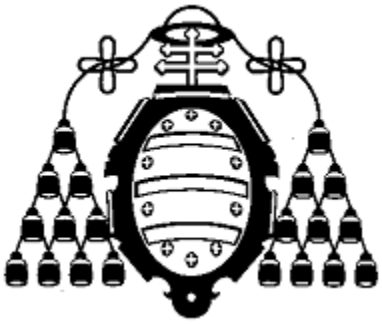


UNIVERSIDAD DE OVIEDO

CENTRO INTERNACIONAL DE POSTGRADO

MASTER EN INGENIERÍA MECATRÓNICA

TRABAJO FIN DE MÁSTER

Control de motores BLDC de un multicóptero a través de una interfaz móvil

09/2016

Christian HOHMANN **Prof. Juergen WALTER**

[Firma] **[Firma]**

INDICE GENERAL

1 RESUMEN

En este trabajo se plantea el desarrollo de un banco de pruebas para medir el empuje de motores BLDC con hélices.

Para esto, se ha realizado primero una revisión de la capacidad de medida de la célula de carga utilizada. Las optimizaciones incluyen la adición de un otro motor con una hélice y la reconstrucción del banco de pruebas, en base a la propuesta de construcción del multicóptero tripulado. Una revisión del software de la placa del microcontrolador ha conducido a una mejora, en términos de aumento de la eficiencia. La placa de desarrollo de *Silicon Laboratories* utilizado anteriormente, también encuentra un uso para esta versión del banco de pruebas. Con el uso de este banco de pruebas, se desea medir la fuerza de empuje y las revoluciones de los Motores BLDC. El conocimiento de la respuesta al escalón resultante al cambio repentino de la velocidad del motor, es un componente importante para el reglaje de vuelo de un multicóptero. La comparación de los resultados de las medidasde fuerza de empuje con los resultados de una simulación de un otro trabajo de fin de master se usa para comprobar la confiabilidad de este banco de pruebas.

Como en un proyecto anterior sobre este banco de pruebas, el banco se va a dividir en dos subsistemas. Por esto una aplicación para Smartphone se ha desarrollada con la plataforma de desarrollo de Android Studio. Con esto son posible un arranque y la parada de los motores y también la medida de las revoluciones actual y la fuerza de empuje.

PALABRAS CLAVE

Banco de pruebas – Fuerza de empuje – Aplicación Android – Multicóptero

2 INTRODUCCION

En los últimos años una nueva tendencia ha surgido en el sector del ocio. Mediante el uso de un Multicóptero, es posible disparar fotos desde el aire para todo el mundo. Profesionalmente estos dispositivos se utilizan también para descargar paquetes o para las emisiones de televisión de eventos deportivos. El vuelo de un Multicóptero tripulado es interesante porque la experiencia de vista de pájaro por el usuario puede ser experimentado. Hay, por ejemplo, no rotor de cola o timón requerido ya que el cambio de la elevación o la propulsión es tomado por el cambio en la velocidad de los motores. Dado que la potencia de accionamiento es exclusivamente eléctricamente, el camino de entrada es muy tranquilo y amigable con el medio ambiente.

En cuanto al comportamiento y la seguridad de vuelo, este diseño podría ser interesante en el campo del tráfico de cercanías. El fallo de un rotor, en un Multikopter con seis hélices, son compensados por la operación de los otros cinco rotores y el objeto volador todavía puede ser guiado a un aterrizaje seguro. Ellos ofrecen varias ventajas sobre otras aeronaves, como por ejemplo no se requiere una pista de aterrizaje.

En el desarrollo de un multicóptero, varios factores deben ser considerados. Importantes son el posicionamiento, la selección de los accionamientos eléctricos, la regulación del sistema y el conocimiento del comportamiento del salto dela fuerza de empuje de las hélices con los motores. La regulación está diseñada por medio de una unidad de control, que se fija de forma permanente en el multicóptero. Para que este sistema funcione en términos de un funcionamiento estable de vuelo, también interesa el tiempo de subida de la hélice, en caso de cambio repentino en la velocidad del motor, y el empuje actual al llegar a una velocidad del motor.

Sobre la base de un trabajo de estudio para el desarrollo de un Hexacopter en formato modelo, un Multicóptero con seis botalones para el vuelo tripulado ha sido desarrollado en hardware y software. Para un brazo de este multicóptero se avanza en la presente tesis un banco de pruebas ya disponible. Con este fin, en un primer parte se describe el planteamiento del problema y las tareas que se deben hacer. Después es descrito el estado actual de la técnica del banco de pruebas. En esta sección debe estar claro cómo funciona el banco actual y los puntos en lo que se necesitan mejoras. En la tarea posterior se discute más que cambios se introducen. Después en un análisis estático del sistema de medida sin hélice se debe demostrar de cómo el sensor utilizado proporciona valores fiables. A continuación, se explica cómo el banco se cambia en términos de diseño y de software. Una parte importante de este trabajo, es el desarrollo de una aplicación de teléfono con la cual se controla y evalúa el banco. Por último, se comprueba con el banco desarrollado, la medida de empuje cuando se cambia rápidamente la velocidad de rotación. Los resultados se compararon con los de una simulación de una tesis anterior.

3 PLANTEAMIENTO DEL PROBLEMA Y DE LAS TAREAS

3.1 Problemas

El objetivo de esta tesis es la extensión del banco de pruebas para medir el empuje de una unidad de multicóptero, en el contexto del desarrollo de un multicóptero tripulado. Para que estas medidas puedan realizar de forma realista, el banco de pruebas necesita ser reconstruido y cambiado en hardware y software.

Los problemas actuales son:

- No es posible medir el empuje real. La célula de carga se coloca entre el brazo de soporte y el motor. Entonces la fuerza de empuje que los motores ejercen sobre el brazo de soporte no se puede medir. La construcción del banco de pruebas no se inspira en la del multicóptero tripulado, como descrito en la introducción.
- El software no es adecuado para el ensayo de diferentes escenarios de prueba. Además, la eficiencia del programa es pobre.
- No hay botón de parada de emergencia no se encuentra
- La evaluación con *Simulink* es engorrosa porque hay que llevar un ordenador para evaluar los datos.

3.2 Tareas

Cambios en el banco de pruebas:

- Montar el segundo motor e implementar el control
- Ajustar a la estructura del multicóptero tripulado.
- Cambiar la posición de la celda de carga -
- Poner en práctica la medida de velocidad
- Aplicación de Smartphone para el control y la evaluación del banco de pruebas

Las medidas con el banco de pruebas son:

- Extender las medidas de 5 a 10 segundos, para investigar la fuerza de empuje después de apagar el motor.
- Medidas de empuje a diferentes velocidades.
- Medida de empuje al cambiar la velocidad durante una prueba
- Determinar el tiempo entre el momento de inicio, en el que no se mide empuje y el momento cuando se mide uno.
- Determinar el tiempo requerido para cambiar entre dos velocidades.

Estas medidas son importantes para regular el comportamiento de vuelo de multicóptero.

El diagrama de bloques en la Figura 3.1 describe el nuevo sistema.

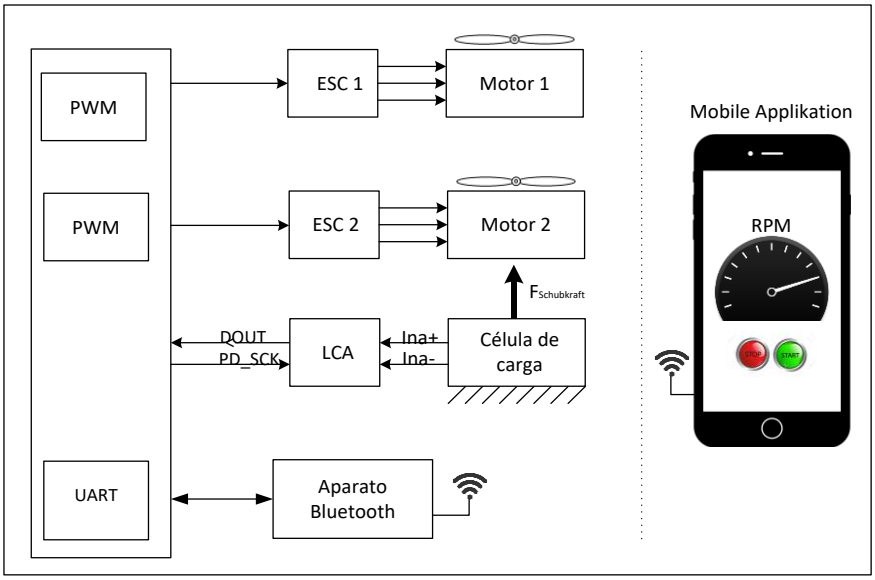


Figura 3.1 Diagrama de bloques del sistema

4 BANCO DE PRUEBAS ACTUAL

A continuación, vamos a explicar cómo funciona el banco de pruebas actual y cómo esto debe desarrollarse más.

En las siguientes secciones, los componentes y el funcionamiento de la cadena de medida se consideran con más detalle. Empezaremos con el sistema de los accionadores, y luego discutir los sensores y la transmisión de datos de medida.

4.1 Hardware

Placa Microcontrolador

El microcontrolador de 8 bits C8051F580-TB (Figura 4.1) del fabricante *Silicón Laboratories*, representa un componente central para un multicóptero y este banco de pruebas. Debe tener periféricos adecuados para el procesamiento de las señales de entrada y de salida. Por otra parte, una cierta potencia de cálculo a condición de que todos los comandos y, tareas se pueden procesar en los ciclos de tiempo requerido. Una parte de la periferia del dispositivo, que se requieren para el desarrollo de la plataforma de pruebas, incluyen la interfaz UART, matrices contador programable (PCA), y varios temporizadores.

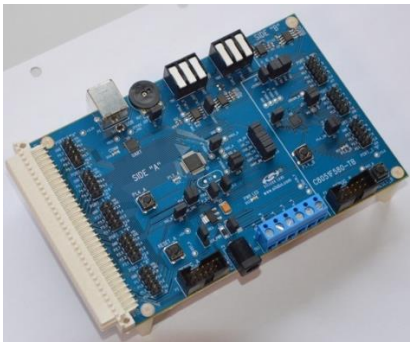


Figura 4.1 Placa microcontrolador

Motores BLDC y Modulación de ancho de pulso (PWM)

El motor CC sin escobillas Q80 (BLDC) de la compañía *Hacker Motor GmbH* se utiliza para accionar la hélice. La fuente de alimentación está asegurada por dos baterías de polímero de litio (Li-Po) y el motor sólo puede ser operado a través de un controlador electrónico de velocidad (ESC) adecuada para el motor.

Figura 4.1 Q80 BLDC Motor [1]

Con un ESC, se ajusta la señal PWM 50 Hz al motor. La periferia del microcontrolador libra dos matrices contadoras programables (PCA0 y PCA1). Con éstos, una señal PWM se puede generar.

4.2 Sensor

La célula de carga es el componente más importante de los instrumentos electromecánicos de la cadena de medida. Forma la interfaz entre la entrada mecánica (potencia) y la salida eléctrica (voltaje). Las deformaciones pueden ser detectadas por medio de medidores de deformación, y se convierten en una señal eléctrica.

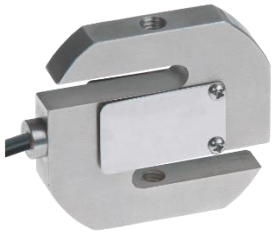


Figura 4.2 Célula de carga S40S-G3-0020 de *Bosche GmbH*

Amplificador de célula de carga (LCA)

El LCA convierte la señal analógica de la célula de carga en un valor de 24 bits. Con este, los valores de peso se pueden leen después de la calibración. El amplificador de medida HX711 puede digitalizar valores analógicos 90 Hz (es decir, 90 valores por segundo).

Figura 4.3 Load Cell Amplifier HX711, [2]

4.3 Transmisión de datos con UART y Bluetooth

Con el fin de mostrar los valores medidos de la célula de carga gráficamente utilizando *Simulink*, se utiliza una interfaz universal de transmisor receptor asíncrono. Con el UART, se pueden transmitir 8 bits a la vez.

Con dos módulos Bluetooth, es posible la comunicación inalámbrica en serie entre el ordenador y el banco de pruebas (ver Figura 5.6). La transferencia de datos puede llevarse a cabo con el protocolo de puerto serie Bluetooth (SPP).

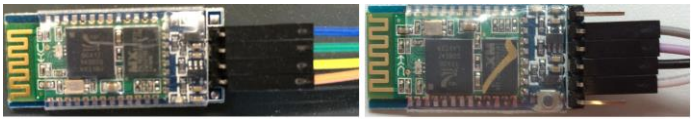


Figura 4.4 EGBT-046S (Slave) y EGBT-045MS (Master)

4.4 Matlab Simulink

Con Simulink y sus herramientas, tales como la configuración de serie, se puede recibir y leer datos a través de UART y el puerto COM del ordenador. Esto no requiere un código de programa.

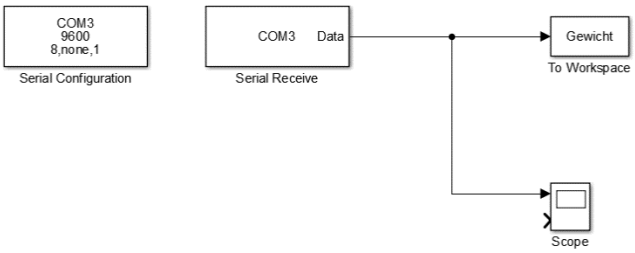


Figura 4.5 Modelo Simulink para recibir y mostrar las medidas del empuje

La medida de empuje del sistema se inicia después de lanzar una simulación en Simulink. Por último, los datos se muestran como un gráfico, visto en la Figura 4.2. En X tenemos el tiempo y en Y la fuerza de empuje en gramos.

