.YCbCrPositioning: 1
Exif.Image.YResolution: 72/1
Exif.Iop.InteroperabilityIndex: R98
Exif.Photo.ApertureValue: 20000/10000
Exif.Photo.BrightnessValue: 218/100
Exif.Photo.ColorSpace: 1
Exif.Photo.ComponentsConfiguration: 1 2 3 0
Exif.Photo.DateTimeDigitized: 2018:05:11 13:23:42
Exif.Photo.DateTimeOriginal: 2018:05:11 13:23:42
Exif.Photo.ExifVersion: 48 50 50 48
Exif.Photo.ExposureMode: 0
Exif.Photo.ExposureProgram: 3
Exif.Photo.ExposureTime: 32997/1000000
Exif.Photo.FNumber: 20000/10000
Exif.Photo.Flash: 0
Exif.Photo.FlashpixVersion: 48 49 48 48
Exif.Photo.FocalLength: 30390/10000
Exif.Photo.ISOSpeedRatings: 64
Exif.Photo.InteroperabilityTag: 900

```

```

Exif.Photo.MaxApertureValue: 20000/10000
Exif.Photo.MeteringMode: 2
Exif.Photo.PixelXDimension: 1920
Exif.Photo.PixelYDimension: 1080
Exif.Photo.ShutterSpeedValue: 4921521/1000000
Exif.Photo.WhiteBalance: 0
Exif.Thumbnail.Compression: 6
Exif.Thumbnail.ImageLength: 48
Exif.Thumbnail.ImageWidth: 64
Exif.Thumbnail.JPEGInterchangeFormat: 1036
Exif.Thumbnail.JPEGInterchangeFormatLength: 245
Exif.Thumbnail.ResolutionUnit: 2
Exif.Thumbnail.XResolution: 72/1
Exif.Thumbnail.YResolution: 72/1
Xmp.dc.title: lang="x-default" Jorge401

```

Ilustración 246 Ejecución y metadatos por pantalla 1. Fuente: Elaboración propia.

Ilustración 247 Ejecución y metadatos por pantalla 2. Fuente: Elaboración propia.



metadatos.py

12.2 Metadatos.py

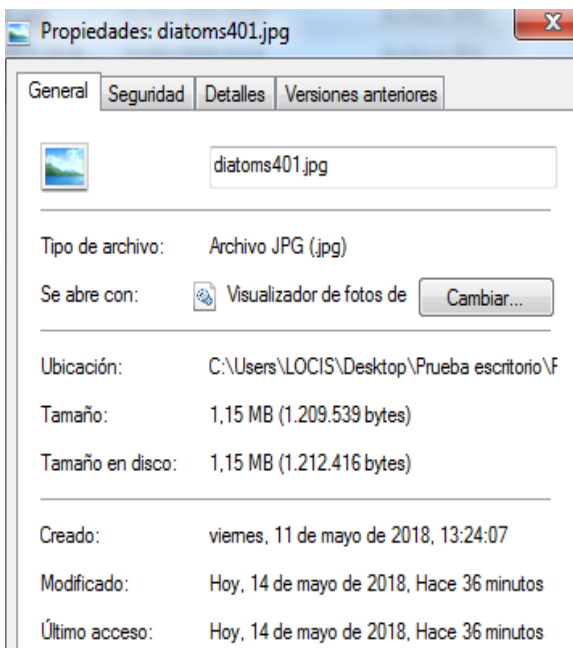


Ilustración 248 Propiedades de la imagen visto en

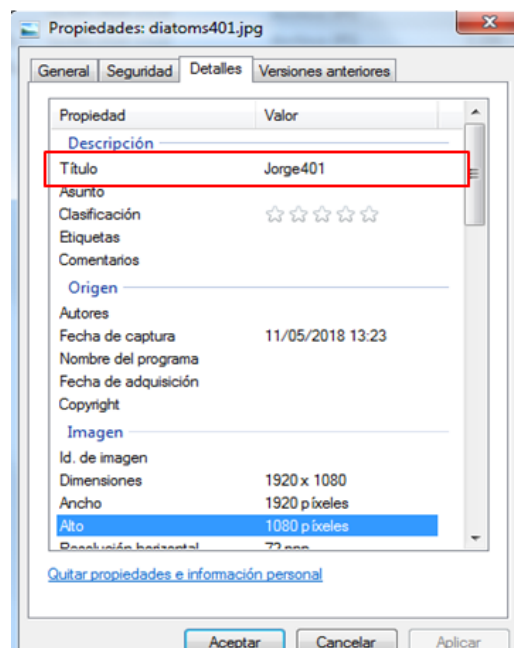


Ilustración 249 Propiedades de la imagen visto

Windows. Fuente: Elaboración propia.

en Windows. Fuente: Elaboración propia.

Para poder acceder a los metadatos referentes a información GPS (no accesibles a simple vista) es necesario descargar la librería Python ExifTool de Phil Harvey, directamente desde su sitio web (<https://www.sno.phy.queensu.ca/~phil/exiftool/index.html>) o desde algún repositorio.

Si descargamos la librería directamente desde el link se deben seguir una serie de pasos para proceder a su uso:

- Descomprimir el archivo .tar en la Raspberry, con el siguiente comando:

```
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo gzip -dc Image-ExifTool-10.96.tar.gz | tar -xf -
```

Ilustración 250 Instalación de un archivo comprimido .tar. Fuente: Elaboración propia.

- Comando por consola `cd Image-ExifTool-10.96`
- En ese directorio se tecléa:
 - `perl Makefile.PL`
 - `make test`
 - `sudo make install`

Ahora se puede usar el comando `exiftool` para crear un script como el siguiente encargado de dotar de metadatos GPS a una imagen:

```
lat=435915793# dato segun salida gps metido a mano
lat=1.0*lat #para convertir de int a float
lat=lat/10000000#esta multiplicado por 10 elevado a 7
print lat
import os
lon="1"#num de prueba poner datos gps
lat=str(lat)
print lon,lat
image="/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB/diatoms401.jpg"
command="sudo exiftool -GPSLongitude="+lon+" -GPSLatitude="+lat+" "+image
print command
os.system(command)
```

Ilustración 251 Script `DD_to_DMS.py`, dedicado a abrir un campo GPS para las imágenes RGB. Fuente: Elaboración propia.

Este script primero recibe un input (con el mismo formato que se recibiría desde la controladora siguiendo la comunicación MavLink), y después mediante el comando `exiftool` lo incrusta en la imagen .jpg de prueba como información GPS.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python DD_to_DMS.py
43.5915793
1 43.5915793
sudo exiftool -GPSLongitude=1 -GPSLatitude=43.5915793 /home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB/diatoms401.jpg
1 image files updated

```

Ilustración 252 Ejecución del programa anterior. Fuente: Elaboración propia



DD_to_DMS.py

12.3 DD_to_DMS.py

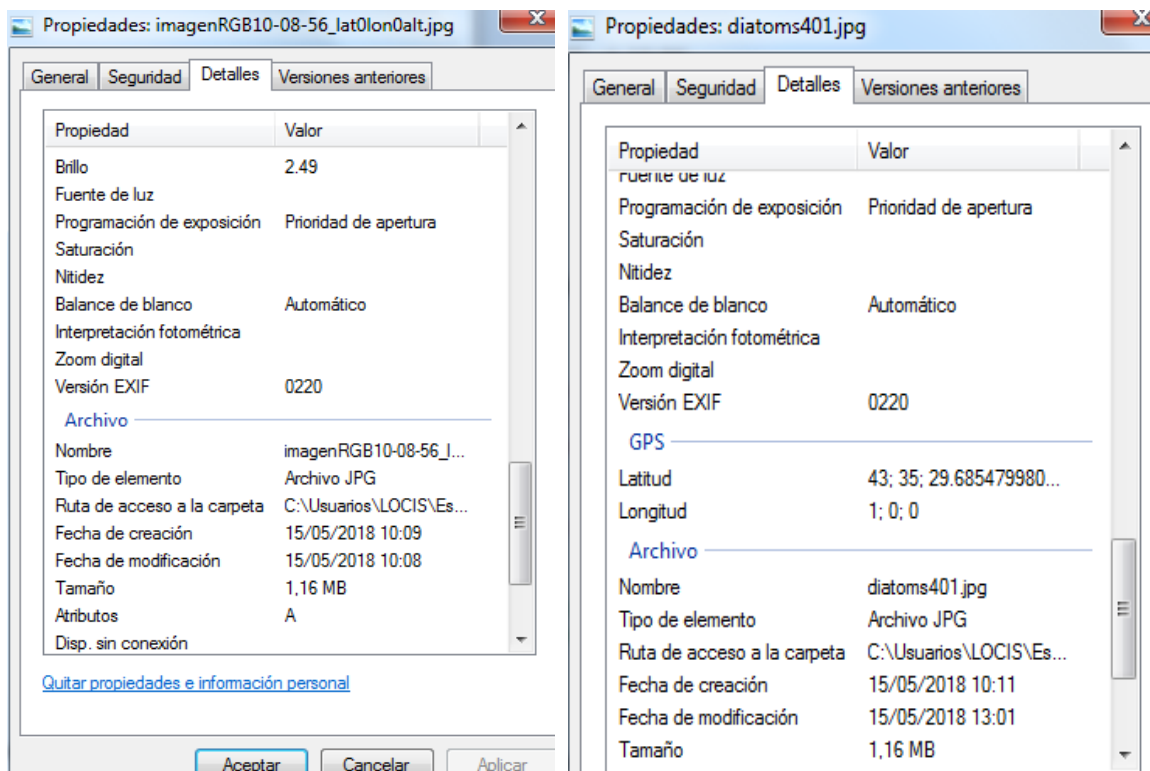


Ilustración 253 Imagen estándar con metadatos referentes a información GPS ocultos. Fuente: Elaboración propia.

Ilustración 254 Imagen objetivo del script tras ejecutarlo. Fuente: Elaboración propia.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python metadatos.py
Exif.GPSInfo.GPSLatitude: 43/1 35/1 61538/2073
Exif.GPSInfo.GPSLongitude: 1/1 0/1 0/1

```

Ilustración 255 Metadatos referentes a GPS de la imagen objetivo, hasta ahora ocultos. Fuente: Elaboración propia.



metadatos.py

12.4 metadatos.py . Fuente Elaboración propia.

Nota: Se puede observar como la latitud cambia de formato decimal DD a formato “DMS” (degree, minutes, seconds)

Posteriormente se intentará integrar la información GPS real (procedente de la controladora) en el mismo campo correspondiente a los metadatos de cada una de las imágenes.

12.7.- INTEGRACIÓN CON EL “MAIN” SCRIPT (AUTOMATICO.PY)

Se pretende implementar la solución que proporciona la posición GPS al script principal encargado de sacar fotos, para con ello poder referenciar a cada una de las fotos RGB con la latitud y longitud correspondientes al punto donde fueron sacadas.

Para ello se plantean las siguientes opciones:

- Realizar el incrustado de datos GPS como pos-procesado una vez en tierra
- Guardar los datos GPS y la referencia a la foto correspondiente para realizar la incrustación de datos GPS en la rutina de fin de misión
- Realizar la escucha de datos GPS como subproceso, y únicamente pasar los datos cuando se necesiten.
- Comando por consola

12.7.1.- Incrustado de datos GPS como posprocesado

Se implementa una solución consistente en incrustar datos GPS en las fotos RGB una vez en la estación de tierra usando el siguiente script:

```

GNU nano 2.2.6                                Fichero: incru_gps.py

import os
import string

"""saca las latitudes y long de todas las fotos dentro
del directorio ruta e incrusta como info gps"""
lon=[]
lat=[]
#con esto se ensena el primer elemento
#listado=os.listdir('/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB')[0]
# output= ref imagenRGB09-19-28_lat43.5231568lon-56112348alt.jpg

```

Ilustración 256 Script dedicado a incrustar datos GPS en posprocesado. Fuente: Elaboración propia.

```

def find_lat(ref):
    global lon,lat
    corte_anterior=ref.find("lat")+3
    #print corte_anterior
    lati=ref[corte_anterior:]
    corte_post=len(ref)-ref.find("lon")
    #print corte_post
    lati=lati[:-corte_post]
    #print lati
    return lati
def find_lon(ref):
    global lon,lat
    corte=ref.find("lon")+3
    print corte
    longi=ref[corte:]
    if ref.find("alt")!=-1:
        corte=len(ref)-ref.find("alt")
        print corte
        longi=longi[:-corte]
        return longi
    else:
        corte=len(ref)-ref.find(".j")
        print corte
        longi=longi[:-corte]
        return longi

```

```

try:
    path='/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB' #a directorio
    #para leerlos todos los files del directorio
    for file in os.listdir(path):
        #do something
        listado=file
        print listado
        lat=find_lat(listado)
        lon=find_lon(listado)
        print lat
        print lon
        lat=str(lat)
        lon=str(lon)
        listado=str(listado)
        #exiftool trabaja con string convertir antes
        image= path+"/"+listado #ruta a la foto
        image=str(image)
        command="sudo exiftool -GPSLongitude="+lon+" -GPSLatitude="+lat+" "+image
        print command
        os.system(command)
        #exif(lon,lat,image)
except KeyboardInterrupt:
    print("Saliendo a peticion")

```

Ilustración 257 Script dedicado a incrustar datos GPS en posprocesado. Fuente: Elaboración propia.

Ilustración 258 Script dedicado a incrustar datos GPS en posprocesado. Fuente: Elaboración propia.

Para ello se busca la latitud y la longitud correspondiente (en el nombre de la foto) y se incrusta como información de metadatos.

Esta solución se considero aceptable en un primer momento, pero se descarto por no compaginar con el objetivo de automatizar el proyecto.



incru_gps.py

12.5 incru_gps.py

12.7.2.- Comando por consola

Para realizar esta opción se debe introducir por consola el siguiente comando cada vez que el script principal saque una foto, y realizar así la incrustación de datos de igual forma que se hizo con el script DD_to_DMS:

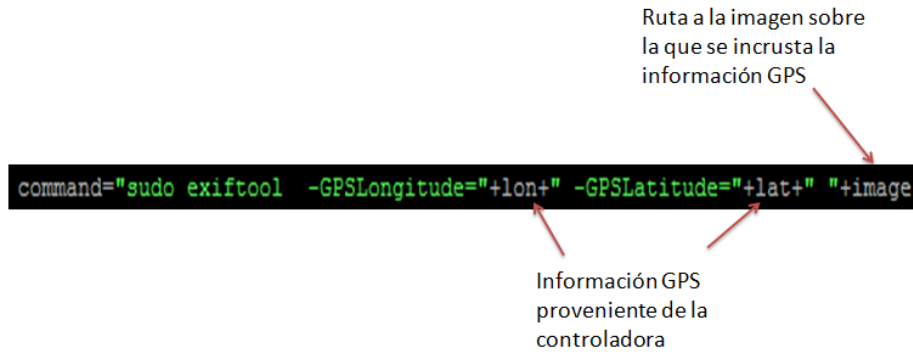


Ilustración 259 Comando por consola, adaptado a un script para usar “exiftool”. Fuente: Elaboración propia.

El script general se resume en:

- Primero se añaden las librerías dronekit y GExiv2, la primera permite realizar la conexión con la controladora y obtener los datos GPS, la segunda es importante para la visualización de metadatos.

```
import os
from picamera import PiCamera #as camera
#import RPI_to_PIX
from dronekit import connect
import dronekit
from gi.repository import GExiv2
from math import *
```

Ilustración 260 Importación de nuevas librerías. Fuente: Elaboración propia.

Posteriormente se realiza la conexión de la Raspberry con el vehículo (controladora de vuelo), especificando para ello el tipo de conexión, el número de baudios y que espere a realizar la conexión antes de saltar a la siguiente línea del script. De la siguiente manera:

```
lat=1
lon=1
alt=""
try:
    #inicializacion de variable contador
    i=0
    vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)#conexion con controladora por AMA0
```

Ilustración 261 Conexión de la controladora de vuelo con la Raspberry pi. Fuente: Elaboración propia.

En el bucle desde el que se da la orden de disparo de fotos se añade el oyente, que recibe la posición GPS de la controladora:

```
vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
time.sleep(0.1)#para darle tiempo al listener
```

Ilustración 262 Oyente dentro del bucle infinito de toma de fotos. Fuente: Elaboración propia.