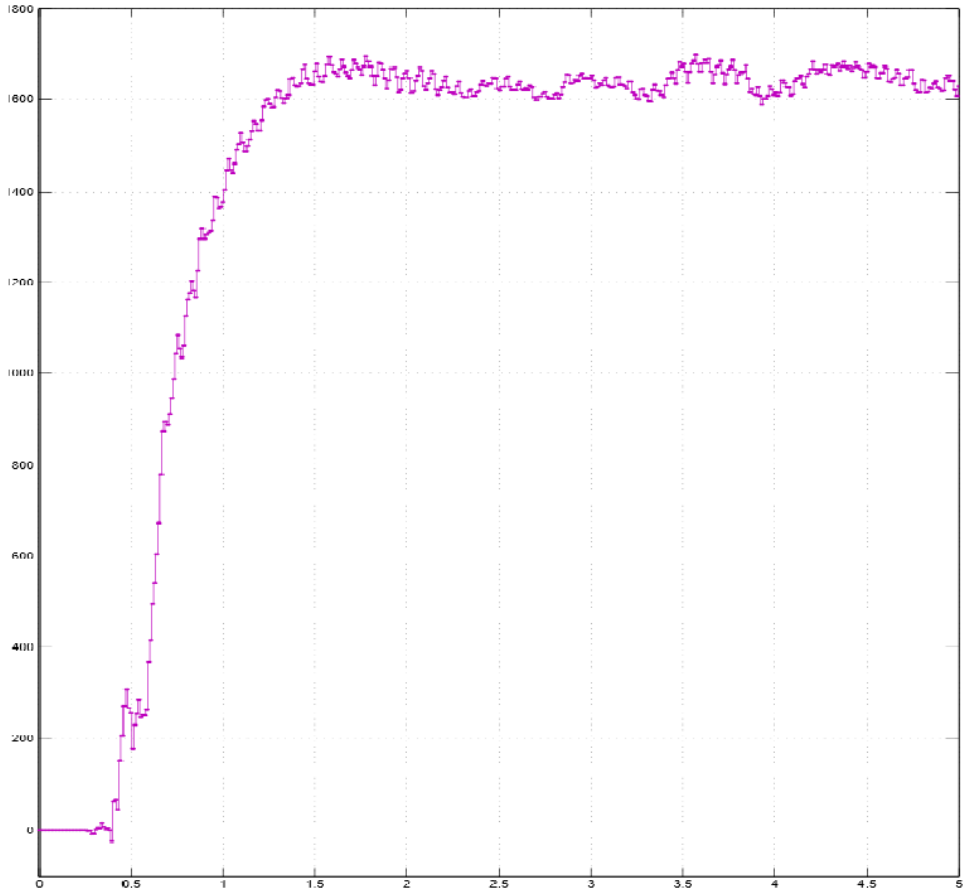


Figura 4.2 Fuerza de empuje en Simulink

4.5 Software

La cadena de medida y por lo tanto las 3 partes principales del software son:

- Accionar el motor BLDC a través del ESC

- La detección de los datos medidos, que son proporcionados por la célula de carga, y se transmiten a la placa de microcontrolador a través de la conexión en serie del LCA
- La transmisión de los datos de medida de Simulink usando UART y Bluetooth

La placa de microcontrolador comienza con la inicialización del periférico, cuando está conectado a la fuente de alimentación. El controlador de velocidad del motor también requiere una inicialización, que se lleva a cabo inmediatamente después de arrancar el MC por software. La medida de empuje de 5 segundos se lanza después de tarar la célula de carga, pulsando el interruptor en el puerto 1.4. Al pulsar el botón el PWM en el puerto 0.0 de la placa MC cambia y el ESC establece la velocidad adecuada para el motor. La medida se consigue simultáneamente con el arranque del motor. Después de 5 segundos de tiempo de medida y restablecimiento del PWM a cero, los motores se detienen. Posteriormente, los datos de medida son transmitidos por el UART. En la Figura 4.3 se puede ver el flujo del programa.

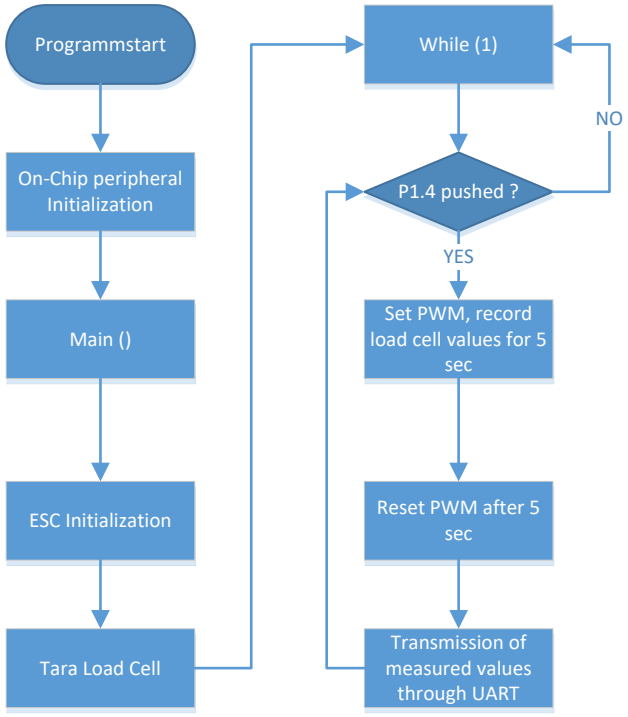


Figura 4.3 Flujo del programa C del banco actual

4.6 Fuerza de empuje y medida de la velocidad

En la Figura 4.4, la relación parabólica entre la velocidad y el empuje puede ser vista. El tiempo transcurrido entre el momento de inicio y el máximo empuje a una velocidad dada, no se determinó con esta versión del banco de pruebas. Como se explicará más adelante, un empuje por motor de 98 N (es decir, alrededor de 10 kg) a una velocidad de 4700 rev/min es un requisito para el vuelo estacionario. De acuerdo con la Figura 4.4, se puede suponer que estos valores se pueden lograr.

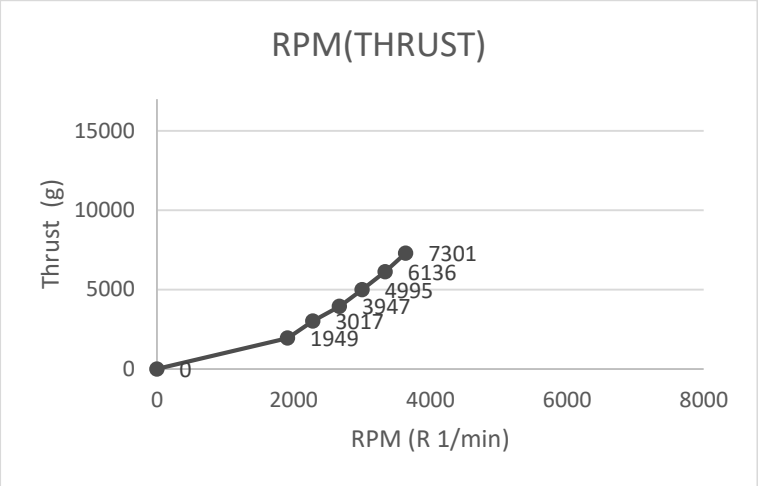


Figura 4.4 Revoluciones en función de la fuerza de empuje del motor BLDC y hélice

5 PRUEBAS ESTATICAS CON LA CÉLULA DE CARGA

Con el fin de verificar la exactitud y fiabilidad de las lecturas de la célula de carga, se realiza un análisis de la capacidad de medida con varios pesos certificados. Para ello, varias medidas se llevan a cabo con 25 valores medidos cada vez. Por esto el sistema se libera del motor y la hélice, y se hacen pruebas estáticas.

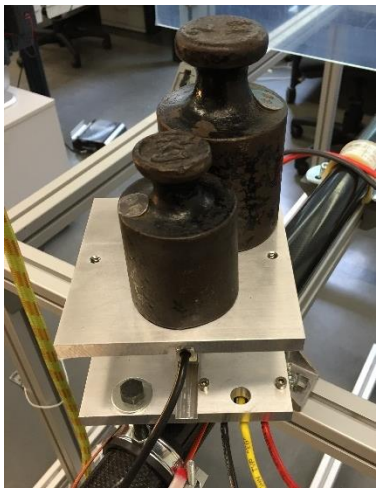


Figura 5.1 Los pesos se colocan sobre la célula de carga

Al medir, los siguientes aspectos son importantes:

- Precisión
- Repetitividad

- Reproducibilidad
- Estabilidad
- Linealidad

Método:

Para el análisis, se consideran la ubicación y la dispersión del valor medido en una zona de tolerancia. Los índices de capacidad C_g y C_{gk} especifican si un dispositivo de medida es adecuado para la zona. Estos valores se calculan para una serie de medidas que incluye 25 valores.

Los valores de análisis son los siguientes (todos los pesos están en gramos):

Tabla 5.1 Resultados del análisis de capacidad de medida

Peso de prueba	Tolerancia	Resolución	Promedio	Desviación estándar	C_g	C_{gk}
1000	200	10	999,32	0,6354	2,96	2,86
2000	200	10	1995,96	0,6758	9,87	7,87
3000	200	10	3002,6	0,6455	10,33	8,99

En el diagrama de la Figura 5.2 se puede ver la linealidad. El peso medido de acuerdo con el peso calibrado en todo el rango de los pesos sin el daño de la tolerancia.

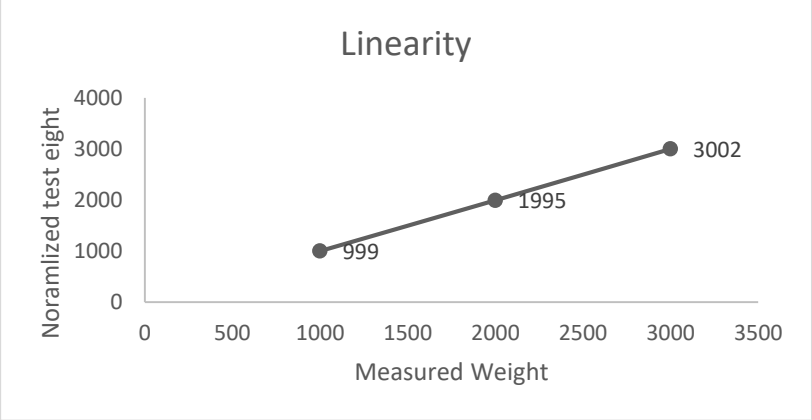


Figura 5.2Linealidad del método de medida

Los valores de referencia C_g y C_{gk} de 1,33 no se descarga inferior. Por consiguiente, el análisis del sistema de medida ha demostrado que el sensor es capaz de realizar medidas de forma fiable y dentro de las tolerancias. Linealidad también se da como la desviación estándar de todos los pesos medidos para girar alrededor del mismo valor de aproximadamente 0,6.

6 CONVERSION DEL BANCO DE PRUEBAS

6.1 Diseño

Figura 6.1 Nuevo diseño del banco de pruebas

En la Figura 6.1, se muestra el nuevo banco de pruebas con los dos motores y hélices.

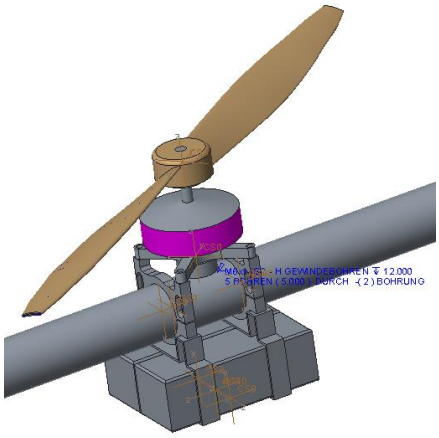


Figura 6.2 Montaje del motor usando abrazaderas de tubo al brazo de soporte.

La construcción del montaje de los motores, se basa en el diseño de la del multicóptero tripulado.

6.2 Hardware

Puesta en marcha del segundo motor

Otra ESC genera una señal PWM para el segundo motor. Esta señal también puede ser generada con la PCA0 de periferia de microcontrolador. El PCA0 tiene 6 módulos diferentes que pueden generar una señal PWM. Con los dos PCA de la

placa, se pueden en teoría generar todas las señales PWM para los 12 motores del multicóptero tripulado.

Célula de carga

Ambos motores producen un empuje máximo de 17 kg cada uno. En un empuje resultante de 34 kg, la célula de carga utilizada anteriormente ya no es adecuado con una capacidad nominal de 20 kg. Se sustituye por un sensor idéntico con una carga nominal de 50 kg.

Con el fin de medir la fuerza de empuje, que ejercen ambos motores en el brazo, la posición de la célula de carga se modifica y ajusta por debajo del brazo de soporte en el bastidor (Figura 6.3).

Figura 6.3 Nueva Fijación de la célula de carga

Medida de la velocidad

Para la medida de la velocidad, dos barreras de luz réflex (uno por motor) del tipo MRL601 son utilizados.

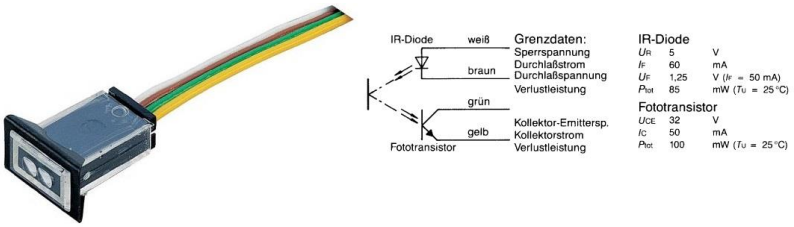


Figura 6.4 Barrera de luz réflex MRL 601 y tabla de datos

La barrera de luz consta de un diodo transmisor de infrarrojos y un fototransistor, que actúa como un receptor. En un reflejo, la corriente se pasa a través del

fototransistor. Si el sensor no refleja, no pasa ninguna corriente, y no hay ninguna señal de salida. Con el fin de probar el sensor, antes se genera un pequeño circuito con dos resistencias. Con un osciloscopio, se puede observar el cambio de señal.

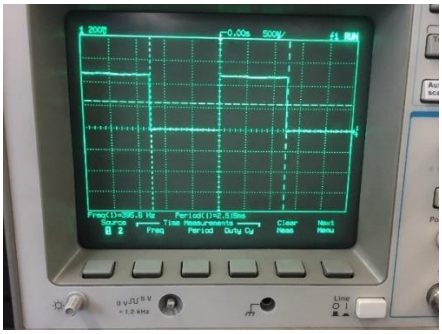


Figura 6.5 Señal de la medida de las revoluciones en el osciloscopio

La medida de la velocidad con una barrera de luz que se refleja, funciona mediante la reflexión y no la reflexión sobre una superficie seleccionada. El motor está marcado por un blanco y negro cinta rayada (4 blanco y 4 rayas negras). En cada reflexión, un borde es visible en el osciloscopio. Este borde se detecta con la placa de evaluación, y en el código fuente, el número de revoluciones por minuto se puede calcular. La barrera de luz se fija como en Figura 6.6.

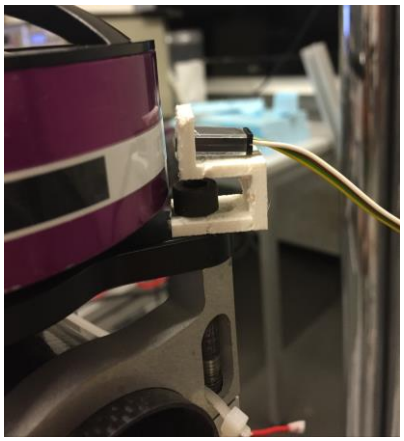


Figura 6.6 Soporte para la barrera y patrón de bits fijado contra el motor

Cuatro rayas blancas representan cuatro bordes. Esto corresponde a una revolución. Para calcular la velocidad de rotación, también se requiere un componente de tiempo. Es decir, si el tiempo entre dos flancos, y el número de éstos es conocido por revolución, se puede calcular la velocidad.

La velocidad de rotación es el inverso del tiempo de circulación. ($n = \frac{1}{T}$) entonces:

$$RPM = \frac{60\ s}{4 * T}$$

Formula 1 Cálculo de las revoluciones por minuto

con:

- T: Tiempo entre dos flancos

El oscilograma de la Figura 6.5 muestra una frecuencia de 395,6 Hz. Con la fórmula 1 se calcula una velocidad de rotación de 5934 rpm. La velocidad se transmite también a través del UART.

Parada de emergencia

Por razones de seguridad, se implementa una parada emergencia.

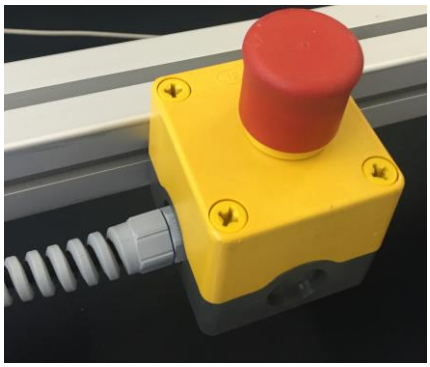


Figura 6.7 Botón de parada de emergencia

6.3 Software

Nuevo programa C

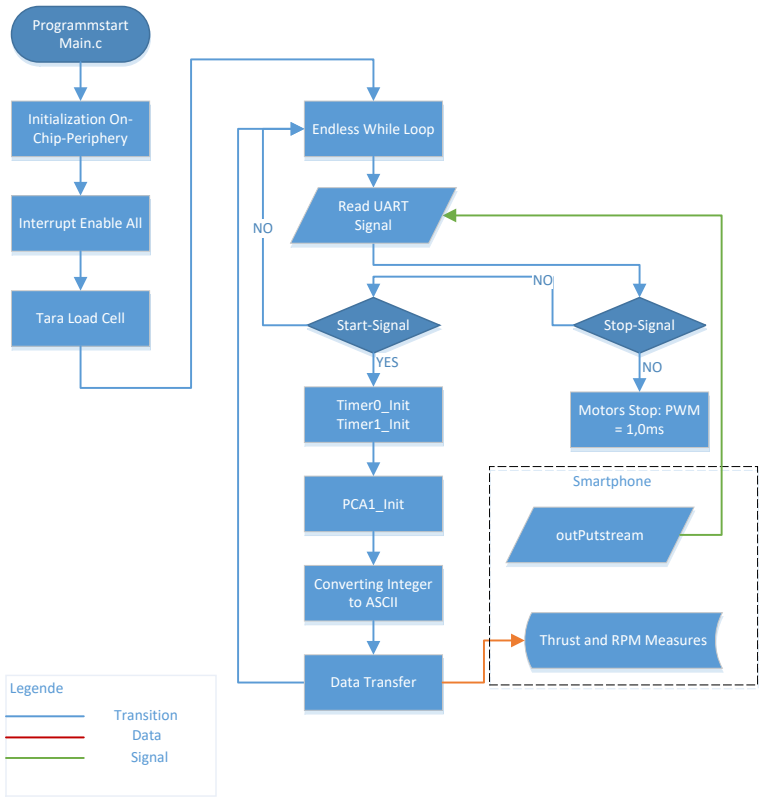


Figura 6.8 Flujograma nuevo del programa C

El comienzo del programa no difiere del de la versión anterior. Cuando se inicia el programa, después de la inicialización de periféricos, se tara la célula de carga.

Los cambios en el programa se llevan a cabo en el control de motores y en las medidas de empuje. Dado que la operación se pondrá en marcha mediante la aplicación móvil en el futuro, el programa debe ajustarse en consecuencia. La forma más sencilla es la transferencia de un carácter ASCII en la interfaz UART, que el programa C compara con la condición de una sentencia if. Una vez que se produce una coincidencia, la función asignada se puede iniciar.

Para arrancar los motores y las medidas de la fuerza de empuje, se requiere una "a". Posteriormente, la inicialización del temporizador 0 y el temporizador 1. Ambos lanzamientos se configuran de manera que cada 11,1 milisegundos se activan en función de interrupciones. Este tiempo corresponde a la velocidad de datos del LCA. En la rutina de servicio de interrupción ISR_Timer0 la función para registrar los valores de medida de células de carga se lleva a cabo. En el ISR_Timer1 la PCA0 genera una señal PWM para el accionamiento de los motores. Casi al mismo tiempo la PCA1 se inicializa y comienza el ISR para la detección de bordes, que se utiliza para la medida de la velocidad.

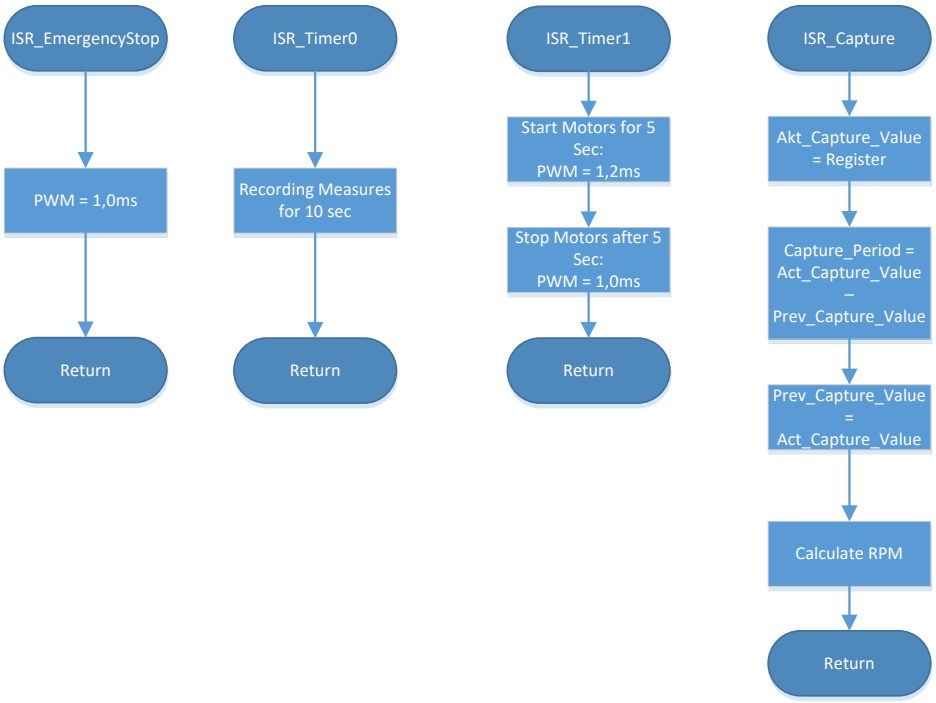


Figura 6.9 Flujograma de las interrupciones

Transferencia de datos a la aplicación de Smartphone.

Al final de cada pasada del bucle, los datos de la fuerza de empuje y la velocidad de rotación se envían a la aplicación. Así, los datos pueden ser procesados por el Smartphone, los valores medidos obtenidos del formato de *integer* se convierten a caracteres ASCII y se envían como una cadena de datos.

Para que más adelante la aplicación reconozca que lugares en la cadena de signos corresponden al empuje, y cuál son de la velocidad, se utilizan delimitadores. Además, se utilizan otros signos para marcar el comienzo y el final de cada cadena.

Parar los motores

La parada de los motores, acepta una función separada que es independiente de las interrupciones. Aquí también una sentencia if se lleva a cabo en el bucle sin fin del programa principal. Una vez otro símbolo se recibe que "a" en la placa del microcontrolador, el ESC envía una señal PWM de 1,0 ms y los motores se detienen. En cada parada, todos los temporizadores e interrupciones se inicializan. Por lo tanto, un reinicio de las medidas es posible sin deber reinicializar el microcontrolador.

7 APLICACIÓN SMARTPHONE

La aplicación se llevó a cabo con Android Studio. Con esto, el banco de pruebas puede ser controlado y evaluado. Una de las características de la aplicación es el arranque y la parada de los motores con la medida de empuje que se ejecutan simultáneamente. Eso funciona con el UART. Una vez a través de UART, el símbolo deseado se obtiene por la aplicación, una marche de prueba puede llevarse a cabo con su flujo de costumbre.

7.1 Comunicación entre Placa microcontrolador y aplicación

Los datos se envían al Smartphone con Bluetooth. El adaptador de Bluetooth del dispositivo móvil puede conectarse al módulo Bluetooth EGBT con el protocolo serie bluetooth descrito antes. Con este fin, en el dispositivo receptor se debe establecer un bluetooth socket, que inicia la conexión, y puede ser leído y escrito en los datos.

7.2 Funciones de la aplicación – Las Activities

Device List Activity

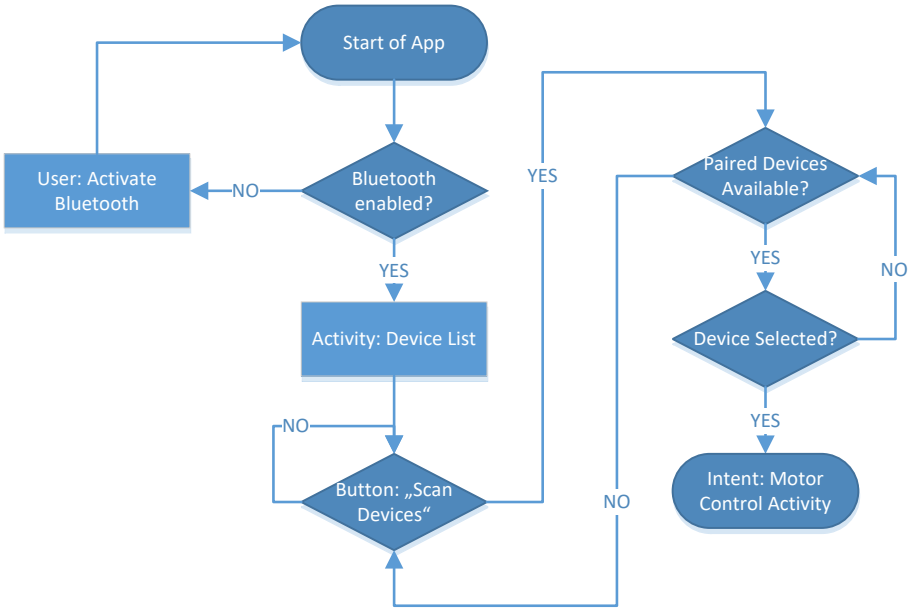


Figura 7.1 Flujo de programa de la device list Activity

Pulsando el botón "Scan Devices " (véase la Figura 7.1 y la Figura 7.2), se presenta al usuario en la pantalla “Device List Activity”, que dispositivos están disponibles actualmente. En esta activity también se comprueba si el adaptador Bluetooth está activo, si no, al usuario se le pedirá activarlo. Con inicio de la conexión, se pasa a la pantalla de “Motor Control”.

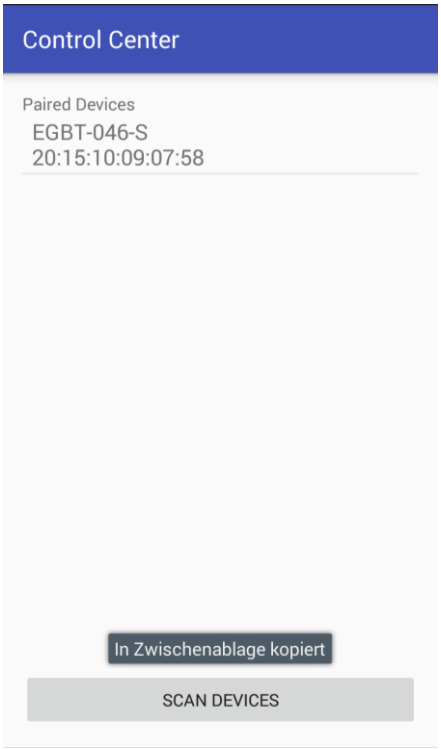


Figura 7.2 Device List Activity

Motor Control Activity

Aquí el bluetooth socket está creado y puede leer los datos recibido por UART. Al escribir sobre el socket, los caracteres individuales pueden ser enviados a la interfaz UART de la placa de microcontrolador.

Las funcionas para el uso y la evaluación de la activity son los siguientes:

- Arrancar y bloquear el arranque de los motores.

  El botón de inicio no está activo en principio (véase la Figura 7.3), para evitar que los motores se arranquen inadvertidamente. El botón se activa con la cerradura que está representada por un interruptor deslizante (véase la Figura 7.4).

- Parar los motores

  Parar los motores es posible en cualquier momento. Mediante un mensaje de advertencia se muestra al usuario el efecto de esa tecla.