```

def listener_gps_raw(self,name,posgps):#funcion oyente par
#print ("lat:%s\tlon:%s\tsat.number:%s\ten:%s" %(posgp
global lat,lon#,alt
lat=posgps.lat
lon=posgps.lon
        # lat=int(lat)
lat=1.0*lat #para convertir de int a float
lat=lat/10000000#esta multiplicado por 10 elevado a 7
lat=str(lat)

        # lon=int(lon)
lon=1.0*lon #para convertir de int a float
lon=lon/10000000#esta multiplicado por 10 elevado a 7
lon=str(lon)
alt=posgps.alt

```

Ilustración 263 Función que se dispara y proporciona las posiciones al activar el oyente. Fuente: Elaboración propia.

Una vez se tiene la posición se puede incrustar dando el “command” de la foto ”comando por consola adaptado” por consola, usando para ello “os.system(command)”

```

pi@raspi_locis:~/Desktop/CAPTURASDEVUELO $ sudo python automatico.py
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
1 image files updated
1 image files updated
1 image files updated
1 image files updated
1 image files updated
1 image files updated
1 image files updated

```

Ilustración 264 Resultado de ejecutar el script automatico.py. Fuente: Elaboración propia.

De modo que el script “automatico.py” se actualiza de la siguiente forma:

```

GNU nano 2.2.6                               Fichero: automatico.py
import os
from picamera import PiCamera #as camera
#import RPI to PIX
from dronekit import connect
import dronekit
from gi.repository import GEXiv2
from math import *

#inicializacion de pines GPIO

gpio.setmode(gpio.BCM) #designacion de pines GPIO en modo BCM
buzzer_pin=27
gpio.setup(buzzer_pin, gpio.OUT)
gpio.output(buzzer_pin, gpio.LOW)

gpio.setup(05, gpio.IN, pull_up_down=gpio.PUD_UP) #señal fin de mision, designa
gpio.setup(06, gpio.IN, pull_up_down=gpio.PUD_UP) #señal de disparo, designaci

gpio.setup(24, gpio.OUT)
disparoVUE= gpio.PWM(24 , 50)
disparoVUE.start(5)
camera=PiCamera()
time.sleep(5)
lat=1
lon=1
alt=""

```

```

try:
#inicializacion de variable contador
i=0
vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)
def beep (): #en la primera prueba que se hizo beep era free
global buzzer_pin #probar esta linea, no creo que import
gpio.output(buzzer_pin,gpio.HIGH)
time.sleep(2)
gpio.output(buzzer_pin,gpio.LOW)
time.sleep(2)
def capturaVUE ():
disparoVUE.ChangeDutyCycle(10)
time.sleep(0.5)
disparoVUE.ChangeDutyCycle(5)
def listener_gps_raw(self,name,posgps):#funcion oyente para
#print ("lat:%s\tlon:%s\tsat.number:%s\tten:%s" %(posgps.
global lat,lon#,alt
lat=posgps.lat
lon=posgps.lon
# lat=int(lat)
lat=1.0*lat #para convertir de int a float
lat=lat/10000000#esta multiplicado por 10 elevado a 7
lat=str(lat)
# lon=int(lon)
lon=1.0*lon #para convertir de int a float
lon=lon/10000000#esta multiplicado por 10 elevado a 7
lon=str(lon)
alt=posgps.alt

```

Ilustración 265 Actualización automatico.py inicio. Fuente: Elaboración propia.

```

def interrupt(channel): #definicion del programa de interrupcion
os.system('sudo python /home/pi/Desktop/CarpetaCompartidaPRUEBA/localizacion_fotos.py')
os.system("find /home/pi/Desktop/CAPTURASDEVUELO -type f -name '*.jpg' -exec cp -rf {} /home/pi/Desktop/CarpetaCompartidaPRUEBA/ \;")
os.system("find /media/pi/9016-4EF81 -type f -name '*.jpg' -exec cp -rf {} /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR \;") #copia
os.system("find /media/pi/9016-4EF82 -type f -name '*.jpg' -exec cp -rf {} /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR \;") #copia
os.system("find /media/pi/9016-4EF81 -type f -name '*.jpg' -exec rm -rf {} \;") #borra las fotos de la SD de la FLIR
os.system("find /media/pi/9016-4EF82 -type f -name '*.jpg' -exec rm -rf {} \;") #borra las fotos de la SD de la FLIR

tiempo_i=[]
tiempo_antes_com=[]
tiempo_despues_com=[]
lati=[]
longi=[]
#chequeo de la orden de fin de mision
gpio.add_event_detect(05,gpio.FALLING,callback = interrupt) #interrupcion por deteccion de pulso Low o 0 logico en el pin GPIO 05, llama
#inicio de bucle infinito de chequeo de señal de disparo

```

```

while True:
status=gpio.input(06) #chequeo del pin GPIO 06
if status == False: #si se recibe un pulso se pasa a toma de imagenes
i=i+1 #actualizacion de contador
vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
time.sleep(0.1)#para darle tiempo al listener
tiempo=time.strftime("%H-%M-%S") #llamada al tiempo actual en formato 24 horas
tiempo_i.append(tiempo)
image="/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB/imagenRGB%s_lat%s_lon%s_alt%s.jpg"%(tiempo,lat,lon,alt)
camera.capture(image) #captura de imagen con RaspicAM, etiquetado con la hora del disparo correspondiente
#metadata = GEXiv2.Metadata(image)
#metadata["Xmp.dc.title"] = "lat:%s\tlon:%s\talt%s\tten %s"%(lat,lon,alt,tiempo)#altera los metadatos de l
#metadata.save_file()
#lat=str(lat)#igual este comandodebe ir antes de image
image=str(image)
lati.append(lat)
longi.append(lon)
command="sudo exiftool -GPSLongitude="+lon+" -GPSLatitude="+lat+" "+image
tiempo_antes_com.append(tiempo)
os.system(command)#check si sabe donde esta el exiftool
tiempo_despues_com.append(tiempo)
capturaVUE() #captura de imagen con cámara FLIR VUE PRO, la imagen se guarda en un dir

```

```

except KeyboardInterrupt:
print("saliendo del programa a peticion")
print tiempo_i
print tiempo_antes_com
print tiempo_despues_com
print lati
print longi
#except:
print("ocurrio error o excepcion")
finally:
gpio.cleanup() #limpia todos los puertos

"""prueba el main y si no permite interrupcion por teclado con con
aviso de error y limpieza de puertos"""
#-----Fin de Programa-----

```

Ilustración 266 Actualización automatico.py final . Fuente: Elaboración propia.



automatico.py

12.6 Actualización de automatico.py

Es importante borrar las imágenes originales, puesto que el programa crea una copia con datos gps y una original sin ellos. Usamos para ello el siguiente comando por consola:

	imagenRGB10-47-2...	17/05/2018 10:47	Archivo JPG	1.149 KB
	imagenRGB10-47-2...	17/05/2018 10:47	Archivo JPG_ORIG...	1.150 KB

Ilustración 267 Imagen original e imagen con datos GPS incrustados. Fuente. Elaboración propia.

- `find RGB -type f -name '*original*' -delete` (dar este comando desde la carpeta compartida)

Para borrar las imágenes en conflicto que pueda crear Syncthing (software usado para sincronizar la carpeta compartida de la Raspberry y la estación de tierra) se usará el mismo comando cambiando “original” por “conflict” ó “syncthing” para el caso de aquellos archivos que se sincronizaron de manera incorrecta.

Si se usa el siguiente script, se muestran por consola la información GPS de cada una de las fotos del directorio objetivo:

```
GNU nano 2.2.6                                Fichero: gpsinfo.py
from gi.repository import GExiv2
import os
# da GPS de todas las fotos de un file
path='/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB' #a directorio
for file in os.listdir(path):
    listado=file
    print listado
    image=path+"/"+listado
    metadata = GExiv2.Metadata(image)
    print "time\t"+metadata['Exif.Photo.DateTimeOriginal']
    print "lon\t"+metadata['Exif.GPSInfo.GPSLongitude']
    print "lat\t"+metadata['Exif.GPSInfo.GPSLatitude']
```

Ilustración 268 gpsinfo.py Fuente: Elaboración propia.

```

pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python gpsinfo.py
imagenRGB16-11-47_lat0.0lon0.0alt.jpg
time      2018:05:16 16:11:47
lon       0/1 0/1 0/1
lat       0/1 0/1 0/1
imagenRGB16-46-02_lat0.0lon0.0alt.jpg
time      2018:05:16 16:46:02
lon       0/1 0/1 0/1
lat       0/1 0/1 0/1
imagenRGB16-46-10_lat0.0lon0.0alt.jpg
time      2018:05:16 16:46:10
lon       0/1 0/1 0/1
lat       0/1 0/1 0/1
imagenRGB16-29-02_lat0.0lon0.0alt.jpg
time      2018:05:16 16:29:02
lon       0/1 0/1 0/1
lat       0/1 0/1 0/1
imagenRGB16-29-25_lat0.0lon0.0alt.jpg
time      2018:05:16 16:29:25
lon       0/1 0/1 0/1
lat       0/1 0/1 0/1

```

Ilustración 269 Output de gpsinfo.py en la oficina. Fuente: Elaboración propia.

<pre> GNU nano 2.2.6 Fichero: gpsinfo.py from gi.repository import GExiv2 import os # da GPS de todas las fotos de un file path="/home/pi/Desktop/CarpetaCompartidaPRUEBA/RGB" #a directorio for file in os.listdir(path): listado=file print listado image=path+"/"+listado metadata = GExiv2.Metadata(image) print "time\t"+metadata['Exif.Photo.DateTimeOriginal'] print "lon\t"+metadata['Exif.GPSInfo.GPSLongitude'] print "lat\t"+metadata['Exif.GPSInfo.GPSLatitude'] </pre>	<pre> pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA \$ sudo python gpsinfo.py imagenRGB16-11-47_lat0.0lon0.0alt.jpg time 2018:05:16 16:11:47 lon 0/1 0/1 0/1 lat 0/1 0/1 0/1 imagenRGB16-46-02_lat0.0lon0.0alt.jpg time 2018:05:16 16:46:02 lon 0/1 0/1 0/1 lat 0/1 0/1 0/1 imagenRGB16-46-10_lat0.0lon0.0alt.jpg time 2018:05:16 16:46:10 lon 0/1 0/1 0/1 lat 0/1 0/1 0/1 imagenRGB16-29-02_lat0.0lon0.0alt.jpg time 2018:05:16 16:29:02 lon 0/1 0/1 0/1 lat 0/1 0/1 0/1 imagenRGB16-29-25_lat0.0lon0.0alt.jpg time 2018:05:16 16:29:25 lon 0/1 0/1 0/1 lat 0/1 0/1 0/1 </pre>
--	---

Ilustración 270 gpsinfo.py Fuente: Elaboración propia.

Ilustración 271 Output de gpsinfo.py en la oficina. Fuente: Elaboración propia.



gpsinfo.py

12.7 gpsinfo.py

El día 17/05/2018 se realiza una prueba en el exterior de la oficina del script automatico.py con los siguientes resultados:

```

pi@raspi_locis:~/Desktop
>>> APM:Copter V3.5.5 (2
>>> PX4: 0384802e NuttX:
>>> Frame: QUAD
>>> PX4v2 001A0029 31345
>>> u-blox 1 HW: 0007000
1 image files update
1 image files update
1 image files update
1 image files update
1 image files update
^Csaliendo del programa
['10-35-57', '10-36-01',
['10-35-57', '10-36-01',
['10-35-57', '10-36-01',
['43.522476', '43.522476
['-5.6112609', '-5.61126

```

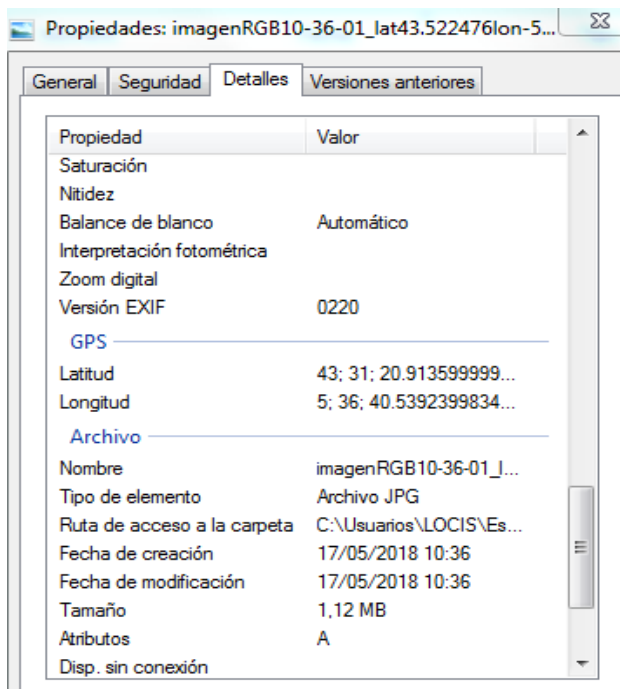


Ilustración 272 Ejecución manual del programa para poder ver resultados por pantalla. Fuente: Elaboración propia.

Ilustración 273 Foto con datos GPS incrustados. Fuente: Elaboración propia.

```

imagenRGB10-48-30_lat43.522476lon-5.6112609alt.jpg
time    2018:05:17 10:48:30
lon     5/1 36/1 196291/4842
lat     43/1 31/1 13071/625
imagenRGB16-46-02_lat0.0lon0.0alt.jpg
time    2018:05:16 16:46:02
lon     0/1 0/1 0/1
lat     0/1 0/1 0/1
imagenRGB10-48-54_lat43.522476lon-5.6112609alt.jpg
time    2018:05:17 10:48:54
lon     5/1 36/1 196291/4842
lat     43/1 31/1 13071/625
imagenRGB10-52-58_lat43.522476lon-5.6112609alt.jpg
time    2018:05:17 10:52:58
lon     5/1 36/1 196291/4842
lat     43/1 31/1 13071/625

```

Ilustración 274 Output gpsinfo.py en la carpeta compartida. Fuente: Elaboración propia

De tal modo que se puede caracterizar la prueba como positiva,dejándose de lado el estudio de las otras alternativas y quedando solucionado la geolocalización de las fotos RGB.

12.8.- NOMBRADO DE FOTOS FLIR CON DATOS GPS

La cámara térmica, por si misma es capaz de adquirir información GPS como metadatos, pero usando para ello un puerto de telemetría (del cual en principio no se puede disponer).

Es por esto que se realizará un proceso similar al de las fotos RGB para adquirir esta información.

Se pretende renombrar las fotos FLIR con un nombre acorde a su posición GPS, primero por medio de un script de Python: llamado renameflir.py ubicado en la carpeta compartida (que luego se implementará en el script general automatico.py).



renameflir.py

12.8 renameflir.py

```

try:
import RPi.GPIO as gpio
import os
import time
import string
import shutil
from dronekit import connect
import dronekit
import datetime as dt

gpio.setmode(gpio.BCM)
gpio.setwarnings(False)
gpio.setup(24, gpio.OUT)
disparoVUE= gpio.PWM(24 , 50)
disparoVUE.start(5)
time.sleep(2)
lat=1
lon=1
listado=[]
flir="hi"
def listener_gps_raw(self,name,posgps):
#print ("lat:%s\tlon:%s\ttsat.numbe
global lat,lon#,alt
lat=posgps.lat
lon=posgps.lon
# lat=int(lat)
lat=1.0*lat #para convertir de int
lat=lat/10000000#esta multiplicado
lat=str(lat)

# lon=int(lon)
lon=1.0*lon #para convertir de int
lon=lon/10000000#esta multiplicado
lon=str(lon)
alt=posgps.alt
def capturaVUE ():
disparoVUE.ChangeDutyCycle(10)
time.sleep(0.5)
disparoVUE.ChangeDutyCycle(5)
#localiza los archivos del tipo es
def listad(path):

```

Ilustración 275 renameflir.py en carpeta compartida. Fuente: Elaboración propia.

```

def listad(path):
for root, dirs, files in os.walk(path):
for file in files:

if file.endswith(".jpg"):
global flir
flir=os.path.join(root,file)
global listado
listado.append(flir)
print os.path.exists(os.path.join(root,file)
print "flir es %s"%flir
#return os.path.join(root,file)
"""
if file.find("*.jpg")!=-1:
print os.path.join(root,file)
"""

vehicle = connect('/dev/ttyAMA0',baud=57600, wait_ready=True)
path1='/media/pi'
path='/media/pi/9016-4EF81'
path2='/home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR' #a di
#para leerlos todos los files del directorio

```

Ilustración 276 renameflir.py en carpeta compartida. Fuente: Elaboración propia.

```

while True:

capturaVUE()
time.sleep(4.5)#alsacar la foto hay que dejar un time pra poder
#acceder a ella
listad(path1)
print path1
print "la direccion a la foto es %s "%flir
print os.path.exists(path1)
vehicle.add_message_listener('GPS_RAW_INT',listener_gps_raw)
time.sleep(0.1)#para darle tiempo al listener
tiempo = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
newpath=path2+"/"+lat=%slon=%s"%(lat,lon)+'.jpg'
newpath=os.path.join(path2,"timestamp=%slat=%slon=%s"%(tiempo,lat,lon)+'.jpg")
print os.path.exists(path2)
print listado
#print os.path.exists(flir)
print flir
print "la nueva direccion es %s"%newpath
shutil.move(flir,newpath)
"""probar con esto y si no renombrar en mismo directorio y luego mover"""
print os.path.exists(newpath)

except KeyboardInterrupt:
print "saliendo a peticion"

```

Ilustración 277 renameflir.py en carpeta compartida. Fuente: Elaboración propia.

Dicho script:

- Manda la orden a la cámara térmica para sacar una foto.
- Busca la foto dentro de la cámara, le asigna un nombre con su “timestamp”, “lon”, “lat” y la ubica en la carpeta compartida.

```

-rw-r--r-- 1 root root 60142 may 25 2018 timestamp=2018-05-25 11:39:12lat=43.522476lon=-5.6112609.jpg
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $

```

Ilustración 278 Ejemplo de nombrado de foto térmica. Fuente: Elaboración propia.

Posteriormente se actualiza el script para incrustar metadatos de posición GPS en las imágenes térmicas:



renameflir.py

12.9 Actualización del programa para incrustar metadatos en las imágenes. Fuente: Elaboración propia.

```
la dirección a la foto es /media/pi/9016-4EF81/20180528_155105.jpg esTrue
la nueva dirección es /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-28-16-55-05lat=43.522476lon=-5.6112609.jpg y es True
comando es sudo exiftool -GPSLongitude=-5.6112609 -GPSLatitude=43.522476 /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-28-16-55-05lat=43.522476lon=-5.6112609.jpg
1 image files updated
```

Ilustración 279 Ejemplo renameflir.py . Fuente. Elaboración propia.

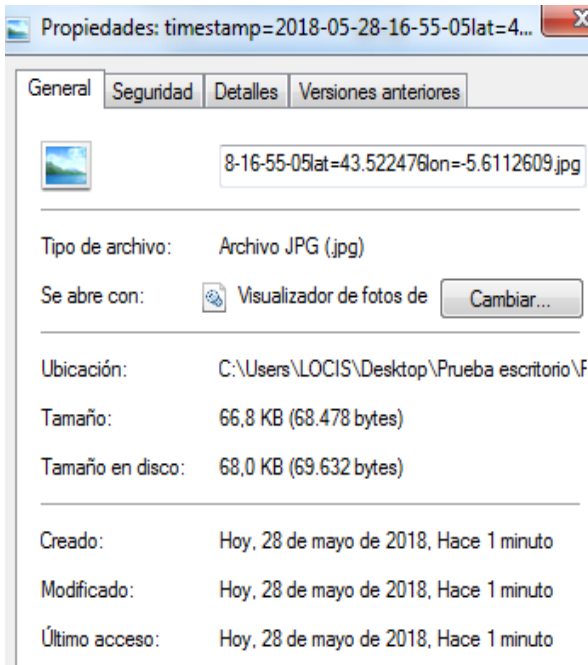


Ilustración 280 Propiedades de la foto generada . Fuente: Elaboración propia.

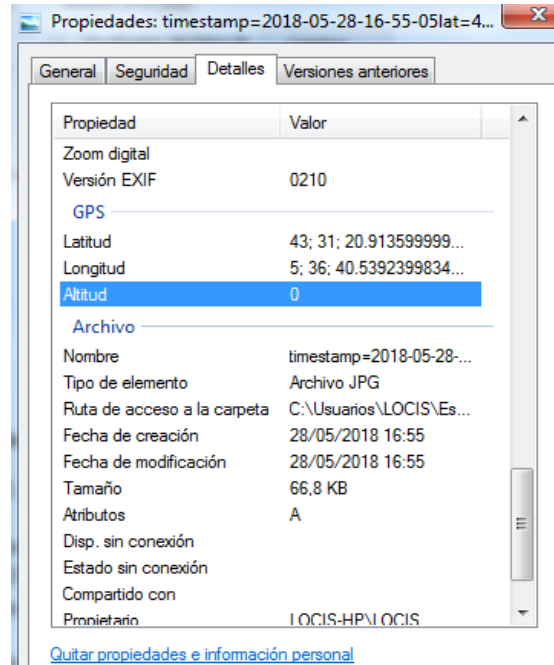


Ilustración 281 Propiedades de la foto generada . Fuente: Elaboración propia.

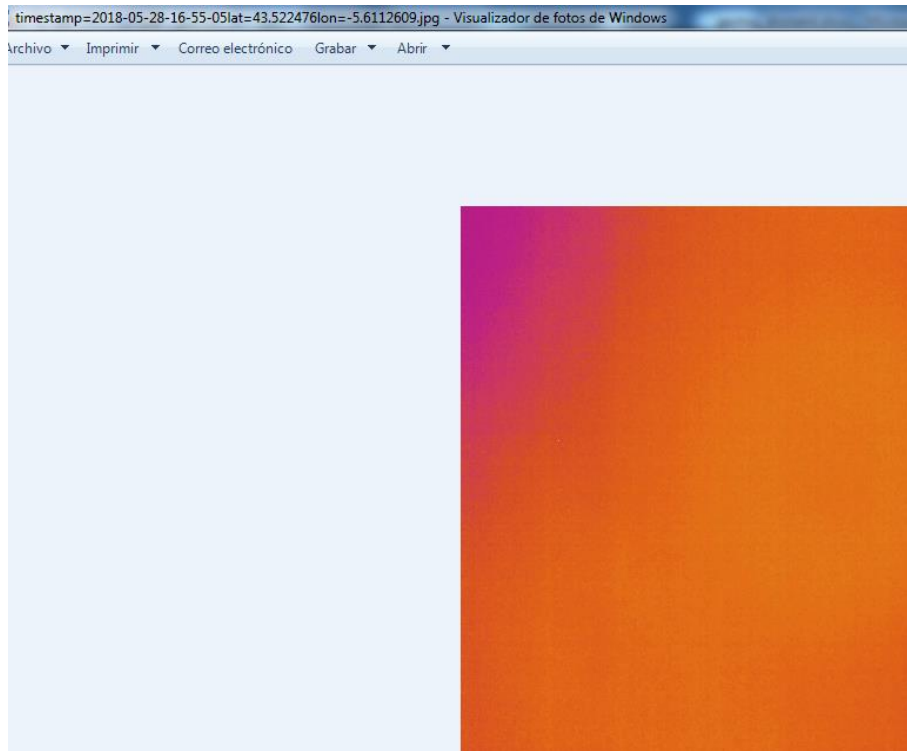


Ilustración 282 Foto generada (visión térmica de la mesa, por eso es uniforme). Fuente: Elaboración propia.

12.8.1.- Solución de errores:

- “Link 1 down”

Cuando se conecta la cámara FLIR y la Pixhawk con la Raspberry alimentada mediante el USB desde la torre del ordenador no se consigue establecer un puente MAVLink entre la controladora y la Raspberry (error link 1 down).

Se sospecha que este problema sea debido a una falta de alimentación al conectar los dos dispositivos a la Raspberry al mismo tiempo.

Desde una conexión USB-microUSB se puede suministrar un máximo de 1,5 A lo que es insuficiente para alimentar la Raspberry, la controladora, la cámara RGB y la cámara FLIR.

Se proba a realizar la alimentación de la forma convencional (usando la batería de vuelo del dron, dado que esta alimenta a la controladora y ya no tendría que ser alimentada desde la Raspberry).

Otra opción si aún así no llegamos a la alimentación de todos los dispositivos sería alimentar todo con la batería menos la Raspberry con el cable microUSB.

Al alimentar desde la batería no se produce ningún problema de conexión con la controladora:

```
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python renameflir.py
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
flir global es hi
True
flir es /media/pi/9016-4EF81/20180525_103304.jpg
True
flir es /media/pi/9016-4EF81/20180525_103517.jpg
/media/pi
la direccion a la foto es /media/pi/9016-4EF81/20180525_103517.jpg
True
True
['/media/pi/9016-4EF81/20180525_103304.jpg', '/media/pi/9016-4EF81/20180525_103517.jpg']
/media/pi/9016-4EF81/20180525_103517.jpg
la nueva direccion es /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-25 11:39:07lat=1lon=1.jpg
```

Ilustración 283 Uso de renameflir.py. Fuente: Elaboración propia.

12.8.2.- Integración en el proyecto

Se pretende integrar el geotiquetado de las fotos térmicas en el script principal de toma de fotos, para ello:



automatico.py

Ilustración 284 Actualización de autoamatico.py Fuente: Elaboración propia.

```
pi@raspi_locis:~/Desktop/CarpetaCompartidaPRUEBA $ sudo python automatico.py
<Screen 'lid'>
>>> APM:Copter V3.5.5 (27229c83)
>>> PX4: 0384802e NuttX: 1bcae90b
>>> Frame: QUAD
>>> PX4v2 001A0029 3134510F 36313935
  1 image files updated
la direccion a la foto es /media/pi/9016-4EF81/20180530_095609.jpg
la direccion a la foto es /media/pi/9016-4EF81/20180530_095609.jpg esTrue
la nueva direccion es /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-30-12-04-00lat=43.522476lon=-5.6112609.jpg y es True
comando es sudo exiftool -GPSLongitude=-5.6112609 -GPSLatitude=43.522476 /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-30-12-04-00lat=43.522476lon=-5.6112609.jpg
  1 image files updated
  1 image files updated
la direccion a la foto es /media/pi/9016-4EF81/20180528_151647.jpg
la direccion a la foto es /media/pi/9016-4EF81/20180528_151647.jpg esTrue
la nueva direccion es /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-30-12-04-10lat=43.522476lon=-5.6112609.jpg y es True
comando es sudo exiftool -GPSLongitude=-5.6112609 -GPSLatitude=43.522476 /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-30-12-04-10lat=43.522476lon=-5.6112609.jpg
  1 image files updated
  1 image files updated
la direccion a la foto es /media/pi/9016-4EF81/20180528_140859.jpg
la direccion a la foto es /media/pi/9016-4EF81/20180528_140859.jpg esTrue
la nueva direccion es /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-30-12-04-26lat=43.522476lon=-5.6112609.jpg y es True
comando es sudo exiftool -GPSLongitude=-5.6112609 -GPSLatitude=43.522476 /home/pi/Desktop/CarpetaCompartidaPRUEBA/FLIR/timestamp=2018-05-30-12-04-26lat=43.522476lon=-5.6112609.jpg
  1 image files updated
  1 image files updated
```

Ilustración 285 Ejecución de automatico.py actualizado. Fuente: Elaboración propia.

La primera “image file updated” corresponde con una imagen RGB, luego se ve como se busca una foto térmica en el directorio /media/pi (denominación de la cámara térmica) para luego cambiarle el nombre e incrustarle información GPS (por eso se ven dos “image file updated”).

Nota: Es importante tener permiso de escritura sobre la carpeta destino “FLIR”, se consiguen con sudo chmod 777 FLIR por consola de comandos.

```
drwxrwxrwx 2 pi pi 24576 may 31 11:51 FLIR
```

Ilustración 286 Permisos en la carpeta destino. Fuente: Elaboración propia.

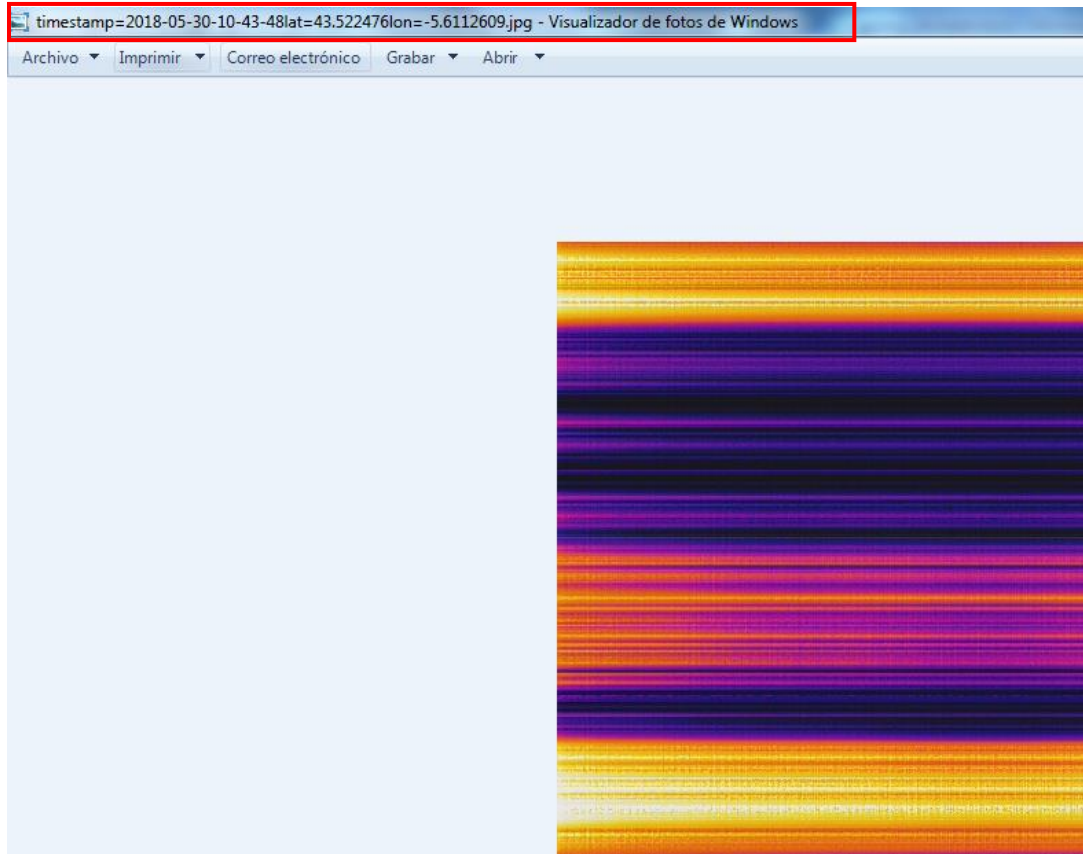


Ilustración 287 Imagen generada por automatico.py actualizado. Fuente: Elaboración propia.

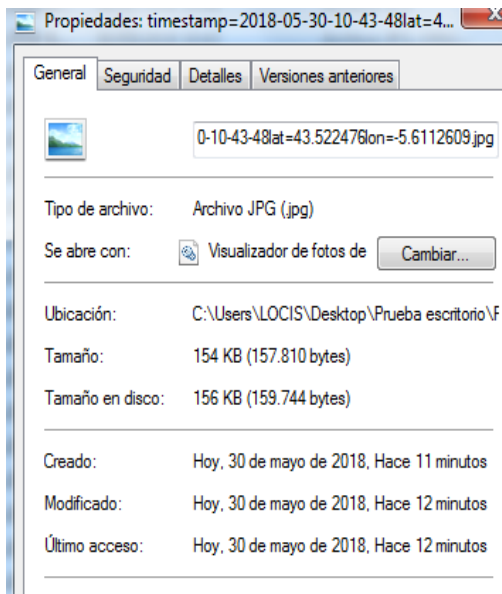


Ilustración 288 Imagen generada por automatico.py actualizado. Fuente: Elaboración propia.

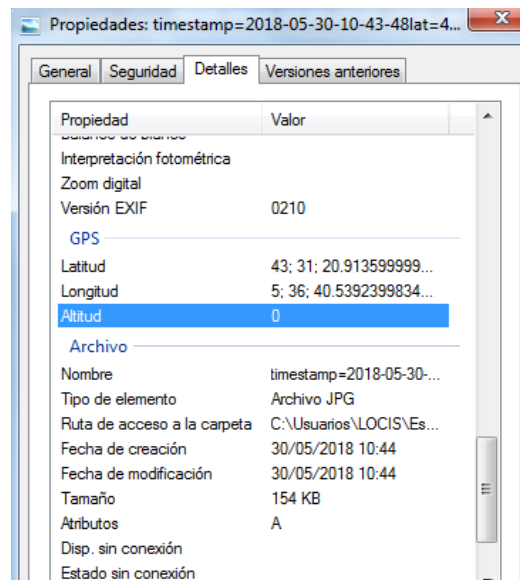


Ilustración 289 Imagen generada por automatico.py actualizado. Fuente: Elaboración propia.

13. Pruebas de vuelo

13.1.- INTRODUCCIÓN

En este documento se pretende recopilar las pruebas de vuelo realizadas durante las prácticas formativas (de 12-febrero-2018 a 31-junio-2018) del alumno Jorge Vales-Villamarín Sanjurjo en la empresa LOCIS SIGHTECH relacionadas con los siguientes proyectos:

- UAV-Inspection
- UAV-Torres de tensión

Se pretende recopilar el mayor contenido de información y documentación gráfica para dar respuesta a las consiguientes preguntas:

- Resumen
- Fecha y lugar de la realización de la prueba
- Motivo de la prueba de vuelo
- Misión (Waypoints) a realizar
- Resultados de la prueba
- Conclusiones
- Otros comentarios

13.2.- PRUEBA DE VUELO ROCES 1

13.2.1.- Resumen

Se realizan dos vuelos sobre paneles fotovoltaicos a diferentes alturas, 13m y 26m, con una velocidad de vuelo durante la captura de 1m/s.

El script de funcionamiento estaba configurado para enviar una orden de captura de imagen por segundo.

13.2.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

13.2.3.- Fecha y lugar de realización

El día 13 de febrero de 2018 se realizó una prueba de vuelo ubicada en el campo de fútbol del TSK Rocés, equipo patrocinado por TSK (empresa colaboradora en el proyecto UAV-Inspection).

13.2.4.- Motivo de la prueba

Se pretende realizar una prueba de funcionamiento básico (vuelo y captura de imagen RGB+Térmica) del dron UAV-Inspection.

13.2.5.- Misión por realizar

Se espera que el dron realice un recorrido en forma de rectángulo sobre las placas solares ubicadas en el campo de fútbol del equipo TSK Rocés.

Para ello es necesario especificar una sucesión de Waypoints como se aprecia a continuación:

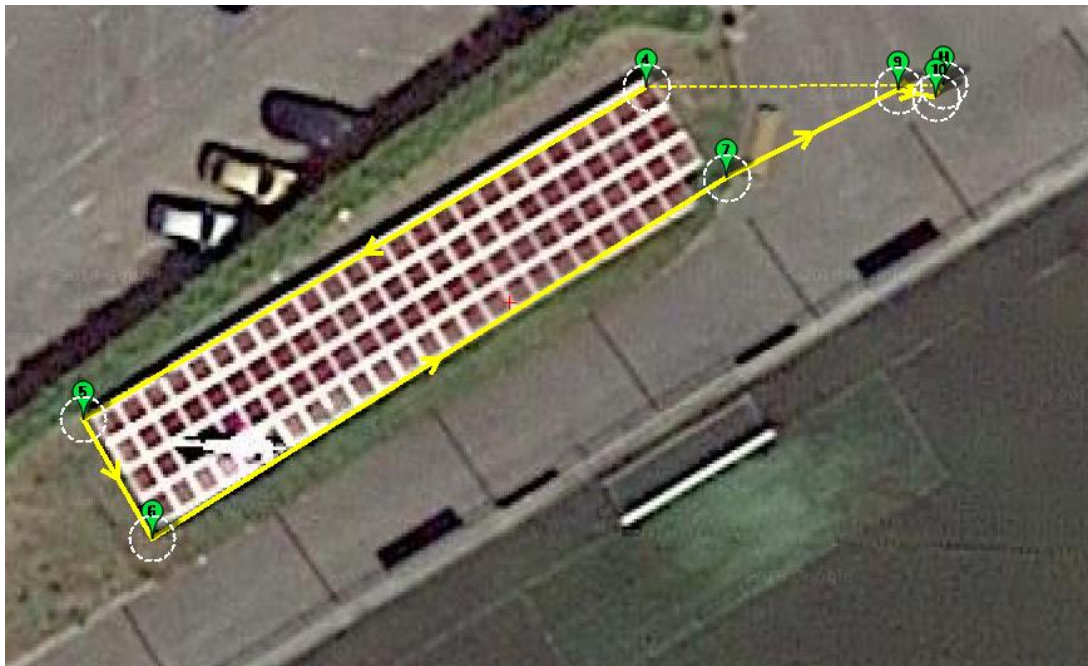


Ilustración 290 Misión a realizar durante la prueba. Fuente: Servidor interno LOCIS

Waypoints															
Radio WP	Radio Perder Tiempo por Defecto				Terrain	Verify Height	Agregar	Alt Wam	Spline						
1	60	100						0							
	Comandos	Dela			Lat	Long	Alt	Borrar	Amb	Abajo	Grad %	Angle	Dist	AZ	
1	TAKEOFF	0	0	0	43,519974	-5,675825	13	X			0	0	0	0	
2	DO_CHANGE_SPEED	1	1	0	0	0	0	X			0	0	0	0	
3	DO_SET_RELAY	0	1	0	0	0	0	X			0	0	0	0	
4	WAYPOINT	0	0	0	43,519964	-5,67598	13	X			-189,2	-62,1	27,4	270	
5	WAYPOINT	0	0	0	43,519835	-5,676281	13	X			0,0	0,0	28,2	239	
6	WAYPOINT	0	0	0	43,519789	-5,676244	13	X			0,0	0,0	5,9	150	
7	WAYPOINT	0	0	0	43,519929	-5,675937	13	X			0,0	0,0	29,2	58	
8	DO_SET_RELAY	0	0	0	0	0	0	X			0	0	0	0	
9	WAYPOINT	0	0	0	43,519963	-5,675845	13	X			0,0	0,0	8,3	63	
10	LAND	0	0	0	43,51996	-5,675825	13	X			0,0	0,0	1,6	102	

Ilustración 291 Waypoints o correspondientes a la prueba. Fuente: Servidor interno Locis.

En donde se realiza la siguiente sucesión:

- Comando 1: Despegue (TAKEOFF) y ascensión a 13m de altura.

- Comando 2: Cambio de velocidad a 1m/s (DO_CHANGE_SPEED).
- Comando 3: Establecer el voltaje del pin del primer relé como 1, correspondiente con “On” y con 3,3 V en PIXHAWCK (DO_SET_RELAY). Esta orden según el script automatico.py se corresponde con inicio de toma de fotos RGB y térmica.
- Comando 4: Coordenadas del primer waypoint (punto 4), correspondiente con el primer vértice del rectángulo.
- Comando 5: Coordenadas del segundo waypoint (punto 5), correspondiente con el segundo vértice del rectángulo.
- Comando 6: Coordenadas del tercer waypoint (punto 6), correspondiente con el tercer vértice del rectángulo.
- Comando 7: Coordenadas del cuarto waypoint (punto 7), correspondiente con el cuarto vértice del rectángulo.
- Comando 8: Establecer el voltaje del pin del primer relé como 0, correspondiente con “Off” y con 0V en PIXHAWCK (DO_SET_RELAY). Correspondiente con la orden de dejar de sacar fotos.
- Comando 9: Coordenadas del último waypoint (punto 9), correspondiente con el punto de aterrizaje.
- Comando 10: Orden de aterrizaje (LAND).

Para la segunda misión realizada justo después de la primera solamente es necesario cambiar la altura especificada en cada uno de los puntos implicados de 13m a 26m.

Los archivos .txt que detallan la misión para cada uno de los casos son:

```

hgc WPL 110
0 1 0 16 0 0 0 43.519965 -5.675821 37.259998 1
1 0 10 22 0.000000 0.000000 0.000000 0.000000 43.519974 -5.675825 13.000000 1
2 0 10 178 1.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1
3 0 10 181 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1
4 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519964 -5.675980 13.000000 1
5 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519835 -5.676281 13.000000 1
6 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519789 -5.676244 13.000000 1
7 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519929 -5.675937 13.000000 1
8 0 10 181 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1
9 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519963 -5.675845 13.000000 1
10 0 10 21 0.000000 0.000000 0.000000 0.000000 43.519960 -5.675825 13.000000 1

```

Ilustración 292 Archivo .txt con los datos de la primera misión de vuelo. Fuente: Servidor interno LOCIS.

```

hgc WPL 110
0 1 0 16 0 0 0 43.519965 -5.675821 37.259998 1
1 0 10 22 0.000000 0.000000 0.000000 0.000000 43.519974 -5.675825 26.000000 1
2 0 10 178 1.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1
3 0 10 181 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1
4 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519964 -5.675980 26.000000 1
5 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519835 -5.676281 26.000000 1
6 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519789 -5.676244 26.000000 1
7 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519929 -5.675937 26.000000 1
8 0 10 181 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1
9 0 10 16 0.000000 0.000000 0.000000 0.000000 43.519963 -5.675845 26.000000 1
10 0 10 21 0.000000 0.000000 0.000000 0.000000 43.519960 -5.675825 26.000000 1

```

Ilustración 293 Archivo .txt con los datos de la segunda misión de vuelo. Fuente: Servidor interno LOCIS.

13.2.6.- Resultados de la prueba

13.2.6.1.- Captura de imágenes

Según el archivo log de ambas misiones registrado por la controladora de vuelo, el tiempo de captura de imagen, tiempo transcurrido entre la orden de inicio de captura de fotos y la de cese, fue de aproximadamente 100 segundos. Tiempo durante el cual se han obtenido:

- Primer vuelo (correspondiente a altura de 13m):
 - No se encuentran imágenes de ninguno de los dos tipos al volver a la oficina, no se puede asegurar si se borraron o si directamente no se sacaron.
- Segundo vuelo (correspondiente a altura de 26m):
 - 56 imágenes RGB
 - 34 imágenes TIR

Será necesario incidir en:

- Por qué no existen imágenes de la primera misión.

13.2.6.2.- Desarrollo del vuelo

El dron realizó las dos misiones siendo capaz de despegar y aterrizar de forma autónoma y sin incidentes.

Sin embargo en la segunda misión de vuelo, a 26 m, durante el tramo entre los puntos de paso 6 y 7 el dron realizó el desplazamiento de forma lateral, con el morro orientado de forma perpendicular a la trayectoria de vuelo, en lugar de orientado con la trayectoria, como era de esperar.

Será necesario incidir en:

- Vuelo entre los puntos 6 y 7.

13.2.6.3.- Transferencia de imágenes

Se produce la transferencia de imágenes de la segunda misión, pero sin encontrar resultados de la primera.

13.2.7.- Imágenes de archivo

13.2.8.- Conclusiones

Como se comentaba anteriormente los principales problemas descubiertos durante esta prueba hacen referencia a:

- El vuelo entre los puntos 6 y 7
- Inexistencia de fotos de la primera misión

13.2.9.- Vuelo entre los puntos 6 y 7

En el primer caso y analizando la configuración de la controladora, se comprueba como el parámetro WP_YAW_BEHAVIOUR obliga al dron a encarar los puntos de paso al realizar una misión automática. Este comportamiento no se replicó entre los puntos 6 y 7.

El análisis del log de cada misión de vuelo, en lo que concierne al parámetro ATT YAW, que registra los valores de guiñada del dron (consultar más sobre este término en el apéndice bajo el nombre “Controles de vuelo de una aeronave”) muestra un registro del comportamiento anómalo del dron en la segunda misión en el tramo 6-7.

El dron realizo el tramo con una orientación de aproximadamente 50° en el vuelo 1, valor que se esperaba, y de unos 160° en el vuelo 2. Como se puede ver en las imágenes siguientes:

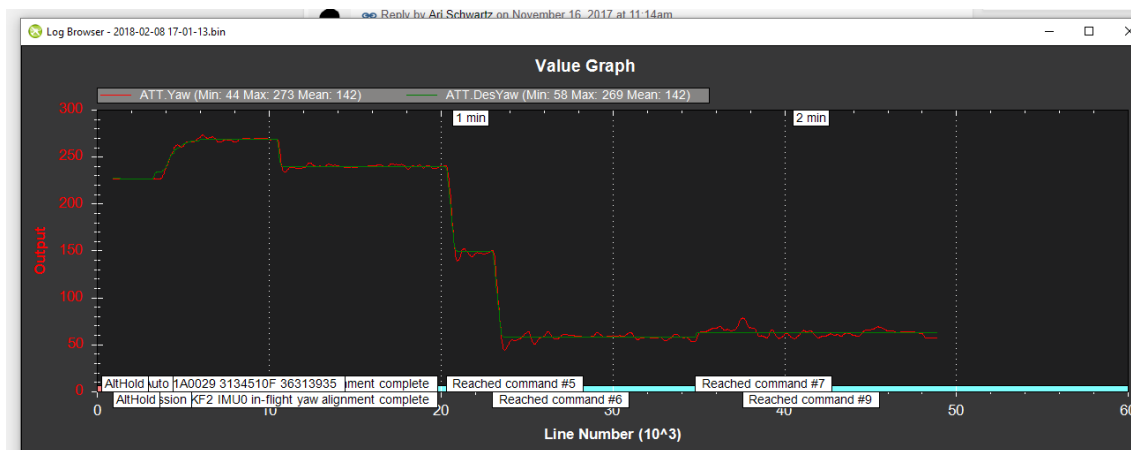


Ilustración 294 Log del parámetro ATT YAW para la misión 1. Fuente: Servidor interno LOCIS.

En la imagen anterior puede verse como la etiqueta “Reached command 6” o alcanzado el comando número 6 (inicio del sexto comando) corresponde con la inclinación de 50° (en el “Output”) llegando al resto de comandos con similar inclinación.

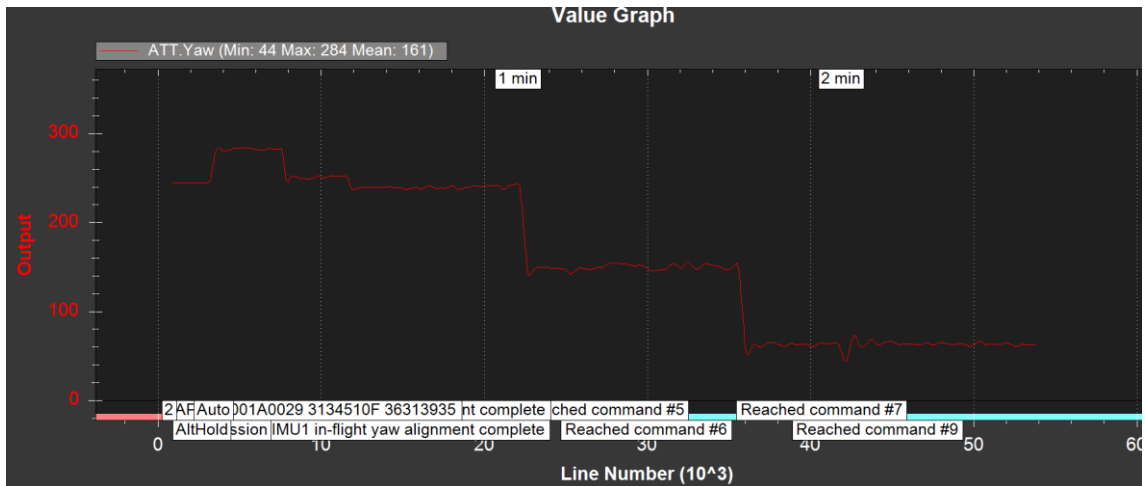


Ilustración 295 Log del parámetro ATT YAW para la misión 2. Fuente: Servidor interno LOCIS.

En este caso, puede verse como la etiqueta correspondiente al inicio del comando 6 corresponde con una inclinación de aproximadamente 160° , según el “Output”.

Para la corrección de este error se profundiza en el origen del dato analizado (YAW), este dato proviene del filtro KALMAN.

El filtro, según la documentación del desarrollador, utiliza datos procedentes de los giroscopios, acelerómetro, brújula, GPS y barómetro para estimar su posición y orientación.

Asimismo según la documentación encontrada en [ardupilot.org](http://ardupilot.org/copter/docs/auto-mode.html) (<http://ardupilot.org/copter/docs/auto-mode.html>) se establece que: “During the mission the pilot’s roll, pitch and throttle inputs are ignored but the yaw can be overridden with the yaw stick. This allows the pilot to for example aim the nose of the copter (which might have a hard mounted camera on it) as the copter flies the mission. The autopilot will attempt to retake yaw control as the vehicle passes the next waypoint.”

Este párrafo implica que durante una misión automática el “yaw” o guiñada puede ser sobre-escrito de forma manual usando el “stick” del radiocontrol.

De este modo, en caso de recibir un input del canal 4 del radio control (correspondiente a la guiñada), el modo AUTO deja la guiñada en manos del usuario hasta el siguiente waypoint.

Al revisar el log de la controladora, se observan pequeñas entradas de ínfimo valor que se sospecha que hayan sido percibidas como un intento de control manual, influyendo así en el comportamiento anómalo de la guiñada del dron en la misión.

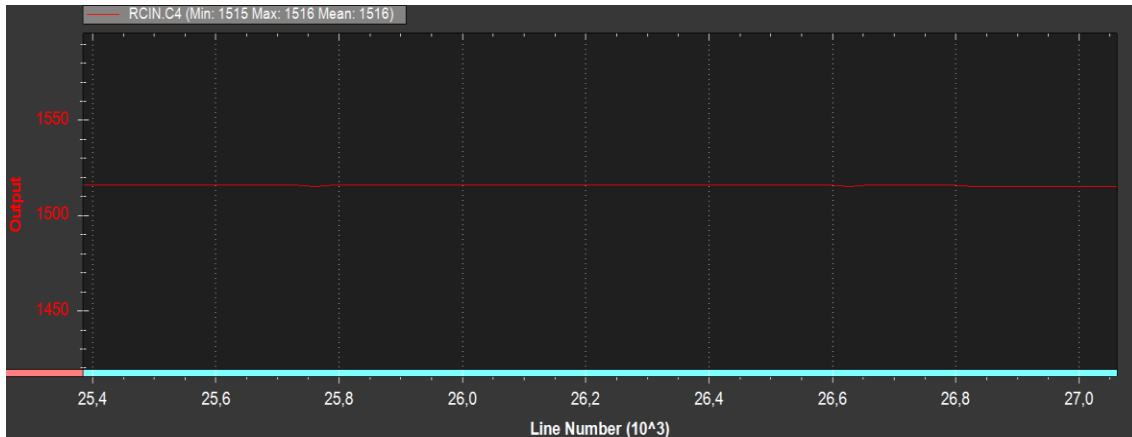


Ilustración 296 Gráfico de entrada del canal 4 mostrado a través del log de la controladora entre los puntos 6-7. Fuente: Servidor interno LOCIS.

13.2.10.- Inexistencia de fotos de la primera misión

Tras analizar el script que se ejecuta desde la Raspberry pi y que controla la cámara RGB, se puede ver como el método de guardado de las fotos incorpora un contador, del modo siguiente:

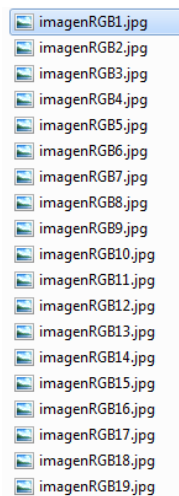


Ilustración 297 Ejemplo de guardado de fotos RGB por contador. Fuente: Servidor interno LOCIS.

Esto da lugar a que:

- Las fotos de la primera misión tendrán la misma notación que las de la segunda misión.
- Al guardar dos archivos con el mismo nombre en un mismo directorio el último sobre-escribe al primero.

Por esto solo es posible ver las fotos tomadas durante la segunda misión, porque las de la primera han sido sobre-escritas y por tanto borradas.

Nota: la parte referente a la programación del script se analizará más en profundidad en el apartado “Programación”.

13.2.11.- Otros comentarios

Se seguirán los siguientes pasos como medidas para subsana miento de errores:

- Es necesario cambiar el “script” para evitar el “pisado” de unas fotos con otras en misiones sucesivas.
- Asimismo se procede a anular el control manual del canal 4 de la emisora de radio control durante las misiones en modo automático para que no pueda causar interferencias con la guiñada del dron.
- También debe prestarse atención a los giros en puntos de trayectoria en posteriores pruebas, en caso de que volviera a repetirse el problema del “YAW”.

Por otro lado esta prueba desvela que:

- La velocidad de vuelo horizontal NO afecta a la cadencia de captura de imagen, como tampoco lo hace la variación en altura.
- La reducción de tiempo entre órdenes de disparo, a partir de 1 s, aumenta el número de imágenes RGB pero no de las imágenes TIR.
- La cámara FLIR VUE PRO tiene un tiempo medio propio (2-3s) para procesar la toma de la imagen, que en principio no es reducible por medios externos.

13.3.- PRUEBA DE VUELO EN CAMPO DE PRUEBAS LOCIS

13.3.1.- Resumen

Se realizan dos pruebas de vuelo simultáneas a 15 m de altura. Con una velocidad de vuelo durante la captura de imágenes de 1 m/s.

En esta prueba se espera que el dron realice las dos misiones de forma automática, incluyendo el despegue y el aterrizaje, ambas configuradas in situ.

13.3.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

13.3.3.- Fecha y lugar de realización

El día 22 de febrero de 2018 se realizó una prueba de vuelo ubicada en el campo de pruebas de Locis (prado de detrás de la oficina).

13.3.4.- Motivo de la prueba

Se pretende recrear en la medida de lo posible la primera prueba Roces en un entorno más cercano a la oficina, para con ello probar la capacidad del dron a la hora de captar imágenes térmicas y RGB.

Comprobar cómo se comporta el dron con la modificación en el archivo de la misión anteriormente usada, en este caso se introdujo un comando LAND.

13.3.5.- Misión por realizar

13.3.6.- Resultados de la prueba

13.3.6.1.- Captura de imágenes

Según el archivo log de ambas misiones registrado por la controladora de vuelo, el tiempo de captura de imagen, tiempo transcurrido entre la orden de inicio de captura de fotos y la de cese, fue de aproximadamente 85 segundos. Tiempo durante el cual se han obtenido:

- Primer vuelo:
 - XX imágenes RGB
 - 43 imágenes TIR
- Segundo vuelo:
 - XX imágenes RGB
 - 40 imágenes TIR

13.3.6.2.- Desarrollo del vuelo

El vuelo se desarrolla sin incidencias, orientándose el dron de forma correcta en cada tramo de la misión, y en ambas misiones.

13.3.6.3.- Transferencia de imágenes

13.3.6.4.- Imágenes de archivo

A continuación se muestra el dron sobre un arbusto, usado como punto de referencia de la zona de aterrizaje, seguido de dos imágenes ejemplo del vuelo, una RGB y otra TIR.



Ilustración 298 Dron a punto de despegar en el campo de pruebas LOCIS. Fuente: Servidor Interno LOCIS.

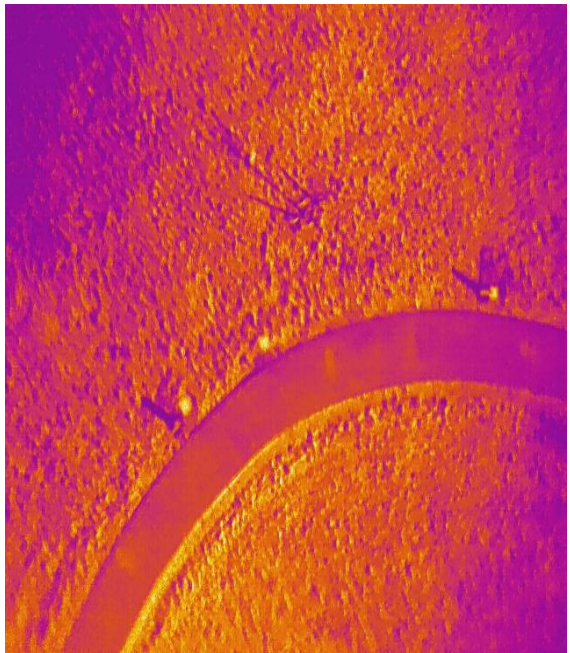


Ilustración 299 Campo de pruebas a vista de dron (RGB + térmica). Fuente: Servidor interno LOCIS.

13.3.7.- Conclusiones

13.3.8.- Otros comentarios

13.4.- PRUEBA DE VUELO ROCES 2

13.4.1.- Resumen

Se realizaron dos vuelos sobre paneles fotovoltaicos a diferentes alturas, 13 m y 26m. Con una velocidad de vuelo durante la captura de imágenes de 1 m/s, esta prueba es una réplica de Roces 1.

En esta prueba se espera que el dron realice las dos misiones de forma automática, incluyendo el despegue y el aterrizaje, ambas configuradas in situ.

13.4.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

13.4.3.- Fecha y lugar de realización