- Recibir los datos como una cadena, y la subdivisión en subcadenas, para mostrar el empuje y las revoluciones actual en la interfaz usuaria en el Smartphone.
- Exportación de datos a un archivo CSV
- Desconectar la conexión Bluetooth

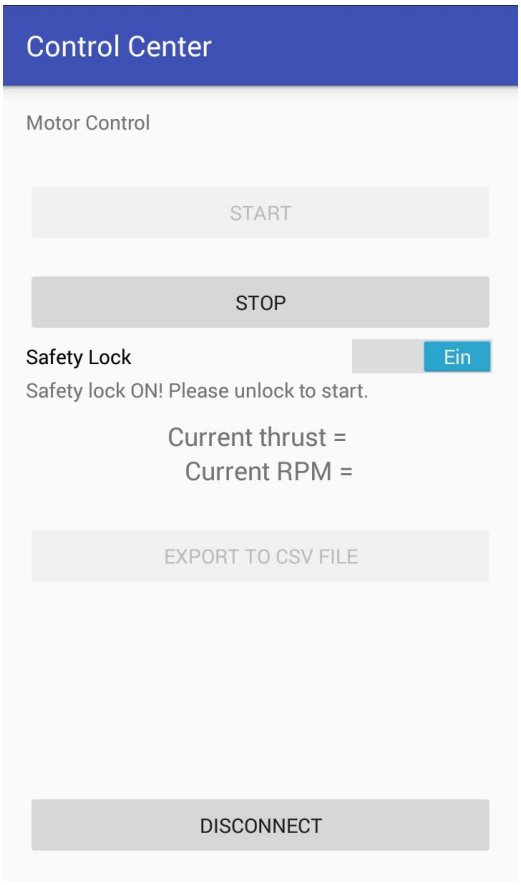


Figura 7.3 Bloqueador de arranque activo

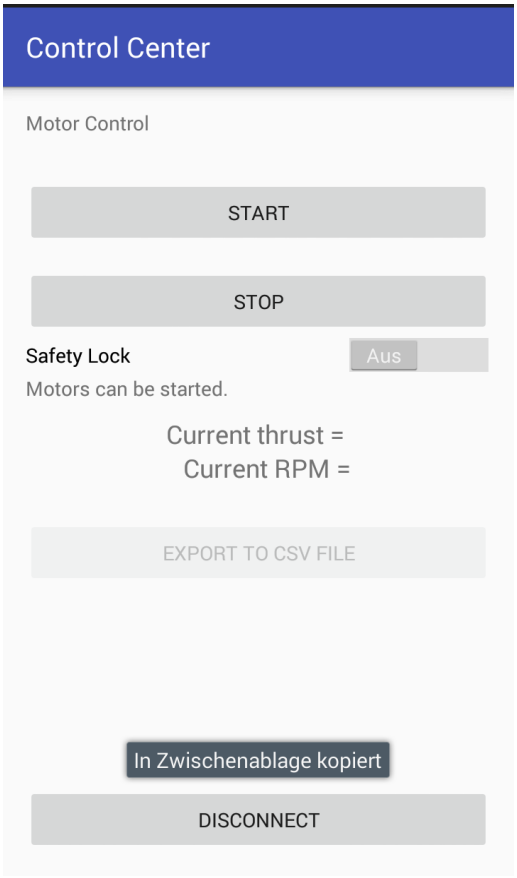


Figura 7.4 Bloqueador de arranque no activo

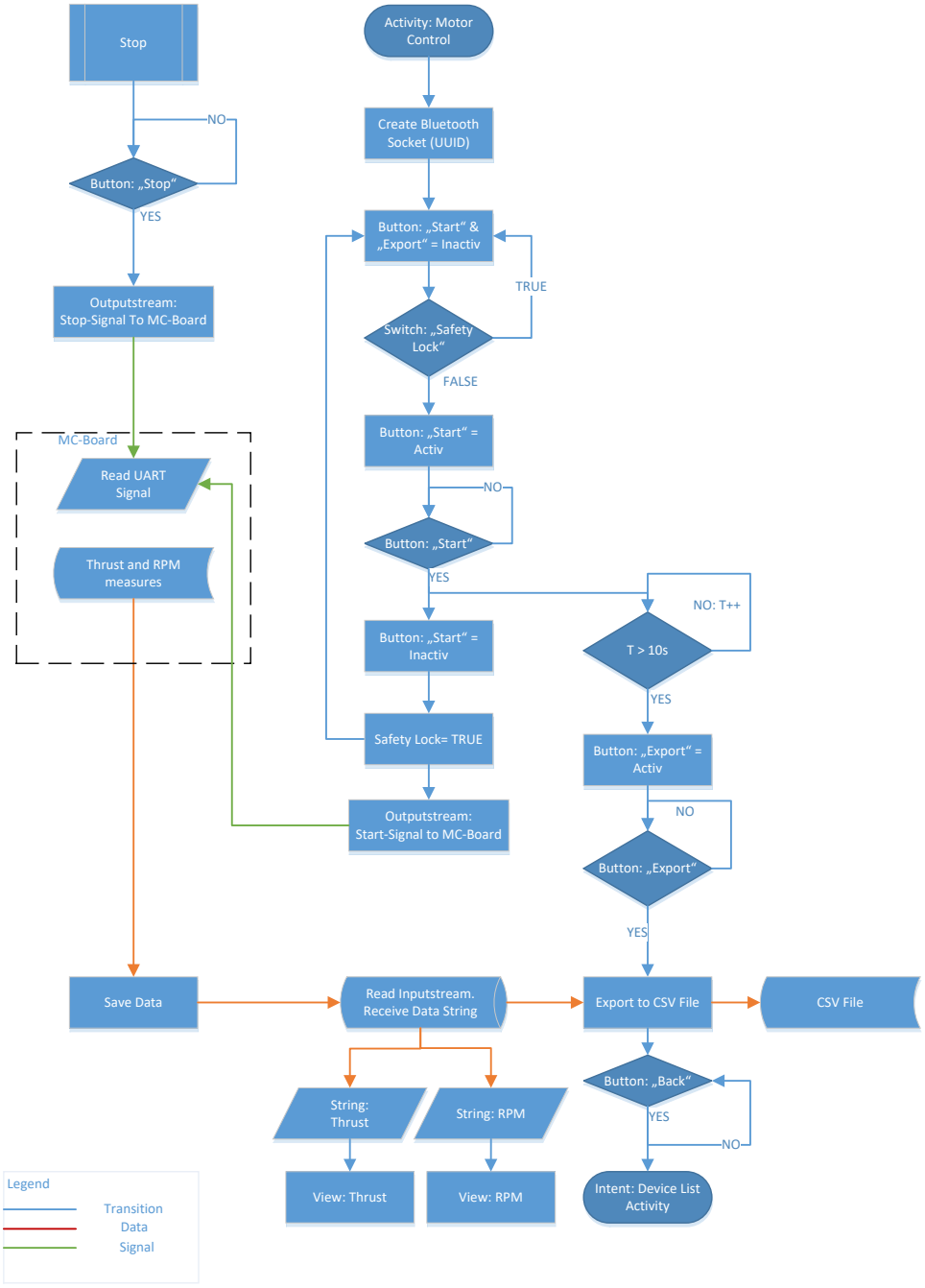


Figura 7.5 Flujograma de la activity motor control de la aplicación

Registro, visualización y exportación de los datos de medida

Al pulsar el botón de arranque, los motores se ponen en marcha y al mismo tiempo se lleva a cabo la grabación de los datos medidos.

Los datos (de empuje y velocidad) se envían por la placa de microcontrolador por UART, después de una conversión del formato integer, como una cadena ASCII con 17 posiciones. Por lo tanto, la aplicación detecta cuando una cadena comienza y termina, incluyendo un # y un carácter ~, que se envían al principio y al final de

cada cadena, respectivamente. Para separar los datos dentro la cadena, con el MC se envía un signo + entre los valores medidos.

Los datos para el empuje y la velocidad de los dos motores son exportados cada uno a un archivo CSV, en una carpeta de archivos de Dropbox en el Smartphone. Estos archivos pueden ser accedidos por Excel, y se pueden crear tablas de datos para analizar la respuesta al escalón.

El desarrollo del banco de pruebas de hardware y software, se ha completado con la aplicación funcionando. En las secciones siguientes se trata de la determinación de la respuesta al escalón y la comparación con una simulación creada en una tesis anterior.

8 PRUEBAS DINÁMICAS

Para la tesis "Desarrollo de un multicóptero tripulado en hardware y software", se han identificado los requisitos del rendimiento necesaria para poner en flotación un multicóptero y simular la velocidad a la que se alcanza la fuerza de empuje requerida.

Después de la ejecución de las medidas de empuje con el banco de prueba, los resultados se comparan con los de la simulación y se comprueba si el banco reproduce valores fiables. Con estas pruebas se debe también investigar la cantidad de tiempo que el motor requiere para lograr el máximo empuje desde parado.

8.1 Medida de la fuerza de empuje y de rebote

Situación inicial para las medidas de empuje

Resultados de la simulación:

- El empuje del motor requerido para que asoma: 98 N (es decir, aproximadamente 10 kg)
- La potencia requerida por el motor: 2,1 kW. Esto ya está dado por el-motor Q80 utilizado.

El banco de pruebas con célula de carga está listo para su uso y se puede controlar y evaluar con la aplicación Android para Smartphone. Una vez que se toman las lecturas en la aplicación, esta exporta a Dropbox para luego ser evaluados con Excel.

Resultados esperados

Los resultados esperados están destinados a proporcionar información sobre el rebote cuando se cambia la velocidad. Se ensaya, el comportamiento del motor cuando se cambia la velocidad desde parado. Por lo tanto, el tiempo entre el momento de inicio de los motores, y el momento en que se mide una primera fuerza de empuje, está determinado. Este es el tiempo requerido para que el motor supere

la fricción estática, que es máxima en parada, y que después se lleva en fricción de deslizamiento. Esta transición también se denomina efecto Stick-Slip.

Medida de la respuesta al escalón a una velocidad de 1900 rev/min.

Después de ejecutar una señal PWM de 1,2 ms, se ponen en marcha los motores. El siguiente diagrama ilustra el resultado:

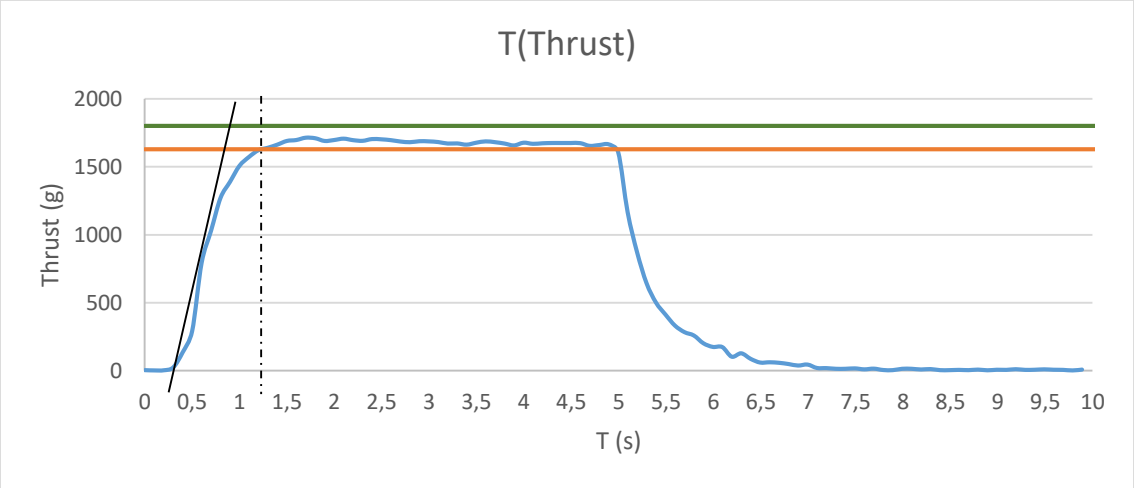


Figura 8.1 Respuesta al escalón a 1900 rev/min

300 ms después del inicio, la fricción estática es superada y 1250 ms más tarde (es decir, 1550 ms después del inicio) el empuje máximo de 1,7 kg se logró. Después de 5 segundos, se apaga el motor y el empuje logrado a 0, 500 ms más tarde.

Medida de la respuesta al escalón a una velocidad de 4000 rev/min.

Después de ejecutar una señal PWM de 1,5 ms, se ponen en marcha los motores. El siguiente esquema es el resultado de esta medida de fuerza de empuje:

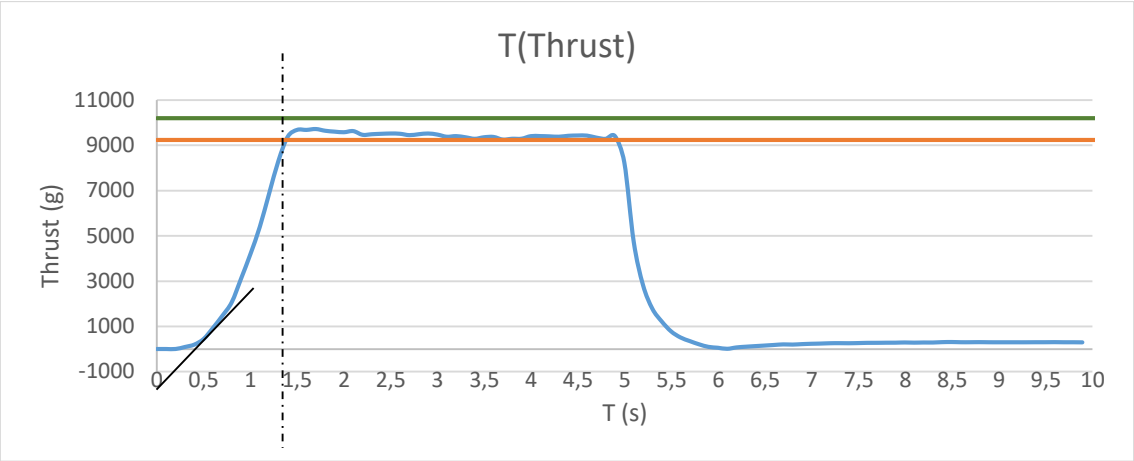


Figura 8.2 Respuesta al escalón a 4000 rev/min

Al arrancar el motor, se observa el fenómeno *stick-slip* antes mencionado. De acuerdo con el diagrama, después de 200 ms, la fricción estática es superada y después de 1200 ms alcanza la fuerza de empuje de 9,7 kg. Después de apagar el motor y retirar la velocidad, la fricción disminuye de nuevo y aumenta la fricción estática. Si la fricción estática es mayor que la fricción de deslizamiento, los motores se detienen. De acuerdo con el diagrama de la Figura 8.2, la fuerza de empuje disminuye después de apagar el motor después de aproximadamente 400 ms a cero. Repitiendo varias veces la medida también se confirma que el resultado es reproducible.

8.2 Comparación de la simulación y pruebas reales.

En otras medidas de empuje con el banco, se verificaron los resultados de la simulación del trabajo mencionado antes. Los requisitos de empuje para que el multicóptero tripulado so pone en flotación, se logró a una velocidad de 4500 rev/min. El máximo empuje de los motores de 17 kg se consigue como en la simulación en alrededor de 5500 rev/min.

Los tiempos medidos, ahora se pueden utilizar para ajustar la regulación del control de vuelo del multicóptero tripulado.

9 CONCLUSIÓN

Con el avance del banco de pruebas y la puesta en práctica de la aplicación para Smartphone, la tarea puesta de este trabajo se cumplió en todos los puntos.

Además, las medidas de empuje a distintas velocidades pudieron llevarse a cabo con éxito. De este modo pudieron verificarse los cálculos y simulación de un trabajo anterior. Las simulaciones han proporcionado un empuje de 10 kg y una velocidad de 4500 rev/min por motor para el vuelo a flotación del multicóptero tripulado.

Los tiempos muertos, los cuales han sido identificados, son datos de entrada importantes para la regulación del control de vuelo de un multicóptero.

9.1 Propuestas

Cuando se utilizó el banco para realizar medidas de empuje de la unidad de un multicóptero, se descubrieron varias características de mejora. Algunos de estos puntos pueden ser de interés para proyectos siguientes.

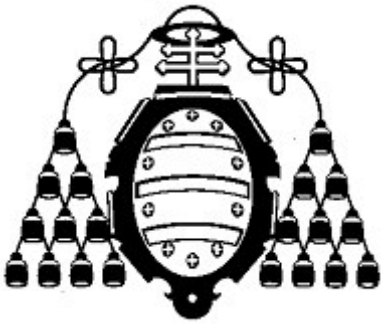
En cuanto al hardware, sería importante que hacer algunos cambios en la conectividad. Por ejemplo, es requerido poner en parada la fuente de alimentación de los motores con un interruptor. Por otra parte, sería importante hacer que el interruptor de parada de emergencia sea portátil, de modo que el usuario del banco de pruebas puede ir de manera segura con esto y en caso de emergencia, el sistema se puede anular.

La aplicación debe cambiarse para el control total del banco. Por ejemplo, sería ventajoso si los diferentes escenarios de prueba se pudieron establecer con la aplicación. Por lo tanto, la investigación de la respuesta al escalón a varias velocidades.

La comunicación entre la placa de microcontrolador y la aplicación tiene que estar mejor conectados para garantizar la seguridad aún mejor.

Eso sería, por ejemplo:

- Reacción de la placa de microcontrolador para la aplicación tan pronto como ha sido recibida una señal de arranque o parada por el mismo.
- Mejor tratamiento de casos de error (errores en la transmisión de valores medidos, por ejemplo)
- Elaboración de un diagrama interactivo para mostrar los resultados de las medidas en la aplicación
- Exportación de datos a un repositorio central en Internet, como Google Drive



UNIVERSIDAD DE OVIEDO

CENTRO INTERNACIONAL DE POSTGRADO

MASTER EN INGENIERÍA MECATRÓNICA

TRABAJO FIN DE MÁSTER

Control de Motores BLDC de un Multicopter a través una interfaz móvil

09/2016

Christian HOHMANN **Prof. Juergen WALTER**

[Firma] **[Firma]**

SPERRVERMERK

Die vorliegende Masterarbeit mit dem Titel „Steuerung von BLDC Motoren über eine mobile Benutzeroberfläche“ enthält vertrauliche Informationen.

Sie ist nur den Betreuern der Arbeit sowie gegebenenfalls dem Prüfungsausschussvorsitzenden des Fachbereichs Mechatronik zugänglich zu machen.

Die Vervielfältigung, Veröffentlichung sowie Weitergabe der Masterarbeit, im Ganzen oder in Teilen, ist grundsätzlich untersagt und bedürfen der vorherigen schriftlichen Genehmigung der Fakultät für Maschinenbau und Mechatronik der Hochschule Karlsruhe – Technik und Wirtschaft.

EIDESSTATTLICHE ERKLÄRUNG

Ich versichere hiermit wahrheitsgemäß die Masterarbeit selbstständig angefertigt sowie alle im Rahmen der Abschlussarbeit verwendeten Hilfsmittel vollständig und genau angegeben zu haben.

Ausführungen und Inhalte, die aus verschiedenen Quellen unverändert übernommen oder dem Inhalt sinngemäß angepasst wurden, wurden an entsprechender Stelle einzeln kenntlich gemacht.

Karlsruhe, 9. September 2016

Unterschrift

KURZFASSUNG

Im Rahmen dieser Arbeit entsteht die Weiterentwicklung eines Prüfstandes zur Schubkraftmessung eines Multikopterantriebes.

Im Ersten Teil der Arbeit wurde eine Überprüfung der Messfähigkeit des verwendeten Kraftmessers, einer Wägezelle, durchgeführt. Die Optimierungen umfassen das Hinzufügen eines weiteren Motors samt Luftschraube, und das Umbauen des Prüfstandes in Anlehnung an die vorgesehene Konstruktion des manntragenden Multikopters. Eine Überholung der Software des Mikrocontrollers hat zu einer Verbesserung in Hinsicht auf eine gesteigerte Effizienz geführt. Das bereits vorher genutzte Mikrocontrollerboard von *Silicon Laboratories* findet auch für diese Version des Prüfstandes Verwendung.

Mit Hilfe dieses Prüfstandes werden die Schubkraft, die Totzeit und die Verzögerungszeit des Erreichens einer Schubkraft gemessen. Darüber hinaus soll ermittelt werden, wieviel Zeit benötigt wird um, bei Änderung der Drehzahl im Schwebeflug eine gewünschte Schubkraft zu erreichen.

Die Kenntnis über die resultierende Sprungantwort, bei plötzlicher Änderung der Motordrehzahl, ist ein wichtiger Bestandteil für die Regelung der Flugsteuerung eines Multikopters. Ein Vergleich der Schubkraftmessung mit den Resultaten der Simulation aus einer anderen Masterarbeit liefert Erkenntnisse über die Zuverlässigkeit dieses Prüfstandes.

Die Auswertung des Prüfstandes über ein externes Teilsystem ist, wie schon beim Vorgängerprojekt, eine Voraussetzung, um die Sicherheit des Benutzers zu gewährleisten. Dafür entsteht mit der *Android Studio*-Entwicklungsplattform eine Smartphone Applikation. Mit dieser ist das Starten und Stoppen der Motoren, sowie die Anzeige der momentanen Drehzahl und Schubkraft möglich.

ABSTRACT

In the present master’s thesis, a test rig for thrust measurement of a multicopter-drive was advanced.

First of all, the measurement capability of the used load cell was verified. The optimizations include adding another brushless DC motor with its propeller and the rebuilding of the test bench based on the design of the man-carrying multicopter. An overhaul of the microcontroller software has resulted in an improvement in terms of better efficiency. The Silicon Laboratories microcontroller board finds also a use for this version of the test stand.

The extend of the thrust force is to be measured with this rig. So the knowledge of the resulting step response to a sudden change of the motor speed is an important component for the calibration of a Multicopter flight control. A comparison of the thrust force measurements with the results of a simulation from another master's thesis, provides insights into the reliability of this test rig.

Moreover, there was programmed a smartphone application with the Android Studio development platform. Through this application it is possible to trigger and stop the brushless DC motors. As well the application will be able to display the current revolutions per minute and thrust of the motors. This application is to satisfy a prerequisite to the safety requirements, which were already important in the previous project. This requires a system which is divided into two subsystems.

The evaluation of the test bench by with an external subsystem is a prerequisite to ensure the user's safety, as it was also in a predecessor project. Therefore, created with the android studio development platform, a smartphone application was implemented. With this App it is possible to start and stop the motors, as well to measure thrust and speed of the same.

RESUMEN

En Este Trabajo el desarrollo de un banco de pruebas para medir el empuje de hélices se plantea.

Para este, primero una revisión de la capacidad de medición de la célula de carga utilizada se ha realizado. Las optimizaciones incluyen la adición de un otro motor con una hélice y la reconstrucción del banco de pruebas, en base a la propuesta de construcción del multicopter tripulado. Una revisión del software de la placa del microcontrolador ha conducido a una mejora, en términos de aumento de la eficiencia. La placa de desarrollo de *Silicon Laboratories* utilizado anteriormente, también encuentra un uso para esta versión del banco de pruebas.

Con el uso de este banco de pruebas, se desea medir la fuerza de empuje de los Motores BLDC. El conocimiento de la respuesta al escalón resultante al cambio repentino de la velocidad del motor, es un componente importante para el reglaje de vuelo de un multicopter. La comparación de las mediciones de fuerza de empuje con los resultados de una simulación de un otro trabajo de fin de master, se usa para comprobar la confiabilidad de este banco de pruebas.

Como en un proyecto anterior sobre este banco de pruebas, el banco se va a dividir en dos subsistemas. Por esto una aplicación para Smartphone se ha desarrollado con la plataforma de desarrollo de Android Studio. Con esto son posible un arranque y la parada de los motores y también la medición de la velocidad actual y de la fuerza de empuje.

Inhaltsverzeichnis

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

API	Application Programming Interface
ASCII	*American Standard Code for Information Interchange*
BLDC	*Brushless Direct Current*
CFK	Carbonfaserverstärkter Kunststoff
CSV	*Comma-separated Values*
DMS	*Dehnungsmessstreifen*
ESC	*Electronic Speed Control*
IDE	*Integrated Development Environment*
LCA	*Load Cell Amplifier*
LiPo	*Litium Polymer*
MAC-Adresse	Media Access Control Adress
MCU	*Microcontroller Unit*
MSA	*Messsystemanalyse*
PCA	*Programable Counter Array*
PGA	*Programmable Gain Amplifier*
SDK	*Software Developement Kit*
SPP	*Serial Port Protocoll*
UART	*Universal Asynchronous Receiver Transmitter*
USB	*Universal Serial Bus*

1 EINLEITUNG

Multicopter sind nicht nur im Hobbybereich sehr populär, sondern sind auch bei immer mehr professionellen oder technischen Anwendungen von Interesse. Aktuell bewegt man sich bei den Ausmaßen der Flugobjekte noch im Modellbaubereich. Jedoch könnten hubschraubergroße Multicopter auch für den Personentransport in Erwägung gezogen werden. Die hier dargestellte Arbeit behandelt einen Teil der nötigen Forschungen, um ein sicheres und ausreichend dimensioniertes manntragendes Fluggerät zu entwickeln.

Der Ausfall eines Rotors kann bei einem Multikopter mit sechs Luftschrauben, durch den Betrieb der übrigen Rotoren kompensiert werden, und das Flugobjekt kann immer noch sicher gelandet werden. Durch die hohe Anzahl an Rotoren ist ein Multikopter einfacher zu fliegen und die Vibrationen sind schwächer als bei einem Helikopter.

Der bemannte Multikopter Flug ist unter anderem deswegen interessant, weil die Erfahrung der Vogelperspektive vom Anwender selbst erlebt werden kann. Im Vergleich zu einem herkömmlichen Helikopter besitzt der Multicopter kein Heckrotor. Ebenfalls ist es nicht erforderlich, einen komplizierten Taumelantrieb für den Vortrieb oder Steuerstangen für den Auftrieb zu implementieren. Ein Multicopter steuert Lage, Vortrieb und Orientierung allein über die Rotordrehzahl. Die Konstruktion und Wartung ist dadurch relativ simpel. Da die Antriebsenergie ausschließlich elektrisch ist, ist die Antriebsweise sehr leise und umweltschonend.

Basierend auf einer Studienarbeit für die Entwicklung eines Hexakopters im Modellflug-Format, wurde ein sechsarmiger Multikopter für den bemannten Flug in Hard- und Software entwickelt. Im Umfang der vorliegenden Masterarbeit wird nun ein Prüfstand, zur Schubkraft- und Drehzahlmessung mit einem Tragarm und zwei Motoren weiterentwickelt.

Bei der Entwicklung von Multikoptern müssen mehrere Faktoren beachtet werden. Wichtig sind die Positionierung, die Auswahl der elektrischen Antriebe, die Regelung des Systems und die Kenntnis über das Sprungverhalten der Schubkraft des Antriebs, beziehungsweise des Gesamtsystems. Die Regelung wird mit Hilfe einer Steuereinheit gestaltet, welche ortsfest am Multikopter fixiert ist. Der Begriff Steuereinheit, oder Flugsteuerung, hat sich in der Multikopterszene eingebürgert, selbst wenn die Hardware mit ihrer Software auch Regelungsaufgaben übernimmt. Damit diese Regelung für ein stabiles Flugverhalten funktioniert, interessieren die Anstiegszeit der Luftschrauben, bei plötzlicher Änderung der Motorendrehzahl und der momentane Schub, bei Erreichen dieser.

Im Rahmen dieser Arbeit wird ein Prüfstand zur Auftriebsmessung einer Luftschraube erweitert. Hierfür werden einführend die Problemstellung und anschließend der aktuelle Stand der Technik beschrieben. In diesem Abschnitt soll

deutlich werden, wie der aktuelle Prüfstand funktioniert und an welchen Stellen Verbesserungen notwendig sind. Bei der anschließenden Aufgabestellung wird weiter darauf eingegangen, welche Änderungen eingeführt werden. Bei einer statischen Analyse ohne Luftschraube soll eine Messfähigkeitsanalyse beweisen, inwiefern der verwendeten Kraftmesser zuverlässige Werte liefert. Anschließend wird erläutert, wie sich der Prüfstand, in Hinsicht auf Konstruktion und Software ändert. Ein wichtiger Bestandteil dieser Arbeit ist die Entwicklung einer Smartphone Applikation, mit der der Prüfstand gesteuert und ausgewertet wird. Schließlich werden mit dem entwickelten Prüfstand Schubkraftmessungen durchgeführt, um das Sprungverhalten bei raschem Ändern der Drehzahl zu prüfen. Die Ergebnisse werden mit denen der Simulation einer vorangegangenen Masterarbeit verglichen.

2 PROBLEMSTELLUNG

Durch die aktuelle Konstruktion des Prüfstandes ist es nicht möglich realitätsnah die Schubkraft zu untersuchen, welche auf den Tragarm ausgeübt wird. Die Position des Kraftmessers zwischen Tragarm und Motor ist ungünstig.

Die Struktur der aktuellen Software unterbindet eine Entkoppelung von Motorenansteuerung und Aufnahme der Messdaten. Das verhindert die Realisierung von mehreren Schubkraft-Messszenarien, die im Zusammenhang dieser Arbeit notwendig sind.

Das Starten der Messungen durch einen Taster direkt am Versuchsaufbau entspricht nicht der Aufteilung in zwei Teilsysteme, welche für die Sicherheit gegeben sein muss. Bisher wurden keine Maßnahmen für einen Notaus implementiert.