El día 7 de marzo de 2018 se realizó una prueba de vuelo ubicada en el campo de fútbol del TSK Roces, equipo patrocinado por TSK (empresa colaboradora en el proyecto UAV-Inspection).

13.4.4.- Motivo de la prueba

Los objetivos principales de la prueba consistían en:

- Realizar dos misiones de vuelo seguidas, situadas en la misma localización y con Waypoints de vuelo similares, solo variando la altura de uno de los Waypoints en el segundo vuelo. Para así poder comprobar la marca temporal de las fotos RGB, evitando que se sobre-escribieran como pasaba antes cuando se diferenciaban por medio de un contador.
- Observar los giros realizados al pasar por los vértices del cuadrilátero que se le especificaba realizar en la misión.
- Observar la desviación en el momento del aterrizaje con respecto al último waypoint especificado, dicha desviación se produce por el margen de error del GPS.
- Observar la transferencia automática de imágenes térmicas desde la memoria de la cámara a la carpeta compartida de la Raspberry Pi.

13.4.5.- Misión por realizar

Se espera que el dron realice un recorrido en forma de rectángulo sobre las placas solares ubicadas en el campo de fútbol del equipo TSK Roces.

Para ello es necesario especificar una sucesión de Waypoints como se aprecia a continuación:

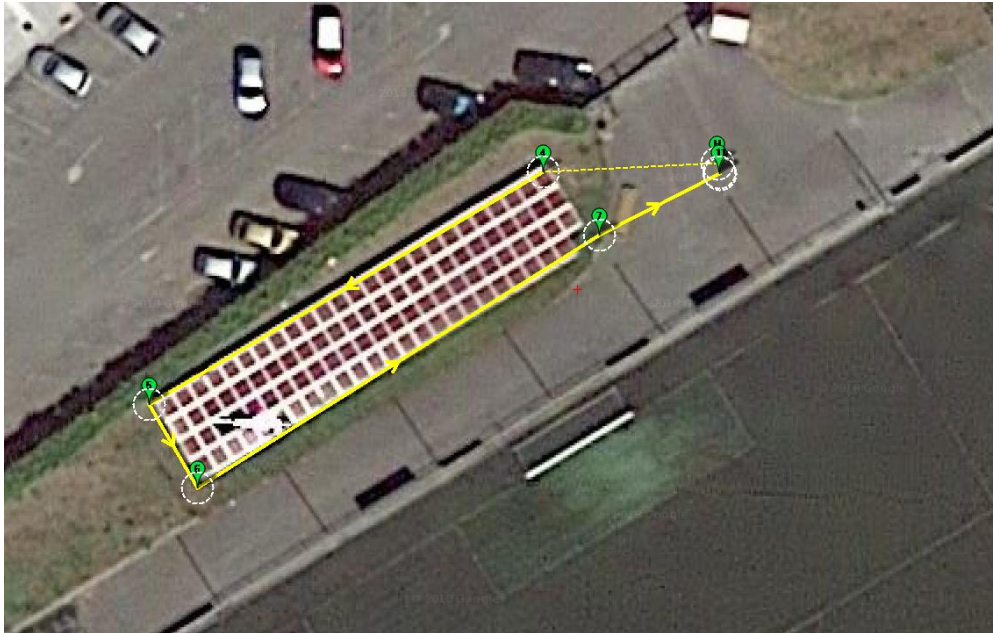


Ilustración 300 Misión a realizar durante la prueba. Fuente: Servidor interno LOCIS

Waypoints															
Radio WP	Radio	Perder	Tiempo	por	Defecto	Terrain	Verify Height	Agregar	Alt Wam	Spline					
1	80	100						Agregar	0						
	Comandos	Dela				Lat	Long	Alt	Borrar	Amb	Abajo	Grad %	Angle	Dist	AZ
1	TAKEOFF	0	0	0	0	43,519974	-5,675825	13	X			0	0	0	0
2	DO_CHANGE_SPEED	1	1	0	0	0	0	0	X			0	0	0	0
3	DO_SET_RELAY	0	1	0	0	0	0	0	X			0	0	0	0
4	WAYPOINT	0	0	0	0	43,519964	-5,67598	13	X			-168,0	-59,2	18,8	261
5	WAYPOINT	0	0	0	0	43,519835	-5,676281	13	X			0,0	0,0	28,2	239
6	WAYPOINT	0	0	0	0	43,519789	-5,676244	13	X			0,0	0,0	5,9	150
7	WAYPOINT	0	0	0	0	43,519929	-5,675937	13	X			0,0	0,0	29,2	58
8	DO_SET_RELAY	0	0	0	0	0	0	0	X			0	0	0	0
9	DO_SET_RELAY	1	1	0	0	43,523433	-5,610743	1	X			0	0	0	0
10	WAYPOINT	0	0	0	0	43,519963	-5,675845	13	X			0,0	0,0	8,3	63
11	LAND	0	0	0	0	43,519964	-5,675846	13	X			0,0	0,0	0,1	324

Ilustración 301 Waypoints o correspondientes a la prueba. Fuente: Servidor interno Locis. En donde se realiza la siguiente sucesión:

- Comando 1: Despegue (TAKEOFF) y ascensión a 13m de altura.
- Comando 2: Cambio de velocidad a 1m/s (DO_CHANGE_SPEED).
- Comando 3: Establecer el voltaje del pin del primer relé como 1, correspondiente con “On” y con 3,3 V en PIXHAWCK (DO_SET_RELAY).
- Comando 4: Coordenadas del primer waypoint (punto 4), correspondiente con el primer vértice del rectángulo.
- Comando 5: Coordenadas del segundo waypoint (punto 5), correspondiente con el segundo vértice del rectángulo.

- Comando 6: Coordenadas del tercer waypoint (punto 6), correspondiente con el tercer vértice del rectángulo.
- Comando 7: Coordenadas del cuarto waypoint (punto 7), correspondiente con el cuarto vértice del rectángulo.
- Comando 8: Establecer el voltaje del pin del primer relé como 0, correspondiente con “Off” y con 0V en PIXHAWCK (DO_SET_RELAY).
- Comando 9: Establecer el voltaje del pin del segundo relé como 1, correspondiente con “On” y con 3,3V en PIXHAWCK (DE_SET_RELAY).
- Comando 10: Coordenadas del último waypoint (punto 10), correspondiente con el punto de aterrizaje.
- Comando 11: Orden de aterrizaje (LAND).

Para la segunda misión realizada justo después de la primera solamente es necesario cambiar la altura especificada en cada uno de los puntos implicados de 13m a 26m.

Los archivos .txt que detallan la misión para cada uno de los casos son:

WGC	WPL	110										
0	1	0	16	0	0	0	0	43.519977	-5.675862	29.169705	1	
1	0	10	22	0.000000	0.000000	0.000000	0.000000	43.519974	-5.675825	13.000000	1	
2	0	10	178	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1	
3	0	10	181	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1	
4	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519964	-5.675980	13.000000	1	
5	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519835	-5.676281	13.000000	1	
6	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519789	-5.676244	13.000000	1	
7	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519929	-5.675937	13.000000	1	
8	0	10	181	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1	
9	0	10	181	1.000000	1.000000	0.000000	0.000000	43.523433	-5.610743	1.000000	1	
10	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519963	-5.675845	13.000000	1	
11	0	10	21	0.000000	0.000000	0.000000	0.000000	43.519964	-5.675846	13.000000	1	

Ilustración 302 Archivo .txt con los datos de la primera misión de vuelo. Fuente: Servidor interno LOCIS.

Archivo	Edición	Formato	Ver	Ayuda								
WGC	WPL	110										
0	1	0	16	0	0	0	0	43.519969	-5.675847	29.124436	1	
1	0	10	22	0.000000	0.000000	0.000000	0.000000	43.519974	-5.675825	26.000000	1	
2	0	10	178	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1	
3	0	10	181	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1	
4	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519964	-5.675980	26.000000	1	
5	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519835	-5.676281	26.000000	1	
6	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519789	-5.676244	26.000000	1	
7	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519929	-5.675937	26.000000	1	
8	0	10	181	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1	
9	0	10	181	1.000000	1.000000	0.000000	0.000000	43.523433	-5.610743	40.000000	1	
10	0	10	16	0.000000	0.000000	0.000000	0.000000	43.519969	-5.675846	26.000000	1	
11	0	10	21	0.000000	0.000000	0.000000	0.000000	43.519969	-5.675847	26.000000	1	

Ilustración 303 Archivo .txt con los datos de la segunda misión de vuelo. Fuente: Servidor interno LOCIS.

13.4.6.- Resultados de la prueba

13.4.6.1.- Captura de imágenes

Según el archivo log de ambas misiones registrado por la controladora de vuelo, el tiempo de captura de imagen, tiempo transcurrido entre la orden de inicio de captura de fotos y la de cese, fue de aproximadamente 100 segundos. Tiempo durante el cual se han obtenido:

- Primer vuelo (correspondiente a altura de 13m):
 - 79 imágenes RGB

- 39 imágenes TIR
- Segundo vuelo (correspondiente a altura de 26m):
 - 78 imágenes RGB
 - 42 imágenes TIR

13.4.6.2.- Desarrollo del vuelo

El dron realizó las dos misiones de vuelo siendo capaz de realizar el despegue y aterrizaje de forma autónoma en ambas, tampoco tuvo lugar ninguna incidencia.

No se apreció comportamiento anómalo en los giros de los puntos de paso, realizándolos de forma suave.

En cuanto a la desviación en el aterrizaje se aprecia un error de 1 m, desde el lugar de aterrizaje teórico al real. Esta medida es aceptable y se encuentra dentro del margen de error especificado por el fabricante del GPS.

13.4.6.3.- Transferencia de imágenes

La transferencia de imágenes térmicas se realiza de forma exitosa al recibir el comando de final de misión, correspondiente con el comando 9, solo para una de las dos misiones de vuelo. Por lo que será conveniente profundizar en este detalle.

Por el contrario, las imágenes RGB aparecen guardadas en el lugar correcto.

Además, se observa como la marca temporal de las fotografías RGB analizadas en la estación de tierra es la correcta, como se aprecia en la siguiente imagen:

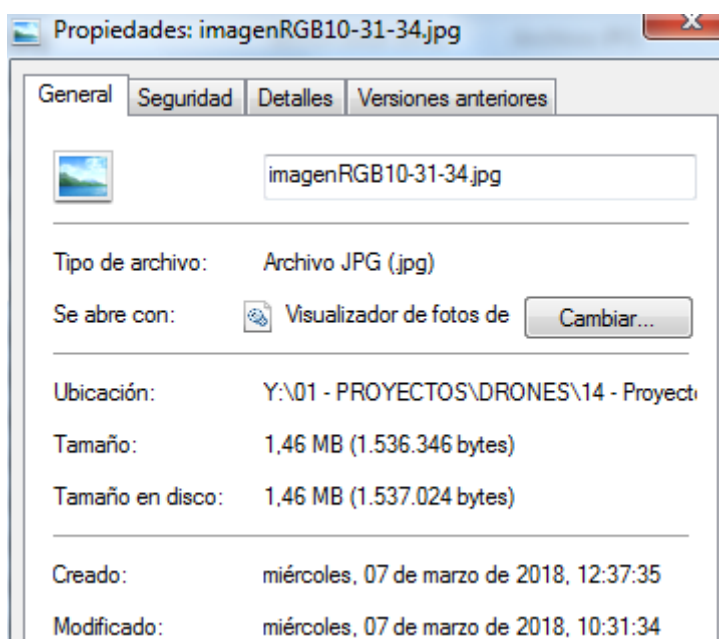


Ilustración 304 Ejemplo propiedades imagen durante la prueba. Fuente: Servidor interno LOCIS.

Podemos apreciar el nombre de la imagen, que sigue un patrón horario de 24 h, del modo hora-minuto-segundo (10-31-34) correspondiente a las diez y media de la mañana aproximadamente.

13.4.6.4.- Imágenes de archivo

A continuación se muestran unas imágenes tomadas durante la realización de la prueba.



Ilustración 305 Dron antes del despegue. Fuente: Servidor interno LOCIS.



Ilustración 306 Dron durante el vuelo. Fuente: Servidor interno LOCIS.

13.4.7.- Conclusiones

Los resultados obtenidos en prueba subsanan los errores cometidos durante la primera prueba Rocés, véase:

- Borrado de fotos de vuelos anteriores al último en sucesivas misiones si no se descargaban.
- Comportamiento anómalo durante los giros.
- Se producen más imágenes RGB al ajustar la cadencia de orden de disparo.

Por otro lado, se observa un problema nuevo:

- No volcado de fotos térmicas en una de las misiones.

13.4.8.- Otros comentarios

Se considera importante seguir investigando el tiempo especificado entre diferentes órdenes de disparo, aunque en principio se considera aceptable pues permite la creación de modelos 3D del entorno a monitorizar.

Además, se comprobó como las imágenes térmicas de la primera misión se encontraban en la carpeta compartida de la Raspberry (como era de esperar), mientras que las de la segunda misión se encontraban en la memoria de la propia cámara FLIR TAU PRO (donde las guarda). Por lo que se confirma un error en la transferencia (volcado de imágenes)

Al comprobar el script de la misión, se puede ver como la transferencia de imágenes chequea primero la conexión a internet antes de realizar el volcado.

Por esto, y debido al entorno en el cual se produjo la prueba pudo haberse dado el caso de que en una misión la Raspberry encontrase una red Wi-Fi a la cual conectarse y por tanto realizar el volcado y en la otra no.

Siendo esta una posible razón que explicara el problema decide cambiarse el chequeo de conexión antes de realizar el volcado de imágenes a la carpeta compartida dentro de la Raspberry Pi.

13.5.- PRUEBA DE VUELO TRANSFERENCIA TIR Y RTK

13.5.1.- Resumen

Se realizaron dos vuelos sobre el campo de pruebas de LOCIS, a 26m (a excepción de un salto). Con una velocidad de vuelo durante la captura de imágenes de 1 m/s.

En esta prueba se espera que el dron realice las dos misiones de forma automática, incluyendo el despegue y el aterrizaje, ambas configuradas in situ.

Además se pretende que la transferencia de imágenes térmicas se realice para cada una de las misiones, no solo para una como paso en la prueba anterior.

13.5.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

13.5.3.- Fecha y lugar de realización

El día 16 de marzo de 2018 se realizó una prueba de vuelo ubicada en el campo de pruebas LOCIS.

13.5.4.- Motivo de la prueba

La prueba consta de un objetivo dual, del tipo:

- Probar la transferencia de imágenes térmicas con un nuevo script que no requiere el chequeo de conexión a internet para realizar el volcado de imágenes desde la memoria de la cámara FLIR TAU PRO a la carpeta compartida de la Raspberry con la estación de tierra.
- Después de haber realizado una sucesión de pruebas de campo con el dispositivo RTK se pretende realizar una primera prueba de vuelo usando el RTK (mirar apartado de la memoria referente a RTK para más información).

13.5.5.- Misión por realizar

Se espera que el dron realice un recorrido poligonal sobre el campo de pruebas LOCIS.

Para ello es necesario especificar una sucesión de Waypoints como se aprecia a continuación:

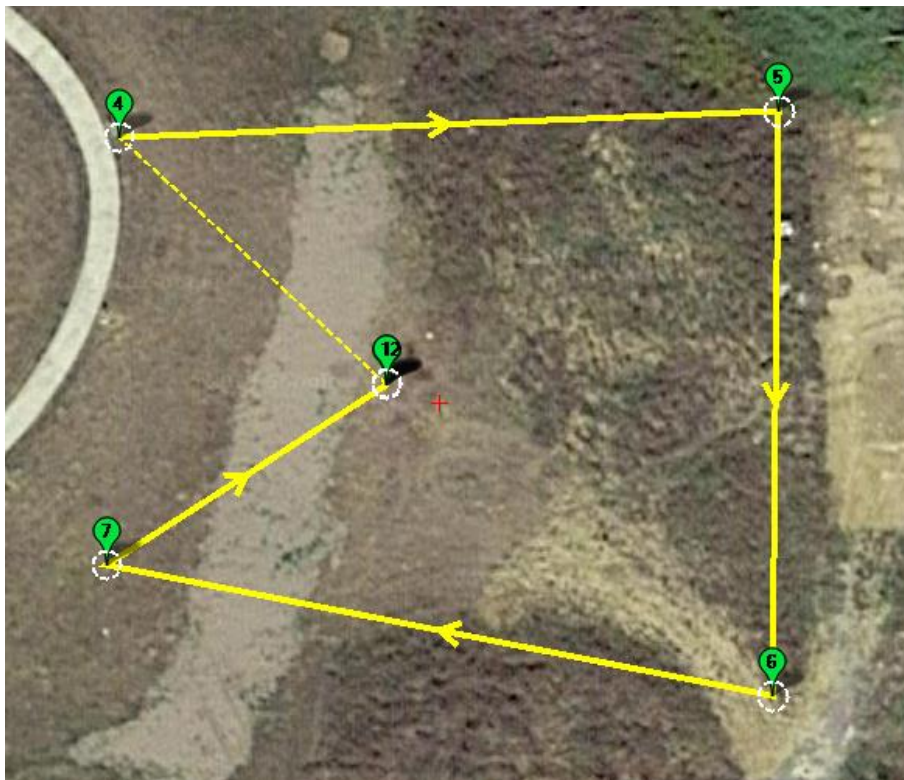


Ilustración 307 Primera misión ubicada en el campo de pruebas LOCIS. Fuente: Servidor interno LOCIS.