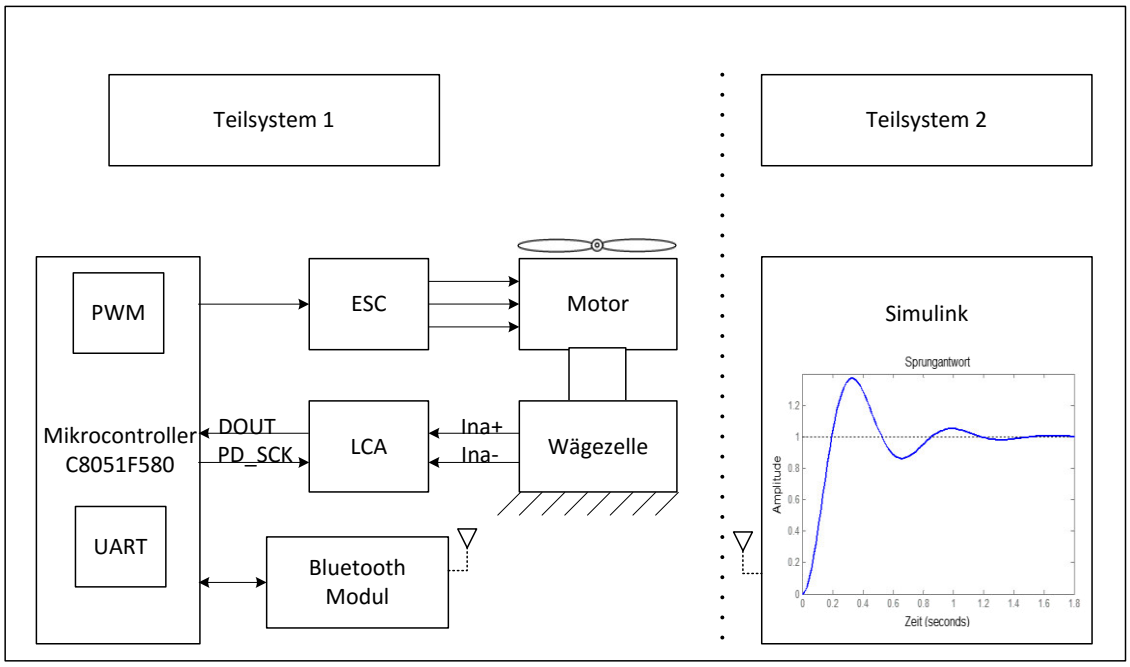


Abbildung 2.1 Blockschaltplan der Systemfunktionen des aktuellen Prüfstandes, [1]

Im Blockschaltplan (Abbildung 2.1) sind beide Teilsysteme zu erkennen. Zum einen (links) das Mikrocontroller-Board welches die Wägezellen-Messdaten über den Messverstärker (LCA) verarbeitet und ein pulsweiten-moduliertes Signal (PWM-Signal) zur Ansteuerung der Motoren generiert. Zum anderen (links), nach der Datenverarbeitung, die Übertragung dieser über UART und Bluetooth an Simulink.

3 VORGÄNGERPROJEKTE

Bei mehreren Vorgängerprojekten für Studien- und Abschlussarbeiten wurden verschiedene Multikopter-Konzepte ausgearbeitet und konstruiert. Darunter sind Konzepte im Modellbau-Format sowie Vorläufer für den bemannten Multikopter-Flug.

Der VC05

Abbildung 3.1 VC05, [2]

Mit der Entwicklung, Konstruktion und Leistungsauslegung des VC05 wurde sich im Rahmen einer Studienarbeit befasst. Bei der Konstruktion im Modellbau-Format wurde besonderen Wert auf die Leichtbaukonstruktion mit karbonfaserverstärktem Kunststoff gelegt. Die entwickelte Konstruktion und Flugsteuerung dient als Grundlange für weitere Projekte an der Hochschule, wie zum Beispiel die Bachelorarbeit „*Airpaint*“, für die ein System für den autonomen Flug entwickelt wurde. Damit ist es möglich dem Multikopter ein zu zeichnendes Motiv vorzugeben, welches dann eigenständig an eine Wand gesprüht wird. Die Wespen-ähnliche und Sandwich-Bauweise des VC05 dient auch als Vorlage für den Multikopter für den bemannten Flug und somit als Grundlage für den Prüfstand.

Bemannter Flug – Volocopter

Abbildung 3.2 Volocopter, [3]

„Volocopter ist die Bezeichnung für ein neuartiges Hubschrauber-Konzept für elektrisch angetriebene personentragende Multikopter […]“, vgl. [4]. Es ist eine Zweisitzer-Maschine, mit 18 elektrisch-angetriebenen, horizontal ausgerichteten Luftschrauben. Mit Hilfe des Vorläuferprojekts der Hochschule, VC25-A, konnten ebenfalls Forschungsarbeiten betrieben werden, welche für den bemannten Flug des Volocopters einen wesentlichen Ausgangspunkt bilden. Unter anderem für die Navigation und das Steuerungssystem.

4 SICHERHEIT

Die Sicherheit ist bei der Verwendung von Motoren mit einer Leistung von 5,5 kW und Luftschrauben mit einem Durchmesser von etwa 30 cm, sehr wichtig. Da entsprechend die Verletzungsgefahr sehr hoch ist, wird bei der Verwendung des Prüfstandes, der Benutzer mit Hilfe von Trennscheiben vor den Antrieben geschützt (siehe Abbildung 4.1 und Abbildung 4.2).

Ein weiterer wichtiger Punkt ist die Aufteilung des Systems in zwei Teilsysteme. Dabei soll der Prüfstand von einem Tragbaren Gerät aus gesteuert und ausgewertet werden können.

Weitere Maßnahmen, wie die Einführung eines Notausschalters, werden in dieser Arbeit noch beschrieben.

Abbildung 4.1 Trennwand

Abbildung 4.2 Trennscheibe

5 STAND DER TECHNIK

Die Bachelorarbeit *„Entwicklung und Aufbau eines Prüfstandes zur Auftriebsmessung von Luftschrauben“* [1] umfasst den wesentlichen Aufbau des Prüfstandes und die Entwicklung einer ersten betriebsfähigen Software. Dessen Stand der Technik wird in diesem Abschnitt erläutert.

5.1 Entwicklungsumgebung

Die Integrierte Entwicklungsoberfläche (IDE), Keil µVision 5 wird für die Programmierung des Mikrocontroller-Boards eingesetzt. Diese kostenpflichtige Software beinhaltet bereits in der kostenlosen Evaluierungsversion nützliche Funktionen. Die integrierten Entwicklungswerkzeuge ermöglichen eine Programmierung in den Sprachen C und Assembler. Mit Hilfe der integrierten Gerätedatenbank, kann die Konfiguration, mittels *Configuration Wizard*[2], für die jeweilige Mikrocontroller Unit (MCU) durchgeführt werden. Um eine erstellte Software-Applikation auf den Flash-Speicher des im nächsten Abschnitt genauer beschriebenen F580 MCU herunterladen zu können, muss darüber hinaus ein Geräte-Treiber für einen USB-Debugger installiert werden. Mit Hilfe der *Silicon Laboratories* Unterstützung können Beispielapplikationen, welche in der IDE von Keil nicht integriert sind, installiert und als Vorlage für den C-Code verwendet werden.

5.2 Mikrocontroller-Evaluierungsboard C8051F580-TB

Der Mikrocontroller stellt für den Multikopter eine zentrale Komponente dar. Auch für den Versuchsaufbau ist er ein wichtiger Bestandteil. Er muss über die entsprechende Peripherie für Verarbeitung der Ein- und Ausgangssignale verfügen. Des Weiteren wird eine bestimmte Rechenleistung vorausgesetzt, damit alle Befehle und Aufgaben in den geforderten Zeitzyklen bearbeitet werden können.

[1] Angefertigt von Nermin Dedic, vgl. [1]

[2] Mit dem *Configuration Wizard* wird die Entwicklung des Initialisierungs-Codes für den Mikrocontroller automatisiert

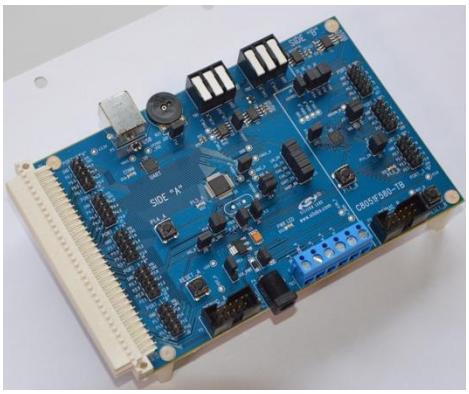


Abbildung 5.1 Mikrocontroller-Board

Ausgewählt wurde der 8-bit Mikrocontroller C8051F580-TB vom Hersteller *Silicon Laboratories* (siehe Abbildung 5.1). Die Mikrocontrollereinheit und die Peripherie sind Bestandteil des Evaluierungsboards.

Eine besondere Eigenschaft des Boards sind zwei voneinander unabhängige Seiten A und B. Beide Seiten werden von derselben Spannungsquelle versorgt und verwenden denselben Typ Mikrocontroller (8051). Die Peripherie des Chips ist auf dem Blockschaltbild in Abbildung 5.2 zu erkennen.

Unter anderem die UART-Schnittstelle, mehrere *Programmable Counter Arrays* (PCA) und verschiedene Timer werden für die Entwicklung der Software dieses Projektes verwendet.

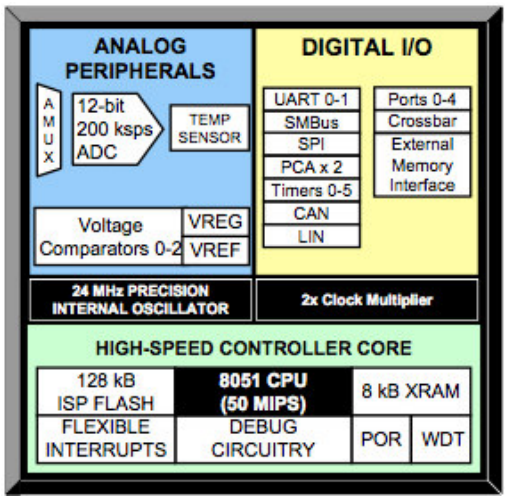


Abbildung 5.2 Peripherie der MCU 8051F580

In den nächsten Abschnitten werden die Komponenten und die Funktionsweise der Messkette näher betrachtet. Dabei wird zunächst auf die Aktorik, anschließend auf die Sensorik und die Übertragung der Messdaten eingegangen.

5.3 Hardware

5.3.1 Aktorik

Motoren

Der bürstenlose Gleichstrommotor *Q80* der Firma *Hacker Motor GmbH* wird genutzt, um die Luftschraube anzutreiben. Über eine Aluminium-Platte und vier Bohrungen wird der Motor an der Wägezelle befestigt.

Ein Auszug aus den technischen Daten, vgl. [5]:

- Leistungsbereich: maximal 3000 W für 15 s
- Gewicht: 610 g
- Max. zulässige Drehzahl: 8000 U/min
- Verwendung ausschließlich mit Drehzahlsteller

Abbildung 5.3 Q80 BLDC Motor, [6]

Die Stromversorgung wird mit 2 Lithium Polymer-Akkus (LiPo) gewährleistet und der Motor darf nur über, für den Motor geeignete, elektronische Drehzahlsteller, betrieben werden.

Elektronischer Drehzahlsteller (ESC): MasterSpin 125 OPTO

Der ESC wird dem Q80 Motor vorgeschaltet und steuert diesen. Er verfügt über zwei Hauptstromkabel (schwarz und rot), an welche die Versorgung angeschlossen wird, sowie drei Motorsteuerleitungen die zum Motor führen. Außerdem besitzt der ESC einen Signalanschluss. Ein ESC funktioniert ähnlich wie ein Transistor. Das Steuersignal wird über den Signalanschluss in ein Leistungssignal für den Motor umgewandelt und wird vom Mikrocontroller zur Verfügung gestellt.

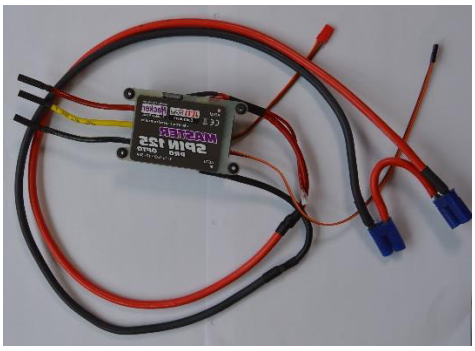


Abbildung 5.4 Master Spin 125 OPTO

Laut Datenblatt, muss der ESC, vor Verwendung mit dem Motor, erst initialisiert werden und mit einer 50 Hz PWM betrieben werden. Ein PWM-Signal zeichnet sich dadurch aus, dass sich innerhalb der festen Periodendauer (hier 50 Hz), ein bestimmtes Verhältnis aus Einschalt- und Ausschaltzeit ergibt. Hierbei wird innerhalb einer Periode nur einmal zwischen High und Low gewechselt. Das Zeitliche Verhältnis zwischen Einschalt- und Ausschaltzeit nennt sich Tastverhältnis (engl. *duty-cycle*). Der Vorgang der Initialisierung besteht darin, dass zunächst die maximale und anschließend die minimale Pulsweite (2ms bzw. 1ms) übermittelt werden.

Die JETI-Box ist ein „universelles Kommunikationsendgerät“, [7]. Sie ermöglicht die einfache Einstellung und das Lesen von Daten eines Drehzahlstellers, wie z.B. Temperatur oder momentane Motordrehzahl. Weitere Informationen über die JETI-Box werden im Datenblatt unter [8] bereitgestellt.

Pulsweitenmodulation

Die ESCs müssen für die bürstenlosen Gleichstrom-Motoren (BLDC), PWM Signale mit einer Periodendauer von 20ms (50Hz) generieren. Mit Hilfe des Datenblatts der F580 MCU kann die Funktionsweise der Pulsweitenmodulation im Mikrocontroller und das einzustellende *Special-Function-Register* ermittelt werden.

Die Peripherie des Mikrocontrollers stellt zwei 16-Bit *Programmable Counter Arrays* (*PCA0* und *PCA1*, siehe Abbildung 5.5) bereit. Beide Einheiten verfügen nochmals über 6 separate 16-Bit *Capture-/Compare*-Module, welche es ermöglichen 6 verschiedene und unabhängige PWM Signale zu generieren.

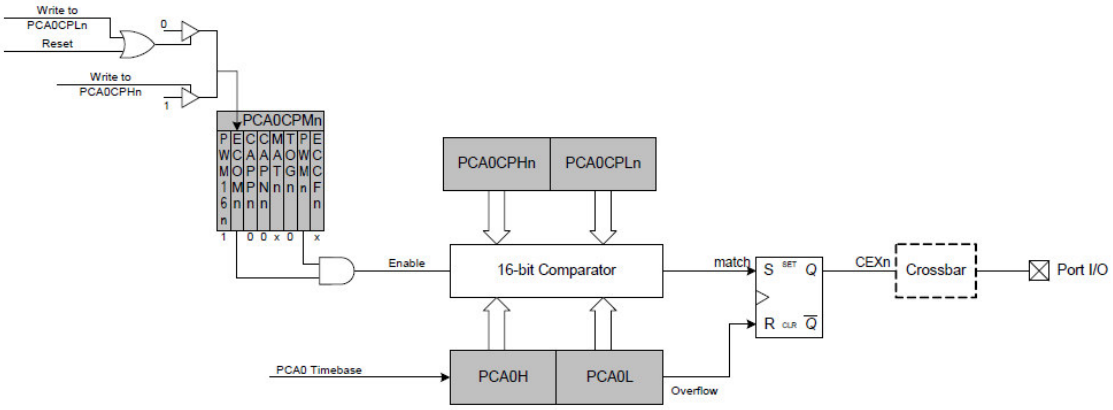


Abbildung 5.5 Capture/Compare Modul im 16-Bit PWM Modus

Der aktuelle Zählerstand der *PCA0*-Einheit, welcher im 16-Bit Modus eingestellt wird, zählt mit jeder Taktflanke (Logisch 1) hoch. In den beiden Vergleichsregister *PCA0CPL0*/*-H0* wird dieser Zählerstand mit einem vorgegebenen Zielwert verglichen. Sobald Zähler und Zielwert übereinstimmen, wird der dem Modul zugewiesene *CEX0*-Pin auf High gesetzt. Im *Configuration Wizard* kann dieser Pin einem Input- oder Output-Port zugeteilt werden. An diesem Port wird das PWM-Signal ausgegeben und kann mittels Oszilloskop überprüft werden. Beim Überlauf des *PCA0* (erstes Zählregister), von 65535 auf 65536, wird der *CEX0*-Pin wieder auf Low gesetzt. In diesem Zustand verweilt der Pegel bis wieder eine Übereinstimmung der Vergleichswerte im *PCA0* festgestellt wird. Somit wird bei jedem Überlauf der Zählvorgang neu gestartet. Durch das Umschalten mit spezifischem Zeitinterwall, ergibt sich ein Rechtecksignal mit einem entsprechenden *Duty-Cycle*.

Damit keine Timing-Probleme auftreten, wird zuerst das Low Byte (*PCA0CPL0*) und anschließend das High Byte (*PCA0CPH0*) im Vergleichsregister beschrieben. Somit wird verhindert, dass das Tastverhältnis auf dem Input/Output-Port ausgegeben wird, bevor das Vergleichsregister mit dem 16 Bit-Wert überschrieben ist.

$$PWM_{16\,Bit} = \frac{65535 - PCA0CP0}{65535}$$

Formel 5.1 Tastverhältnis PWM-Signal

Zur Ermittlung des Tastverhältnisses wird Formel 5.1 verwendet.

5.3.2 Sensorik

Wägezelle

Die Wägezelle ist die wichtigste Komponente in der Messkette elektromechanischer Messgeräte. Sie bildet die Schnittstelle zwischen mechanischer Eingangsgröße (Kraft) und elektrischer Ausgangsgröße (Spannung), welche proportional umgerechnet wird. Eine Wägezelle ist demnach auch ein Kraftsensor, mit dem Unterschied, dass das Gewicht in Kilogramm und nicht die Kraft in Newton gemessen wird. Durch das, im Federkörper enthaltene Metall-Stück, gleicht die Wägezelle einem üblichen Kraftmesser. Dessen Form ändert sich leicht bei Anbringung einer Kraft (elastische Verformung). Diese Verformung kann mit Hilfe von Dehnungsmessstreifen erfasst und in ein elektrisches Signal umgewandelt werden. Je nach Anwendung gibt es verschiedene Formen von Wägezellen. Zum Beispiel ein Doppelbiegebalken (siehe Abbildung 5.6), welcher für kleine Lasten gedacht ist (2-5kg).

Abbildung 5.6 Miniatur-Wägezelle H07A von Bosche Wägetechnik, [9]

Kenngrößen der Wägezelle sind Nennlast (Bereich der Betriebslast), Grenzlast (darf nicht überschritten werden) und Kennwert (Ausgangssignal bei Nennlast).

Für die Erfassung der Schubkraft von Luftschrauben wird eine Wägezelle vom Typ *S40S-G3-0020* verwendet (siehe Abbildung 5.7). Aufgrund der geeigneten Nennlast von 20 kg, der geeigneten Messwertauflösung und der Kalibrierfähigkeit in Gewichtseinheiten, ist die S-förmige Wägezelle für diese Anwendung geeignet. Wie bei einer Sprungfeder kann der S-förmige Sensor bei Druck- bzw. Zugbeanspruchung verformt werden. Nach Beendigung der Beanspruchung, kehrt diese allerdings wieder ein seine Ausgangsposition zurück. Durch die S-Form und die physikalischen Eigenschaften, ist dieser Sensor somit für dynamische Zug-/Druck-Beanspruchungen geeignet.

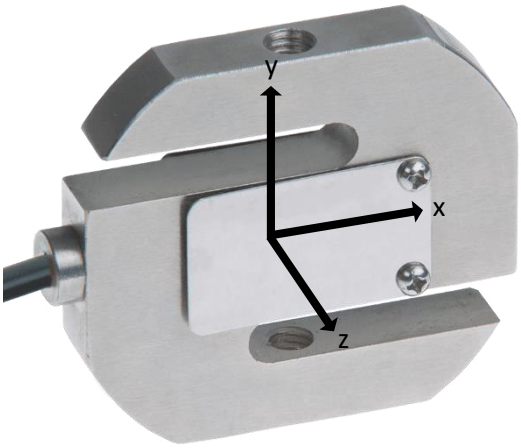


Abbildung 5.7 *Bosche* Wägezelle S40S-G3-0020 mit Koordinatensystem

Bei der Verwendung einer Wägezelle, insbesondere bei der Schubkraftmessung, muss darauf geachtet werden, dass die Grenzlast nicht überschritten wird. Durch die Überschreitung können mittels plastische Verformung nicht-reversible Schäden entstehen.

Dehnungsmessstreifen (DMS) der Wägezelle

DMS sind das Kernstück unterschiedlicher Kraftaufnehmer. Unter anderem werden sie in Sensoren wie einem Drehmomentaufnehmer oder Zug-/Druck-Kraftaufnehmer für das Messen von Kräften, Drücken oder Drehmomenten verwendet. Das Messprinzip ist mit dem einer Federwaage zu vergleichen, welche sich in einen Gleichgewichtszustand bringt und die Masse eines Objektes anzeigen kann. Eine spezielle Messfeder (Stab oder Ring), deren Federsteifigkeit maßgebend für die Anwendung ist, soll die Funktion der Feder übernehmen. Damit die Auslenkung des gemessenen Materials möglichst gering bleibt, wird diese besonders groß definiert (c > 5x104 N/mm). Das Hooksche-Gesetzt, aus Formel 5.2, legt die Grenzen einer angewendeten Gewichtskraft fest. Es besagt, dass „die Wirkende Kraft F, linear von der Dehnung Δl abhängig ist“, [10], wobei D die Federkonstante beschreibt.

$$F = D \cdot \Delta l$$

Formel 5.2 Das Hooksche-Gesetz

Die Streifen werden auf dem Trägermaterial angebracht, z.B. durch Kleben. Technisch gesehen sind DMS Metallfolien, in welche mäander-oder ringförmige Leiterbahnen geätzt werden (Abbildung 5.8). So werden auf kleinster Fläche möglichst lange Leitungen und auf kleinste Längenänderungen große

Widerstandsänderungen erhalten. Damit das Messgitter geschützt ist, wird eine zweite dünne Kunststoffschicht auf der Oberseite angebracht.

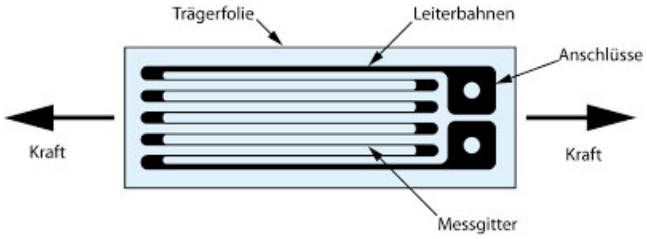


Abbildung 5.8 Folien – Dehnungsmessstreifen, [11]

Um die mechanische Verformung des Sensors festhalten zu können, werden in der Wägezelle vier DMS eingesetzt, welche sich je nach mechanischer Beanspruchung verformen und ein elektrisches Signal ausgeben.

Die Berechnung des Widerstandes unbelasteter DMS, kann mit der Formel 5.3 geschehen. Bei einer Dehnung des DMS nimmt der Widerstand zu und bei einer Stauchung nimmt der Widerstand ab.

$$R = \boldsymbol{\rho} \cdot \frac{\boldsymbol{l}}{\boldsymbol{A}} = \boldsymbol{\rho} \cdot \frac{\boldsymbol{4} \cdot \boldsymbol{l}}{\boldsymbol{D^2} \cdot \boldsymbol{\pi}}$$

Formel 5.3 Widerstand unbelasteter DMS

Wobei gilt:

- ρ: spezifischer elektrischer Widerstand
- l: Drahtlänge
- A: Querschnittsfläche
- D: Draht-Durchmesser

Die relative Widerstandsänderung wird mit dem k-Wert und der relativen Längenänderung beschrieben und wird wie folgt dargestellt:

$$\frac{\Delta R}{R} = k \cdot \frac{\Delta l}{l}$$

Formel 5.4 Relative Widerstandsänderung

Da der *k*-Faktor proportional zur Längenänderung ist, resultiert bei großem *k* folglich eine große Widerstandsänderung. Somit wird ebenfalls die Amplitude des Signals größer. Der *k*-Wert ist vom Material abhängig, mit welchem die Dehnung

bestimmt wird und ist bei den meisten Metallen *k*=2. Äußere Faktoren, wie die Temperatur, haben einen Einfluss auf die DMS und können die Messwerte beeinflussen (Genauigkeit zwischen 1% und 5% bei 20°C).

Weitere Einflüsse und deren Folgen sind:

- Hysterese bei starker Dehnung (Zeitverzug des Ausgabewertes)
- Feuchtigkeit
- Instabile Spannungsversorgung
- Wärme durch Thermospannung oder Widerstandsänderungen in der Zuleitung oder den DMS (Linearität nicht mehr gegeben)
- Magnetische Felder (Beeinflussung des Signals beim Messbrückenausgang)

Die hohe Robustheit der Wägezelle ermöglicht einen Einsatz in der Industrie. Die Nennlasten können von einem Kilogramm bis zu mehreren Tonnen reichen. Bei simpler Anwendung wird eine analoge Signalverarbeitung vorgezogen, wobei heutzutage, um eine hohe Genauigkeit zu erreichen, auch A/D-Wandler für die digitale Verarbeitung eingesetzt werden.

Wheatstone'sche Messbrückenschaltung

Durch die Anordnung der DMS als *Wheatstone*-Brücke kann die einwirkende Kraft in Form einer Widerstandsänderung ausgegeben werden. Es können mit der Wheatstone'schen Messbrücke Bewegungen im µm-Bereich gemessen und in ein interpretierbares Signal umgewandelt werden. Das ist durch die Messung von elektrischen Widerständen und ohmschen Widerstandänderungen möglich. Die Vollbrückenschaltung erreicht die bestmögliche Kompensation von mechanischen Stör- und Temperatureinflüssen.

Wägezellen-Messverstärker (LCA, Abbildung 5.9**)**

Der Wägezellen-Messverstärker ist ein *Breakout*-Board für den integrierten Schaltkreis *HX711*. Dieser konvertiert das analoge 24Bit-Signal der Wägezelle in digitale Werte. Somit ist ein einfaches Auslesen sowie Auswerten der Messwerte gewährleistet. Nach dem Anschließen an das MC-Board können so die Variationen der Wägezellen-DMS ausgelesen und nach einiger Kalibrierung, können genaue Gewichtsdaten abgelesen werden.

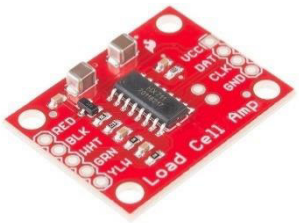


Abbildung 5.9 *Load Cell Amplifier* HX711, [12]

5.3.3 Datenübertragung

Übertragung der Daten mittels UART

Ein *Universal Asynchronous Receiver Transmitter* (UART) ist ein Kommunikationsprotokoll, welches die Grundlage für den in der Industrie oft verwendeten RS232-Standard bildet. Um die Messwerte der Wägezelle graphisch mit Simulink darstellen zu können, wird die UART-Schnittstelle verwendet. Die F580-MCU verfügt mit der UART-Peripherie über eine Kommunikationsschnittstelle mit voll-Duplex Funktion. Damit sich Sender und Empfänger verständigen können, wird eine gemeinsame Baudrate (Bit/s) festgelegt.

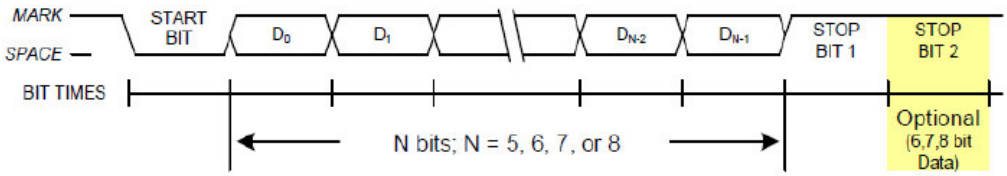


Abbildung 5.10 UART-Byteframe

In Abbildung 5.10 wird das UART-*Byteframe* dargestellt. Die fünf bis acht Datenbits werden von einem Start-Bit und einem Stopp-Bit eingegrenzt.

Bluetooth-Übertragung

Für die mobile Applikation, wird eine Drahtloskommunikation eingerichtet, mit der eine Verbindung zwischen F580-MCU und PC hergestellt wird. Diese geschieht mittels einer seriellen Datenübertragung per Bluetooth, die von zwei EGBT Bluetooth-Modulen übernommen wird (Abbildung 5.11).