Comandos	Dela				Lat	Long	Alt	Borrar	Arrib	Abajo	Grad %	Angle	Dist	AZ
1 TAKEOFF	0	0	0	0	43,519974	-5,675825	26	X	🏠	🏠	0	0	0	0
2 DO_CHANGE_SPEED	1	1	0	0	0	0	0	X	🏠	🏠	0	0	0	0
3 DO_SET_RELAY	0	1	0	0	0	0	0	X	🏠	🏠	0	0	0	0
4 WAYPOINT	0	0	0	0	43,523277	-5,609802	26	X	🏠	🏠	-27,5	-15,4	26,3	313
5 WAYPOINT	0	0	0	0	43,523294	-5,609234	26	X	🏠	🏠	0,0	0,0	45,8	88
6 WAYPOINT	0	0	0	0	43,522928	-5,609239	26	X	🏠	🏠	0,0	0,0	40,7	181
7 WAYPOINT	0	0	0	0	43,52301	-5,609813	26	X	🏠	🏠	0,0	0,0	47,2	281
8 DO_SET_RELAY	0	0	0	0	0	0	0	X	🏠	🏠	0	0	0	0
9 DO_SET_RELAY	1	1	0	0	0	0	0	X	🏠	🏠	0	0	0	0
10 WAYPOINT	0	0	0	0	43,523122	-5,609572	26	X	🏠	🏠	0,0	0,0	23,1	57
11 WAYPOINT	5	0	0	0	43,523123	-5,609572	10	X	🏠	🏠	-14389,1	-89,6	16,0	0
12 LAND	0	0	0	0	43,523123	-5,609572	26	X	🏠	🏠	Infinito	90,0	16,0	180

Ilustración 308 Waypoints correspondientes a la primera misión. Fuente: Servidor interno LOCIS. En donde se realiza la siguiente sucesión:

- Comando 1: Despegue (TAKEOFF) y ascensión a 13m de altura.
- Comando 2: Cambio de velocidad a 1m/s (DO_CHANGE_SPEED).
- Comando 3: Establecer el voltaje del pin del primer relé como 1, correspondiente con “On” y con 3,3 V en PIXHAWCK (DO_SET_RELAY)
- Comando 4: Coordenadas del primer waypoint (punto 4)
- Comando 5: Coordenadas del segundo waypoint (punto 5)
- Comando 6: Coordenadas del tercer waypoint (punto 6)
- Comando 7: Coordenadas del cuarto waypoint (punto 7)
- Comando 8: Establecer el voltaje del pin del primer relé como 0, correspondiente con “Off” y con 0V en PIXHAWCK (DO_SET_RELAY).
- Comando 9: Establecer el voltaje del pin del segundo relé como 1, correspondiente con “On” y con 3,3V en PIXHAWCK (DE_SET_RELAY).
- Comando 10: Coordenadas del quinto waypoint (punto 12)
- Comando 11: Coordenadas del quinto waypoint (punto 12), en este caso se ordena mantener una altura de 10m durante 5 segundos.
- Comando 12: “Land” o despegue en la misma posición.

La segunda misión de la prueba es similar, pero quitando el comando 11 de la anterior.

	Comandos	Dela			Lat	Long	Alt	Borrar	Amb	Abajo	Grad %	Angle	Dist	AZ
1	TAKEOFF	0	0	0	43,519974	-5,675825	26	X	🏠	🏠	0	0	0	0
2	DO_CHANGE_SPEED	1	1	0	0	0	0	X	🏠	🏠	0	0	0	0
3	DO_SET_RELAY	0	1	0	0	0	0	X	🏠	🏠	0	0	0	0
4	WAYPOINT	0	0	0	43,523277	-5,609802	26	X	🏠	🏠	-27,5	-15,4	26,3	313
5	WAYPOINT	0	0	0	43,523294	-5,609234	26	X	🏠	🏠	0,0	0,0	45,8	88
6	WAYPOINT	0	0	0	43,522928	-5,609239	26	X	🏠	🏠	0,0	0,0	40,7	181
7	WAYPOINT	0	0	0	43,52301	-5,609813	26	X	🏠	🏠	0,0	0,0	47,2	281
8	DO_SET_RELAY	0	0	0	0	0	0	X	🏠	🏠	0	0	0	0
9	DO_SET_RELAY	1	1	0	43,523433	-5,610743	40	X	🏠	🏠	0	0	0	0
10	WAYPOINT	0	0	0	43,523122	-5,609572	26	X	🏠	🏠	0,0	0,0	23,1	57
11	LAND	0	0	0	43,523123	-5,609572	26	X	🏠	🏠	0,0	0,0	0,1	0

Ilustración 309 Waypoints correspondientes a la primera misión. Fuente: Servidor interno LOCIS.

13.5.6.- Resultados de la prueba

13.5.6.1.- Captura de imágenes

Según el archivo log de ambas misiones registrado por la controladora de vuelo, el tiempo de captura de imagen, tiempo transcurrido entre la orden de inicio de captura de fotos y la de cese, fue de aproximadamente 95 segundos. Tiempo durante el cual se han obtenido:

- Primer vuelo:
 - 91 imágenes RGB
 - 42 imágenes TIR
- Segundo vuelo:
 - 93 imágenes RGB
 - 45 imágenes TIR

13.5.6.2.- Desarrollo del vuelo

El dron realizó las dos misiones de vuelo siendo capaz de realizar el despegue y aterrizaje de forma autónoma en ambas, tampoco tuvo lugar ninguna incidencia.

13.5.6.3.- Transferencia de imágenes

Tanto las imágenes térmicas como las RGB se encontraban en la carpeta compartida de la Raspberry en el momento de revisar en la oficina, de modo que podemos concluir que el envío fue exitoso.

13.5.6.4.- Imágenes de archivo



Ilustración 310 Imagen RGB de la prueba.
Fuente: Servidor interno LOCIS.

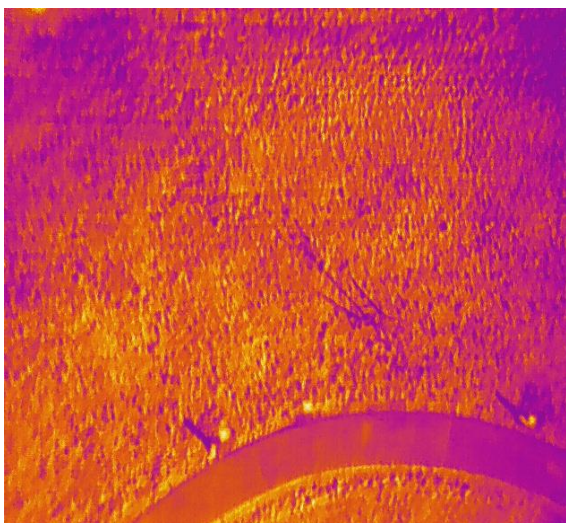


Ilustración 311 Imagen térmica de la prueba.
Fuente: Servidor interno LOCIS.

13.5.7.- Conclusiones

Tanto el resultado de la prueba como el desarrollo de la misión puede calificarse como exitoso, en principio el problema de la no transferencia de imágenes TIR en una de las misiones parece no haberse repetido.

De este modo se entiende el problema como solucionado, debiéndose aun así extremar las precauciones por si volviera a repetirse.

13.5.8.- Otros comentarios

Fue imposible realizar la prueba del dispositivo RTK, pues el módulo usado como BASE fue incapaz de realizar la conexión con el punto de acceso Wi-Fi proporcionado por un terminal móvil en modo “Hotspot” (para más información mirar el apartado de la memoria referido a RTK).

Por lo tanto se apunta la prueba del dispositivo RTK en vuelo como pendiente.

13.6.- PRUEBA DE VUELO AZOTEA TSK

13.6.1.- Resumen

Se realizaron vuelos sobre la azotea del tejado de TSK (empresa colaboradora), a 26m. Con una velocidad de vuelo durante la captura de imágenes de 1 m/s.

En esta prueba se espera que el dron realice las dos misiones de forma automática, incluyendo el despegue y el aterrizaje, ambas configuradas in situ.

13.6.2.- Proyecto al que corresponde la prueba

Esta prueba se realiza dentro del contexto del proyecto UAV-Inspection.

13.6.3.- Fecha y lugar de realización

El día 19 de abril de 2018 se realizó una prueba de vuelo ubicada en la azotea del edificio de oficinas de la empresa TSK.

13.6.4.- Motivo de la prueba

La principal motivación de esta prueba consistía en un estudio exhaustivo del comportamiento de lo que se conoce como un “falso positivo”

Para ello fue necesario comprobar la incidencia del sol a diferentes horas sobre las placas solares de la azotea del edificio a estudiar, estableciendo así una comparativa entre la frecuencia de aparición de falsas alarmas de puntos calientes en relación con los puntos calientes reales.

Nota: El término “punto caliente” hace referencia al sombreado de una célula fotovoltaica, cuando este fenómeno ocurre dicha célula se comportará como una resistencia óhmica pudiendo calentarse hasta el punto de destruirse. En este contexto un falso positivo se produce cuando un reflejo del sol sobre la placa induce, al verlo desde la cámara del dron, a pensar que se trata de un punto caliente cuando en realidad no es así.

Secundariamente, los sucesivos vuelos constituyen una prueba de funcionamiento del LED instalado, que en caso de señal de disparo se ilumina a intervalos regulares de tiempo.

13.6.5.- Misión por realizar

Se realizaron un total de seis vuelos a lo largo del día (uno fue una repetición de otro), usando para ello el mismo archivo de misión:

Se espera que el dron realice un recorrido poligonal empezando en el prado adyacente al edificio, para luego volar sobre cada una de las tres hileras de placas solares ubicadas en la azotea, en un recorrido del modo siguiente:

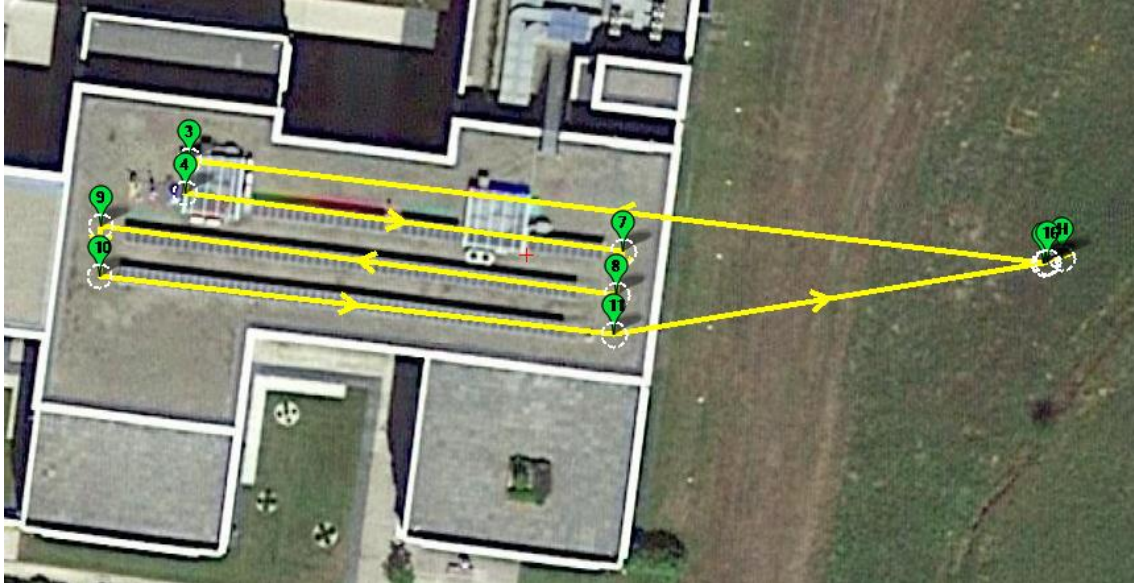


Ilustración 312 Misión inspección de paneles solares en la azotea de edificio. Fuente: Servidor interno LOCIS.

En donde se realiza la siguiente sucesión:

- Comando 1: Despegue (TAKEOFF) y ascensión a 26m de altura (un par de metros por debajo a partir del vuelo 1).
- Comando 2: Coordenadas del primer waypoint (punto 2)
- Comando 3: Coordenadas del segundo waypoint (punto 3)
- Comando 4: Coordenadas del tercer waypoint (punto 4)
- Comando 5: Cambio de velocidad a 1m/s (DO_CHANGE_SPEED).
- Comando 6: Establecer el voltaje del pin del primer relé como 1, correspondiente con “On” y con 3,3 V en PIXHAWCK (DO_SET_RELAY)
- Comando 7: Coordenadas de waypoint (punto 7)
- Comando 8: Coordenadas de waypoint (punto 8)
- Comando 9: Coordenadas del waypoint (punto 9)
- Comando 10: Coordenadas del waypoint (punto 10)
- Comando 11: Coordenadas del waypoint (punto 11)
- Comando 12: Establecer el voltaje del pin del primer relé como 0, correspondiente con “Off” y con 0V en PIXHAWCK (DO_SET_RELAY).
- Comando 13: Establecer el voltaje del pin del segundo relé como 1, correspondiente con “On” y con 3,3V en PIXHAWCK (DE_SET_RELAY).
- Comando 14: Cambio de velocidad a 5m/s (DO_CHANGE_SPEED)
- Comando 15: Coordenadas de waypoint (punto 15).
- Comando 16: “Land” o aterrizaje en la misma posición.

13.6.6.- Resultados de la prueba

13.6.6.1.- Captura de imágenes

Según el archivo log registrado por la controladora de vuelo, se especificará para cada una de las sucesivas misiones el tiempo de captura de imagen, tiempo transcurrido entre la orden de inicio de captura de fotos y la de cese. Las especificaciones de cada uno de los vuelos son:

- Primer vuelo:
 - Tiempo de captura de imagen:
 - 110 imágenes RGB
 - 56 imágenes TIR
- Segundo vuelo:
 - Tiempo de captura de imagen:
 - 107 imágenes RGB
 - 63 imágenes TIR
- Tercer vuelo:
 - Tiempo de captura de imagen:
 - 120 imágenes RGB
 - 70 imágenes TIR
- Cuarto vuelo:
 - El cuarto vuelo tuvo que repetirse, pues la primera vez no se encontraron imágenes térmicas ni en la propia cámara ni en la Raspberry.
 - Tiempo de captura de imagen:
 - 111 imágenes RGB
 - 59 imágenes TIR
- Quinto vuelo:
 - Tiempo de captura de imagen:
 - 95 imágenes RGB
 - 47 imágenes TIR

13.6.6.2.- Desarrollo del vuelo

Los principales problemas a la hora del desarrollo del vuelo fueron:

- La altura del edificio era de aproximadamente 13 metros, con una antena de 4 metros como punto más alto. Dicha antena era, además uno de los puntos de giro en las misiones del dron.
- La altura de vuelo del dron era primero de 26 metros, bajándose después un par de metros.
- Fue necesaria una solución de compromiso entre la seguridad y la cercanía a la hora de sacar fotos.

Para solucionar la incertidumbre se optó por vigilar el vuelo del dron desde la azotea del edificio, para en caso de que se acercase de forma peligrosa a la antenna detener el vuelo automático y retomar el control manual.

Además a partir de la segunda misión el responsable del pos-procesado de fotos pide bajar la altura de vuelo del dron, para poder mejorar la calidad de las fotos tomadas. Lo que hace necesario, también para la segunda misión, el control visual.



Ilustración 313 Control visual desde el tejado de la azotea (vista desde el dron). Fuente: Servidor interno LOCIS.

A pesar de las preocupaciones iniciales todos los vuelos transcurrieron sin incidentes.

13.6.6.3.- Transferencia de imágenes

Debido al elevado número de fotos la transferencia RGB de la carpeta compartida desde la Raspberry a la carpeta compartida vía Syncthing se realizó de forma extremadamente lenta, pero aun así sin errores.

La transferencia de imágenes de la cámara FLIR no se produjo en ninguna de las misiones. Este punto requiere de análisis posterior.

13.6.6.4.- Imágenes de archivo



Ilustración 314 Puesta a punto de los equipos. Fuente: Servidor interno LOCIS.

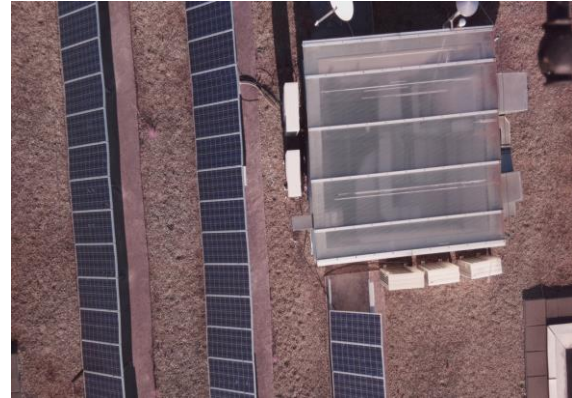


Ilustración 315 Azotea a vista de dron. Fuente: Servidor interno LOCIS

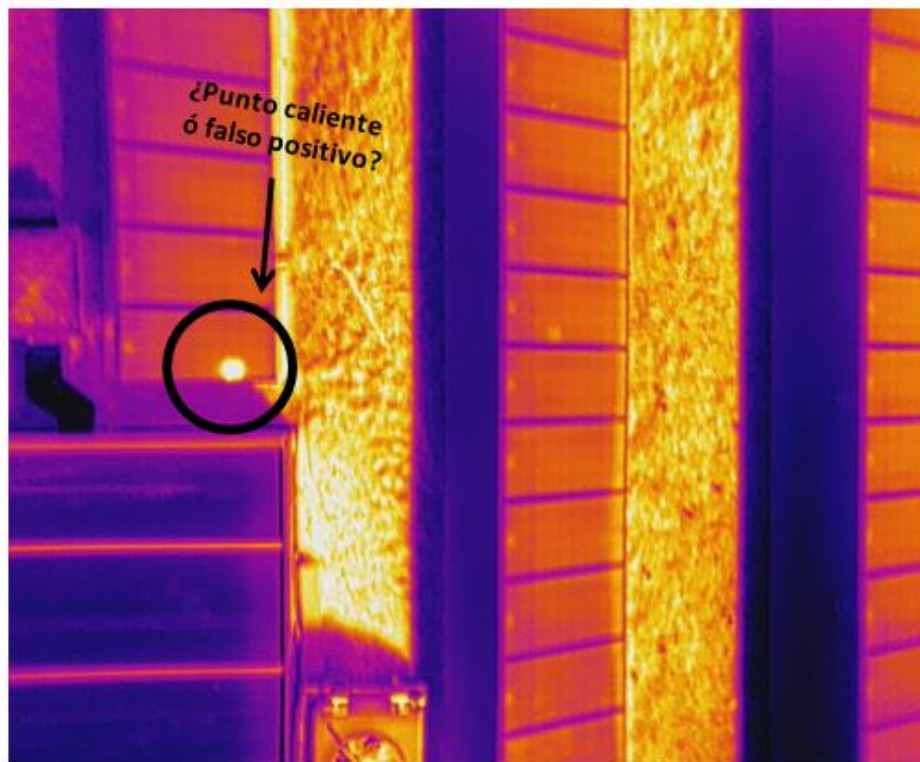


Ilustración 316 Imagen térmica de uno de los vuelos, con detalle de un posible punto caliente. Fuente: Servidor interno LOCIS.

Para poder diferenciar entre puntos calientes y falsos positivos es necesario observar la anomalía en cuestión a lo largo de varias horas. Un punto caliente se mantendrá al paso de las sucesivas misiones en el mismo punto, mientras que un falso positivo no.


```

total 89
drwx----- 2 root root 4096 abr 19 15:21 9016-4EF8
drwx----- 4 pi pi 65536 ene 1 1970 9016-4EF81
drwx----- 2 root root 4096 mar 30 2017 SETTINGS
drwx----- 2 root root 4096 may 3 2017 SETTINGS1
drwx----- 2 root root 4096 ene 30 16:07 SETTINGS2
drwx----- 2 root root 4096 mar 15 17:23 SETTINGS3
drwx----- 2 root root 4096 mar 21 15:29 SETTINGS4
drwxr-xr-x 3 root root 1024 ene 1 1970 SETTINGS5
root@raspi_locis:/media/pi# ls -l
total 25
drwx----- 2 root root 4096 abr 19 15:21 9016-4EF8
drwx----- 2 root root 4096 mar 30 2017 SETTINGS
drwx----- 2 root root 4096 may 3 2017 SETTINGS1
drwx----- 2 root root 4096 ene 30 16:07 SETTINGS2
drwx----- 2 root root 4096 mar 15 17:23 SETTINGS3
drwx----- 2 root root 4096 mar 21 15:29 SETTINGS4
drwxr-xr-x 3 root root 1024 ene 1 1970 SETTINGS5
root@raspi_locis:/media/pi# cd 9016-4EF8
root@raspi_locis:/media/pi/9016-4EF8# ls -l
total 0

```

Ilustración 319 Comando ls -l para visionado por consola. Fuente: Elaboración propia.

Asimismo, en la última imagen podemos ver como no se detectan fotos, aunque sí que se sacaron (confirmación acústica del sonido y confirmación conectando la cámara al PC vía USB).

Las fotos térmicas guardadas en la ruta root no son detectables ni por consola ni por script, pero sí que se ven interactuando cámara-PC.

Solo son accesibles desde las otras dos rutas (las de propiedad pi) por medio de consola o script, que del mismo modo hacen referencia a la cámara FLIR.

Ante el desconocimiento del comportamiento errático de la cámara (o mejor dicho de la ruta que la Raspberry otorga a la cámara) y a falta de una mejor solución se optará por buscar las fotos en todas las rutas que se mostraron anteriormente.

El problema anteriormente descrito sería capaz de explicar tanto el no volcado de fotos térmicas a la Raspberry, como también la ausencia de fotos en una de las misiones.

14. UAV-Torres de tensión

14.1.- SELECCIÓN DE COMPONENTES

14.1.1.- Sensor electromagnético

Se procede a realizar la selección del sensor electromagnético (uno de los componentes del módulo multi-sensor), para ello se realizarán cinco propuestas de las cuales se seleccionará una.

La selección se realizará en base a una serie de parámetros clave:

- Existencia de software adicional: Mayormente hace referencia a la existencia de algún tipo de software de pos-procesado de datos. De existir también se debe tener en cuenta para que sistemas operativos es válido, si es de pago o no, y la sencillez de uso entre otros.
- Rango de frecuencias de trabajo: El campo electromagnético se producirá mayormente en la frecuencia de 50-60 Hz que es la frecuencia de trabajo de los sistemas de potencia en Europa. Lógicamente cuanto mayor rango mejor, pero se debe buscar una solución de compromiso entre el rango de frecuencia y las demás especificaciones.
- Precisión/Tolerancia: Ajuste completo o fidelidad de un dato o medida que es capaz de aportar cada uno de los sensores.
- Rango dinámico de campo eléctrico: Margen que existe entre los niveles de medida superior e inferior.
- Máximo campo eléctrico medible
- Rango dinámico de campo magnético
- Máximo campo magnético medible
- Componentes que es capaz de medir el campo eléctrico: En referencia a la tri-dimensionalización del espacio de medida.
- Componentes que es capaz de medir el campo magnético
- Modos de medida: Principalmente especifica el tipo de output que recibiremos, como por ejemplo valor medio o valor de pico entre otros.
- Velocidad de adquisición: Número de medidas que es capaz de realizar el sensor por segundo, para el caso de sensores digitales.

- Almacenamiento de datos: En este caso es importante saber si la sonda permite almacenaje de datos internos, y en caso de permitirse la cantidad de memoria disponible.
- Interfaz de salida
- Alimentación: Principalmente importante la existencia de batería, y en caso de tener su peso, dimensiones y autonomía. Debe tenerse en cuenta una autonomía mínima ligeramente superior al tiempo de vuelo (determinado por la batería del dron y estimado en 20 minutos)
- Peso: Es un factor clave en la elección de cualquier componente que se acople al dron ya que un peso elevado podría limitar el tiempo de vuelo en gran medida, o en el peor de los casos impedir el despegue.
- Precio: Al contrario de lo que se podría pensar no es de los factores más limitantes, aunque es importante. Esto es debido a que para el éxito del proyecto se exigirá unas prestaciones mínimas al sensor, que jugaran un papel más decisivo en la elección.
- Otras especificaciones

Las cinco sondas propuestas por el área TSC-UNIOVI son las siguientes:

- FM10LS
- NFA1000
- NFA400
- NFA30M
- MC910

Se realizará un análisis de cada una de ellas para poder escoger la que mejor se ajuste a las demandas del proyecto.

14.1.1.1.- FM10LS



Ilustración 320 Sensor FM10LS. Fuente: <https://www.vitalitools.nl/fauser-fm10ls>

Nombre	FM10LS
Software	FM-Data
Rango frec	10Hz-400Hz
Precisión	<5% Hz campo B
	<10% Hz campo E
Rango dinámico campo eléctrico	0-2000 V/m Resolución 0,1 V/m
Máximo campo eléctrico medible	
Rango dinámico campo magnético	0-20000 nT Resolución 1 nT
Máximo campo magnético medible	
Componentes que es capaz de medir-campo eléctrico	Campo eléctrico total
Componentes que es capaz de medir-campo magnético	Bx, By, Bz
Modos de medida	Valor de pico, valor RMS
Velocidad de adquisición	Hasta 4 medidas por segundo
Almacenamiento de datos	Data logger (memoria interna 1 GB)
Interfaz de salida	
Alimentación	2 baterías AA
Peso	64g receptor 400g de sonda B 400g sonda E
Precio	845 sin IVA
Otras especificaciones	Rango Temperatura: 0-40 grados

Ilustración 321 Especificaciones técnicas FM10LS. Fuente: Uniovi.

La primera característica que se enuncia corresponde al software de post-procesado de la sensórica recibida, se aprecia una interfaz sencilla que funciona con el sistema operativo Windows.



Ilustración 322 Software FM-Data. Fuente: https://www.fauser.biz/se/reg_sw_e_antwort.htm
Acceso: 24/4/2018.

Este sensor además cuenta con la posibilidad del uso de filtros paso banda 16,6 Hz , paso alto 50 Hz y paso bajo 2 HZ.

En la medición de componentes de campo eléctrico no es capaz de distinguir entre Ex,Ey, Ez a menos que se acople la sonda de campo eléctrico EPL3.

La interfaz de salida estándar proporciona 4 niveles de tensión analógica correspondientes a E, Bx, By, Bz. Además existe la posibilidad de conexión USB para la descarga de datos al software asociado.

Por lo general es un equipo demasiado pesado para considerar acoplarlo a un dron, dado que en su máxima funcionalidad (es decir incorporando la sonda de campo eléctrico) puede llegar a un Kg de peso, perjudicando en gran medida la autonomía del vuelo.

Su exagerado volumen también es complicado de gestionar a la hora de acoplar en el vehículo aéreo.

Además el uso de baterías AA (pilas convencionales) no es lo más recomendable, tanto por razones económicas y ambientales, como también por la incomodidad de tener que cambiarlas cuando se agoten.

En cuanto al uso de la memoria, se prevé que 1 GB debería ser suficiente, aun así el resto de los modelos tienen un nivel de memoria muy superior.

14.1.1.2.- NFA1000



Ilustración 323 Sensor NFA1000. Fuente:

https://www.google.es/search?q=NFA1000&source=lnms&tbm=isch&sa=X&ved=0ahUKEwi_9bujt6DaAhULvBQKHd71CKkQ_AUICygC&biw=1440&bih=745#imgrc=bY3T5sv93V-pyM: Acceso: 24/4/2018.

Nombre	NFA1000
Software	NFAsoft
Rango frec	5Hz-1MHz
Precisión	5% Hz campo B
Rango dinámico campo eléctrico	0,1-1990 V/m

Máximo campo eléctrico medible	
Rango dinámico campo magnético	0,1-19990 nT
Máximo campo magnético medible	
Componentes que es capaz de medir-campo eléctrico	Ex, Ey, Ez
Componentes que es capaz de medir-campo magnético	Bx, By, Bz
Modos de medida	Valor de pico, valor RMS
Velocidad de adquisición	No especifica
Almacenamiento de datos	Almacenaje SD 4GB
Interfaz de salida	No especifica
Alimentación	2 baterías 3,7 V recargables
Peso	600g
Precio	1625 sin IVA
Otras especificaciones	No especifica

Ilustración 324 Especificaciones técnicas. Fuente: Uniovi.

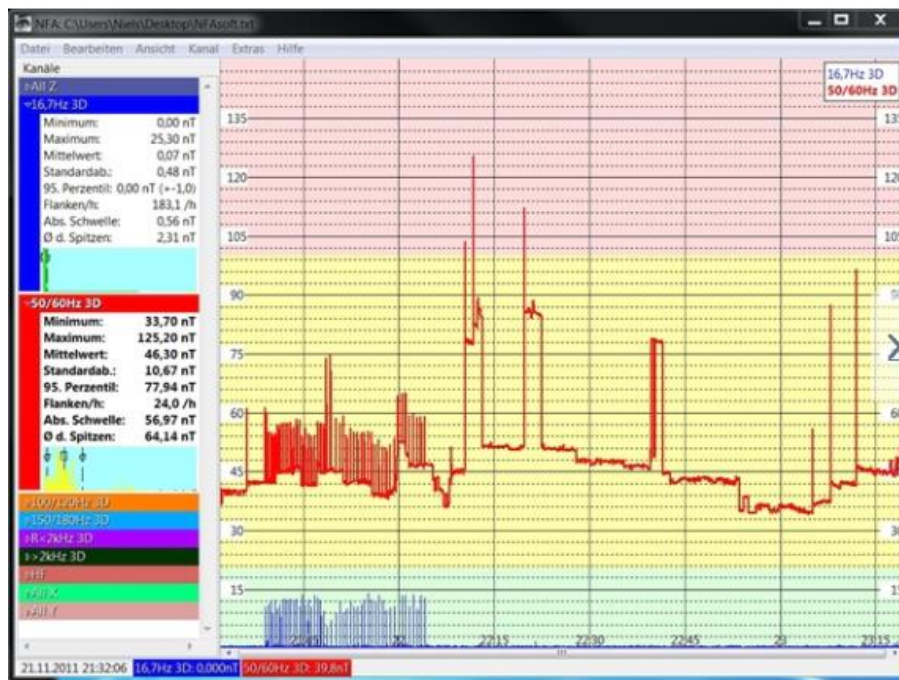


Ilustración 325 NFAsoft. Fuente: <https://www.gigahertz-solutions.de/en/rf-and-emf-meters/low-frequency-emf/nfa-series/334/nfasoft> Acceso: 24/4/2018.

Tanto para la medición de campo magnético como para la de campo eléctrico se permite alternar entre medida componente a componente o medida isotrópica.

Este equipo incorpora una tarjeta SD con mayor capacidad de almacenamiento que el dispositivo anterior.

De forma similar al sensor anterior permite la descarga de datos mediante conexión USB con el software NFAsoft.

El software es de uso sencillo e intuitivo y de uso en Windows y Linux, además es gratuito.

El principal problema de esta sonda radica en su elevado precio, que aparentemente no se justifica a juzgar por sus características.

14.1.1.3.- NFA 400



Ilustración 326 NFA 400. Fuente: <https://www.gigahertz-solutions.de/en/rf-and-emf-meters/low-frequency-emf/nfa-series/332/nfa400> Acceso:05/06/2018.

Nombre	NFA400
Software	NFAsoft
Rango frec	5Hz-400KHz
Precisión	5% Hz campo B
Rango dinámico campo eléctrico	0,1-1990 V/m
Máximo campo eléctrico medible	
Rango dinámico campo magnético	0,1-19990 nT
Máximo campo magnético medible	
Componentes que es capaz de medir-campo eléctrico	Ey
Componentes que es capaz de medir-campo magnético	Bx, By, Bz
Modos de medida	Valor de pico, valor RMS
Velocidad de adquisición	No especifica
Almacenamiento de datos	Almacenaje SD 4GB
Interfaz de salida	No especifica
Alimentación	2 baterías 3,7 V recargables
Peso	560g
Precio	1172 sin IVA
Otras especificaciones	No especifica

Ilustración 327 Especificaciones técnicas. Fuente: Uniovi.

Presenta características muy similares al modelo 1000 de la misma casa, aunque no presenta medida de las componentes x, z de campo eléctrico. Aun así se establece como suficiente para los requerimientos del proyecto.

El rango de media de la frecuencia de los armónicos de la onda fundamental es algo más reducido, pero también se plantea como suficiente para el proyecto.

Este sensor, de entre las cinco opciones, es uno de los más ligeros. Atributo especialmente importante en el vuelo con UAVs.

14.1.1.4.- NFA30M



Ilustración 328 NFA 30M. Fuente: <https://www.amazon.ca/Magnetic-Field-Analyzer-data-logger/dp/B003DKB040>

Nombre	NFA30M
Software	NFAsoft
Rango frec	16Hz-32Hz
Precisión	5% Hz campo B
Rango dinámico campo eléctrico	No especifica
Máximo campo eléctrico medible	
Rango dinámico campo magnético	0,1-19990 nT
Máximo campo magnético medible	
Componentes que es capaz de medir-campo eléctrico	No especifica
Componentes que es capaz de medir-campo magnético	Bx, By, Bz
Modos de medida	Valor de pico, valor RMS
Velocidad de adquisición	
Almacenamiento de datos	Almacenaje SD 4GB
Interfaz de salida	No especifica
Alimentación	2 baterías 3,7 V recargables
Peso	600g
Precio	550 sin IVA
Otras especificaciones	No especifica

Ilustración 329 Especificaciones técnicas. Fuente: Uniovi.

Como los demás modelos de esta familia usa NFAsoft, un software sencillo que permite la descarga y pos-procesado de los datos. Es capaz de operar en Linux y Windows.

Aunque con un precio significativamente inferior, este modelo no permite la medición de campo eléctrico, por lo que a priori no ofrecería las prestaciones necesarias.

Sin tener en cuenta este inconveniente presenta unas características muy similares a los otros dos modelos de la misma casa, teniendo además una batería con más autonomía.

14.1.1.5.- MC910



Ilustración 330 MC91. Fuente: https://www.thomann.de/gb/beyerdynamic_mc_910.htm

Nombre	MC910
Software	No especifica
Rango frec	60Hz
Precisión	No especifica
Rango dinámico campo eléctrico	No especifica
Máximo campo eléctrico medible	
Rango dinámico campo magnético	0,1mV por 100nT
Máximo campo magnético medible	
Componentes que es capaz de medir-campo eléctrico	No especifica
Componentes que es capaz de medir-campo magnético	Correspondiente a orientación
Modos de medida	Analógico
Velocidad de adquisición	
Almacenamiento de datos	No
Interfaz de salida	No especifica
Alimentación	No especifica
Peso	200
Precio	250 sin IVA
Otras especificaciones	No especifica

Ilustración 331 Especificaciones técnicas. Fuente: Uniovi.

Se trata de un modelo mucho menos sofisticado y barato, pero que presenta algunas limitaciones a la hora de aplicar al proyecto.

Aunque en principio se tomó en consideración por ser el más ligero con diferencia, no es capaz de medir campo eléctrico.

Además, mide campo magnético de forma sensible a la orientación espacial, es decir probablemente se hubiesen requerido tres vuelos (cambiando la orientación del sensor en cada uno de ellos) para poder conseguir las tres direcciones espaciales, o implementar la opción de instalar un servo que permitiera el cambio de orientación (con la correspondiente complicación técnica y de peso).

Otro problema de este sensor es la necesidad de incorporar un convertidor A/D a su salida, pues se trata de un dispositivo analógico. De forma que existiría una complicación en cuanto a peso o en cuanto a tratamiento posterior de los datos.

Al disponer de salida analógica por medio de un display en pantalla, no dispone de almacenamiento.

Por todos estos motivos este dispositivo queda descartado como candidato viable.

14.1.1.6.- Selección final:

De las opciones planteadas como posiblemente adecuadas, se destacan los sensores NFA, en concreto el 1000 ó el 400. De estos dos la única diferencia apreciable radica en que el 400 solo es capaz de dar el campo eléctrico en el eje y mientras que el nuevo modelo de la misma casa es capaz de discriminar en cada una de las tres direcciones.

Debido a que no se puede asegurar que tenga algún tipo de utilidad las mediciones de Y por eje y debido también al más que sobrado desempeño en la toma de datos de las torres de tensión, a su menor peso y menor coste se opta por la opción del san

15. Bibliografía:

[1]

«4. Basic Recipes — Picamera 1.10 documentation». [En línea]. Disponible en: <https://picamera.readthedocs.io/en/release-1.10/recipes1.html>. [Accedido: 22-may-2018].

[2]

«5. Advanced Recipes — Picamera 1.6 documentation». [En línea]. Disponible en: <http://picamera.readthedocs.io/en/release-1.6/recipes2.html#capturing-images-while-recording>. [Accedido: 24-may-2018].

[3]

«6.2. Métodos de Búsqueda (Python para principiantes)». [En línea]. Disponible en: http://librosweb.es/libro/python/capitulo_6/metodos_de_busqueda.html. [Accedido: 16-may-2018].

[4]

«10 Screen Command Examples to Manage Linux Terminals». .

[5]

«10.1. os.path — Common pathname manipulations — Python 2.7.15 documentation». [En línea]. Disponible en: <https://docs.python.org/2/library/os.path.html>. [Accedido: 11-may-2018].

[6]

«Bash», *Wikipedia, la enciclopedia libre*. 03-feb-2018.

[7]

«BOE-A-2017-15721.pdf». .

[8]

«BREVE HISTORIA DE LOS DRONES | Proyecto ALFA». [En línea]. Disponible en: <http://www.proyecto-alfa.net/forum/off-topic/breve-historia-de-los-drones>. [Accedido: 11-may-2018].

[9]

«Call exiftool from a python script?», *Stack Overflow*. [En línea]. Disponible en: <https://stackoverflow.com/questions/10075115/call-exiftool-from-a-python-script>. [Accedido: 16-may-2018].

[10]

«Communicating with Raspberry Pi via MAVLink — Dev documentation». [En línea]. Disponible en: <http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html>. [Accedido: 11-may-2018].

[11]

«datetime - Get current time in milliseconds in Python?», *Stack Overflow*. [En línea]. Disponible en: <https://stackoverflow.com/questions/5998245/get-current-time-in-milliseconds-in-python>. [Accedido: 11-may-2018].

[12]

«Decimal Degrees to Degrees, Minutes, Seconds, Python - rextester». [En línea]. Disponible en: <http://rextester.com/BRMA94677>. [Accedido: 15-may-2018].

[13]

«dronekit-python/__init__.py at master · dronekit/dronekit-python». [En línea]. Disponible en: https://github.com/dronekit/dronekit-python/blob/master/dronekit/__init__.py. [Accedido: 11-may-2018].

[14]

«DroneKit's goto() function should have a local/relative option vice only using absolute coordinates · Issue #151 · dronekit/dronekit-python», *GitHub*. [En línea]. Disponible en: <https://github.com/dronekit/dronekit-python/issues/151>. [Accedido: 11-may-2018].

[15]

«EL ORIGEN Y LA HISTORIA DE LOS DRONES», *Hemav*, 07-abr-2016. .

[16]

«EXIF Tags». [En línea]. Disponible en: <https://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/EXIF.html>. [Accedido: 16-may-2018].

[17]

«ExifTool by Phil Harvey». [En línea]. Disponible en: <https://www.sno.phy.queensu.ca/~phil/exiftool/>. [Accedido: 15-may-2018].

[18]

«Force Raspberry Pi output to composite video instead of HDMI | Bhavyanshu's Blog». [En línea]. Disponible en: <https://bhavyanshu.me/tutorials/force-raspberry-pi-output-to-composite-video-instead-of-hdmi/03/03/2014>. [Accedido: 21-may-2018].

[19]

«FTDI Cable 5V VCC-3.3V I/O - DEV-09717 - SparkFun Electronics». [En línea]. Disponible en: <https://www.sparkfun.com/products/9717>. [Accedido: 17-may-2018].

[20]

«GNU screen [quick_reference]». [En línea]. Disponible en: http://aperiodic.net/screen/quick_reference. [Accedido: 29-may-2018].

[21]

J. Schmidt, *gst-rpicamsrc: GStreamer element for the Raspberry Pi camera module*. 2018.

[22]

«How do I extract specific portions of a text file using python?» [En línea]. Disponible en: <https://www.computerhope.com/issues/ch001721.htm>. [Accedido: 16-may-2018].

[23]

«How to add GPS (geolocation) tags to photos», *Robert Zaremba blog*. [En línea]. Disponible en: http://rz.scale-it.pl/2012/01/02/how_to_add_gps_geolocation_tags_to_photos.html. [Accedido: 15-may-2018].

[24]

«How to Hack a Headphone Jack», *Circuit Basics*, 27-mar-2015. [En línea]. Disponible en: <http://www.circuitbasics.com/how-to-hack-a-headphone-jack/>. [Accedido: 21-may-2018].

[25]

«How to rename a file using Python», *Stack Overflow*. [En línea]. Disponible en: <https://stackoverflow.com/questions/2491222/how-to-rename-a-file-using-python>. [Accedido: 11-may-2018].

[26]

C. Lloyd, «How to See an Image's EXIF Data in Windows and macOS». [En línea]. Disponible en: <https://www.howtogeek.com/289712/how-to-see-an-images-exif-data-in-windows-and-macos/>. [Accedido: 14-may-2018].

[27]

I. Hatipoglu, «How to upgrade or downgrade raspberrypi's kernel? (Servoblaster problem Raspberry Pi2)», *Isa Hatipoglu*, 29-sep-2015. .

[28]

«i2c1-bcm2708.dtbo», *Google Docs*. [En línea]. Disponible en: https://drive.google.com/file/d/0B_P-i4u-SLBXb3VIN0N5amVBb1k/view?usp=embed_facebook. [Accedido: 11-may-2018].

[29]

«ImmersionRC 5.8GHz 600mW Tx.pdf». .

[30]

«Installing ExifTool». [En línea]. Disponible en: <https://www.sno.phy.queensu.ca/~phil/exiftool/install.html>. [Accedido: 15-may-2018].

[31]

«Is there a way to substring a string in Python?», *Stack Overflow*. [En línea]. Disponible en: <https://stackoverflow.com/questions/663171/is-there-a-way-to-substring-a-string-in-python>. [Accedido: 16-may-2018].

[32]

«Latest Reach / Reach RS topics - Community Forum». [En línea]. Disponible en: <https://community.emlid.com/c/reach>. [Accedido: 11-may-2018].

[33]

«Lidar Lite V3 - Raspberry Pi - Python - Page 2 - Raspberry Pi Forums». [En línea]. Disponible en: <https://www.raspberrypi.org/forums/viewtopic.php?t=179248&start=25>. [Accedido: 11-may-2018].

[34]

LIDARLite_v3_Arduino_Library: High-performance optical distance sensing. Garmin International, 2018.

[35]

«linux - What's a .sh file?», *Stack Overflow*. [En línea]. Disponible en: <https://stackoverflow.com/questions/13805295/whats-a-sh-file>. [Accedido: 11-may-2018].

[36]

J. Guerrero, «Lista con el nombre de los archivos de una determinada extensión en un directorio mediante Python», *El Blog de José Guerrero*, 20-oct-2013. .

[37]

«Manipulación de metadatos de imágenes con Python 3.x — 2015 — Blog — sirgazil». [En línea]. Disponible en: <https://sirgazil.bitbucket.io/es/blog/2015/manipular-metadatos-de-imagenes-con-python3/>. [Accedido: 14-may-2018].

[38]

«MAVLink», *Wikipedia*. 06-abr-2018.

[39]

«MAVLink Common Message set specifications». [En línea]. Disponible en: <http://mavlink.org/messages/common>. [Accedido: 11-may-2018].

[40]

«MAVLink Messages». [En línea]. Disponible en: http://python.dronekit.io/guide/mavlink_messages.html. [Accedido: 11-may-2018].

[41]

«MAVProxy — MAVProxy 1.6.2 documentation». [En línea]. Disponible en: <http://ardupilot.github.io/MAVProxy/html/index.html>. [Accedido: 11-may-2018].

[42]

«message_factory and send_mavlink(message) docs have incorrect advice · Issue #85 · dronekit/dronekit-python», *GitHub*. [En línea]. Disponible en: <https://github.com/dronekit/dronekit-python/issues/85>. [Accedido: 11-may-2018].

[43]

«Multiple mavlink devices connected to pixhawk on Serial 4/5? - Grupos de Google». [En línea]. Disponible en: <https://groups.google.com/forum/#!topic/drones-discuss/kvRpJCP1-Yw>. [Accedido: 11-may-2018].

[44]

2015 at 5:10am Posted by Stephen Dade on September 10 y S. M. V. Blog, «Pixhawk (and APM) Power Consumption». [En línea]. Disponible en: <https://diydrones.com/profiles/blogs/pixhawk-and-apm-power-consumption>. [Accedido: 25-may-2018].

[45]

«Pixhawk Telemetry Ports - Three Outputs?», *ArduPilot Discourse*. [En línea]. Disponible en: <https://discuss.ardupilot.org/t/pixhawk-telemetry-ports-three-outputs/6626>. [Accedido: 11-may-2018].

[46]

«PyExifTool – A Python wrapper for Phil Harvey’s ExifTool — PyExifTool 0.1 documentation». [En línea]. Disponible en: <https://smarnach.github.io/pyexiftool/>. [Accedido: 15-may-2018].

[47]

«python - OSError: [Errno 18] Invalid cross-device link», *Stack Overflow*. [En línea]. Disponible en: <https://stackoverflow.com/questions/42392600/oserror-errno-18-invalid-cross-device-link>. [Accedido: 24-may-2018].

[48]

«Python Arrays (With Examples)». [En línea]. Disponible en: <https://www.programiz.com/python-programming/array>. [Accedido: 11-may-2018].

[49]

«Python Dictionary». [En línea]. Disponible en: https://www.tutorialspoint.com/python/python_dictionary.htm. [Accedido: 11-may-2018].

[50]

ja.lopez, «Raspberry Pi - Captura con GStreamer», *Cenaptec*, 27-may-2017. [En línea]. Disponible en: <https://cenaptec.com/2017/05/27/raspberry-pi-captura-gstreamer/>. [Accedido: 17-may-2018].

[51]

«Raspberry Pi 3 GPIO Current - Raspberry Pi Forums». [En línea]. Disponible en: <https://www.raspberrypi.org/forums/viewtopic.php?t=151871>. [Accedido: 28-may-2018].

[52]

«Raspberry Pi Model B+ 3.5mm Audio/Video Jack», *Raspberry Pi Spy*, 22-jul-2014. [En línea]. Disponible en: <https://www.raspberrypi-spy.co.uk/2014/07/raspberry-pi-model-b-3-5mm-audiovideo-jack/>. [Accedido: 18-may-2018].

[53]

«Raspberry Pi Model B+ 3.5mm Audio/Video Jack», *Raspberry Pi Spy*, 22-jul-2014. [En línea]. Disponible en: <https://www.raspberrypi-spy.co.uk/2014/07/raspberry-pi-model-b-3-5mm-audiovideo-jack/>. [Accedido: 18-may-2018].

[54]

«Reach RTK docs». [En línea]. Disponible en: <https://docs.emlid.com/reach/>. [Accedido: 11-may-2018].

[55]

«Ryan Firebee», *Wikipedia*. 15-abr-2018.

[56]

Narbonne, *screenutils: Handle gnu-screen: creates/close/list sessions, injects commands..* 2018.

[57]

«serial - Difference between UART and RS-232?», *Electrical Engineering Stack Exchange*. [En línea]. Disponible en: <https://electronics.stackexchange.com/questions/110478/difference-between-uart-and-rs-232>. [Accedido: 11-may-2018].

[58]

«[Solucionado] Eliminar archivos de 0 bytes de tamaño a través | command-line». [En línea]. Disponible en: <https://www.enmimaquinafunciona.com/pregunta/32516/eliminar-archivos-de-0-bytes-de-tamano-a-traves-de-la-linea-de-comandos>. [Accedido: 16-may-2018].

[59]

«STICKY: 26 Common Pitfalls for Beginners - Raspberry Pi Forums». [En línea]. Disponible en: <https://www.raspberrypi.org/forums/viewtopic.php?p=653422#p653422>. [Accedido: 21-may-2018].

[60]

- «Telemetry / Serial Port Setup — Rover documentation». [En línea]. Disponible en: <http://ardupilot.org/rover/docs/common-telemetry-port-setup-for-apm-px4-and-pixhawk.html>. [Accedido: 11-may-2018]. [61]
- «Telemetry OSD for wifibroadcast», *befinitiv*, 06-jul-2015. . [62]
- «TT148 - 4053.pdf». . [63]
- J. Valdez y J. Becker, «Understanding the I2C Bus», p. 8, 2015. [64]
- «unix - How to control screen processes using python», *Stack Overflow*. [En línea]. Disponible en: <https://stackoverflow.com/questions/8315179/how-to-control-screen-processes-using-python>. [Accedido: 23-may-2018]. [65]
- «uPyLoader - simple file transfer and communication - MicroPython Forum». [En línea]. Disponible en: <https://forum.micropython.org/viewtopic.php?t=2245>. [Accedido: 11-may-2018]. [66]
- «Vehículo aéreo no tripulado», *Wikipedia, la enciclopedia libre*. 08-may-2018. [67]
- «VOLUMEN+1.+MEMORIA%2F10.+Comunicaciones.pdf». . [68]
- «Waypoint Protocol - QGroundControl GCS». [En línea]. Disponible en: http://qgroundcontrol.org/mavlink/waypoint_protocol. [Accedido: 11-may-2018]. [69]
- «What is packet? - Definition from WhatIs.com». [En línea]. Disponible en: <https://searchnetworking.techtarget.com/definition/packet>. [Accedido: 11-may-2018].