Bei der Auswahl der Bluetooth Module wurde vor allem darauf geachtet, dass diese ohne weiteres eingesetzt werden können. Das heißt ohne die Notwendigkeit einer vorherigen speziellen Hardwarekonfiguration, um eine Verbindung zwischen den Geräten herzustellen.

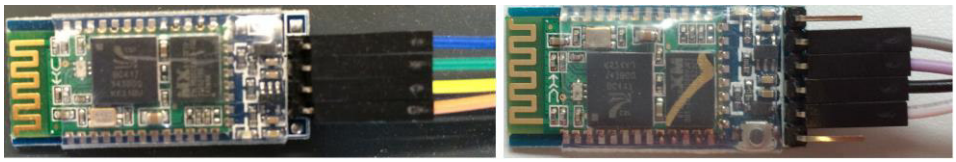


Abbildung 5.11 EGBT-046S (Slave) und EGBT-045MS (Master)

Für den Anschluss der Bluetooth-Module werden neben der 5V Versorgungsspannung und der Masse noch die beiden Anschlüsse *RXD* und *TXD* für die UART-Übertragung benötigt. Die Konfiguration von *Master* und *Slave* muss so erfolgen, dass sobald eine Spannung von 5V anliegt, die Verbindung zwischen beiden Modulen hergestellt wird.

Für die Konfiguration und die Bereitstellung der USB-Verbindung zum PC, wird eine *CP2102 USB-to-UART* Brücke[3] verwendet (Abbildung 5.12). Damit können alle Datentransfers zwischen USB und UART gesteuert sowie USB-Befehle für den UART generiert werden. Für den USB-Anschluss sind keine weiteren Maßnahmen erforderlich, da der Treiber für die *CP2102*-Brücke bereits vom Evaluierungsboard installiert und durch den Computer ein *COM*-Port zugewiesen wurde.

Abbildung 5.12 CP2102-Kommunikationsbrücke

Nun können die Bluetooth-Module über AT-Konfigurationsbefehle konfiguriert werden. Dabei müssen jeweils die *TXD*- und *RXD*-Pins der CP2102-Brücke und der Bluetooth-Module gekreuzt werden.

Die Konfiguration des Slave Moduls beinhaltet folgende Schritte:

- Verbindungstest zum Modul
- Baudrate einstellen (meistens 9600 bps)
- Gerätenamen ändern
- Pin für die Verbindung festlegen

Bei jedem Schritt muss auf eine Bestätigung (*OK*) des Moduls gewartet werden.

So auch beim Master-Modul. Bevor das Modul angeschlossen wird, muss der Konfigurationsmodus durch Betätigen eines Tasters am Rand des Moduls gestartet werden. Damit bei der späteren Verwendung die Verbindung bei Spannungszuführung zwischen den Modulen automatisch hergestellt wird, muss

[3] Von der Firma *Silicon Laboratories*

das Master-Modul dementsprechend eingestellt werden. Dadurch wird verhindert, dass eine Verbindung manuell hergestellt werden muss.

Die Konfiguration des Master-Moduls wird wie folgt durchgeführt:

- Verbindungstest
- Gerätenamen ändern
- Verbindungsdaten (gemeinsame Baudrate, Stoppbits, Parität)
- Pin für die Verbindung
- Mastereigenschaften für dieses Modul einstellen
- SPP-Protokoll initialisieren
- Alle verfügbaren Bluetooth-Geräte auflisten
- Kopplung, Verbindung und Exklusivverbindung mit Slave

Nach der Konfiguration sind die Bluetooth-Module, welche die Verbindung zwischen PC und F580 MCU herstellen, einsatzbereit und die Datenübertragung mittels *Bluetooth Serial Port Protocol* (SPP) initialisiert.

5.4 Externe Messdatenaufnahme mit Matlab Simulink

Die Auswertung der Schubkraftmessung wird von einem Computer aus durchgeführt. Zunächst wird erklärt, wie mit Simulink die Schubkraftmessung visualisiert wird.

Die Matlab Simulink-Simulationssoftware bietet, mit der Aggregation von Modellblöcken, die Funktion einen seriellen Datenstrom über den COM-Port des Computers herzustellen. Zielsystem und erfasste Messdaten können mit Hilfe von benutzerdefinierten Algorithmen in Matlab und über die graphische Benutzeroberfläche in Simulink gesteuert und visualisiert werden.

Mit Simulink und dessen Funktionsblöcken, zum Beispiel *Serial Configuration*, können Daten über UART und die CP2102-Bücke vom COM-Port empfangen und ausgelesen werden. Dazu ist keine Erstellung von Programmcode notwendig. Zusätzlich kann Matlab, empfangene Daten, welche im Matlab-*Workspace* gespeichert werden, in Excel oder andere Datei-Formate exportieren. Auf nachfolgender Abbildung 5.13, sind links die Funktionsblöcke in einem Simulink-Modell dargestellt und rechts Funktionsblöcke zur Auswahl in einem Menü.

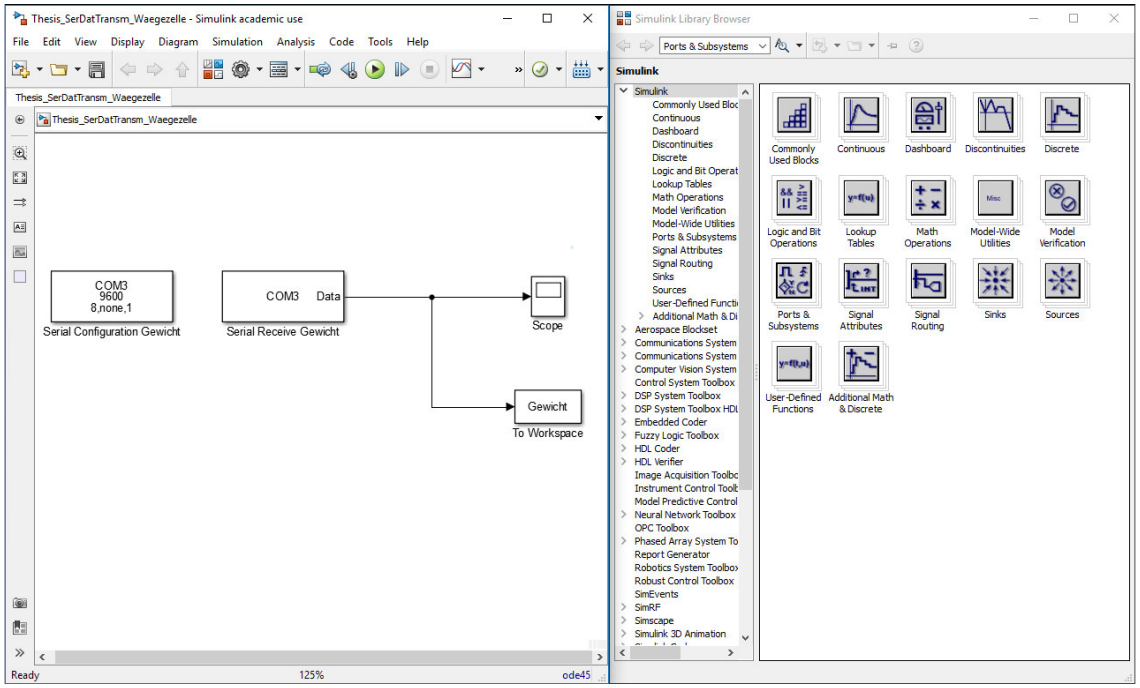


Abbildung 5.13 Funktionsblöcke der Seriellen Datenübertragung in Simulink und Toolbox

5.5 Software

Für die Gewährleistung der Sicherheit, müssen Software und Prüfstand in zwei Teilsysteme unterteilt werden. In diesem Abschnitt werden die Software-Bausteine erklärt.

Während eines Prüflaufes, wird fünf Sekunden lang eine Schubkraft-Messung durchgeführt. Gleichzeitig erzeugt das Mikrocontroller-Board (MC-Board) ein PWM Signal, um die Motoren zu starten und nach fünf Sekunden wieder zu stoppen. Die Ergebnisse werden mit Simulink dargestellt. Mit verschiedenen Prüfläufen sollen mehrere PWM-Signale eingestellt, und somit verschiedene Sprungantworten gemessen werden können.

Die Messkette und somit die 3 Hauptteile der Software sind:

- Ansteuerung des BLDC Motors über den ESC
- Einlesen der Messdaten, welche von der Wägezelle bereitgestellt und über die serielle Verbindung des LCA an das MC-Board übertragen werden
- Übertragung der Messdaten an Simulink mittels UART und Bluetooth

Das Evaluierungsboard beginnt mit der Initialisierung der Peripherie, sobald es an die Spannungsversorgung angeschlossen wird. Der Drehzahlsteller des Motors benötigt ebenfalls eine Initialisierung, welche unmittelbar nach dem Startvorgang der MCU durch die Software durchgeführt wird. Die Schubmessung von fünf

Sekunden kann nach Abschluss des Tariervorgangs der Wägezelle, durch Betätigen des Tasters am Port 1.4 gestartet werden. Durch Drücken des Tasters wird die Pulsweite am Port P0.0 verändert und der ESC stellt eine entsprechende Drehzahl für den Motor ein.

Die Messwertaufnahme erfolgt gleichzeitig mit dem Anlaufen des Motors und wird als Array in einem externen Speicher der MCU gesichert. Die Übertragung der Messdaten über UART erfolgt sobald die Rücksetzung der Pulsweite den Motorstillstand (nach fünf Sekunden) herbeiruft.

In der Tabelle 5.1 wird der Portbelegungsplan der MCU dargestellt, welcher sich während der Implementierung der Software ergeben hat. Diese Ports wurden mit Hilfe des *Configuration Wizard* eingestellt.

Tabelle 5.1 Portbelegungsplan des MC-Boards

Port	Bezeichnung	Signalart
P0.0	CEX0 (PCA0)	Digital (Push-Pull)
P0.4	UART_TX (UART0), EGBT-046S RX-Pin	Digital (Push-Pull)
P0.5	UART_RX (UART0), EGBT-046S TX-Pin	Digital (Open-Drain)
P1.3	LED 1.3	Digital (Push-Pull)
P1.4	Taster 1.4	Digital (Open-Drain)
P2.0	ADSK (PD_SCK-Pin des LCA)	Digital (Push-Pull)
P2.2	ADDO (DOUT-Pin des LCA)	Digital (Open-Drain)

5.5.1 Erzeugung einer Pulsweitenmodulation

Die Initialisierung des ESC erfolgt nach der Zuführung der Spannungsversorgung. Während der Initialisierung erhält der Drehzahlsteller zuerst die maximale, anschließend die minimale Pulsweite. So wird der Bereich der maximalen bzw. der minimalen Motorendrehzahl durch die Software festgelegt. An dieser Stelle muss erwähnt werden, dass in der Regel diese Initialisierung nur einmal durchgeführt wird.

Damit die vom Hersteller vorgegebene Pulsweite von 50 Hz erreicht werden kann, wird der Systemtakt (*SYSCLK*) der MCU auf 12 MHz eingestellt.

Die Zeitbasis der *PCA0*-Einheit wird auf *SYSCLK*/4 gelegt, damit das PWM-Signal von 50 Hz erreicht wird. Demzufolge ergibt sich für die *PCA0*-Einheit ein Takt von 3,0 MHz, was eine Periodendauer von 333 ns darstellt. Bei jeder Taktflanke wird der Zählstand der *PCA0*-Einheit erhöht. Die Periodendauer der Zähleinheit wird

durch die Zeitbasis von 21,84ms[4] und die Anzahl an Zählschritten von 2^{16} = 65536 gegeben.

Der aktuelle Zählerstand wird den Werten im Vergleichsregister *PCA0CPL0* bzw. *PCA0CPH0* gegenübergestellt. Im Vergleichsregister wird der Vergleichswert des Moduls 0 der *PCA0*-Einheit geschrieben, welches im 16-Bit-PWM-Modus konfiguriert wurde. Der *CEX0*-Pin wird auf High-Level gesetzt, sobald es eine Übereinstimmung zwischen Zähler und Vergleichsregister gibt. Dieser Vergleich wiederholt sich periodisch, sobald die *PCA0*-Einheit von 65535 auf 65536 (0000h auf FFFFh) überläuft.

In Abbildung 5.14 ist die Konfigurations-Ansicht des *PCA0* zu sehen. Außer der Aktivierung des PCA, kann hier auch dessen *pre-scaler* (Zeitbasis) eingestellt werden.

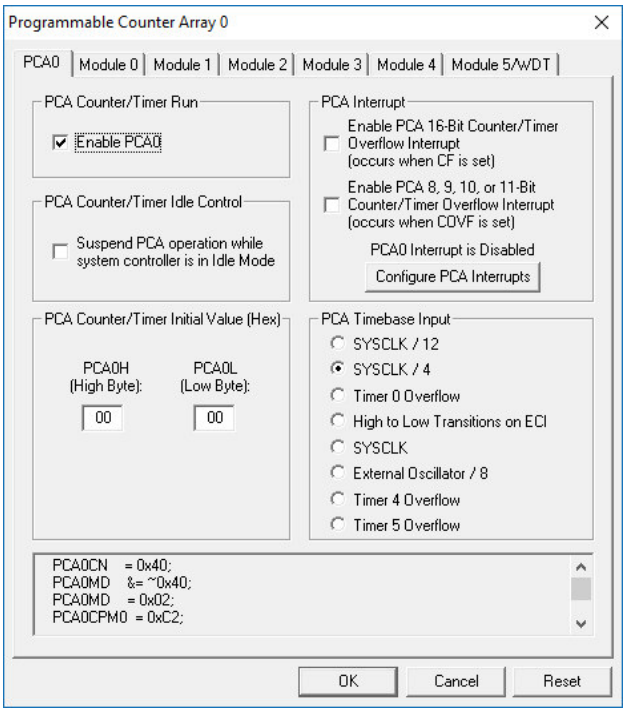


Abbildung 5.14 *Configuration Wizard 2*: *PCA0*

Innerhalb der Periodendauer von 21,84 ms können somit durch die Überschreibung des Vergleichsregisters, beliebige PWM-Signale erzeugt werden.

Für die Berechnung von verschiedenen Vergleichswerten für die Erzeugung von PWM-Signalen wurde folgende Formel verwendet:

[4] 12 MHz ÷ 4 ÷ 65536 = 45,78 Hz ≙ 21,84 ms

$$65535_{T_{PCA0}} \triangleq 21{,}84\ ms$$

$$\frac{65535_{T_{PCA0}}}{21{,}84ms} \triangleq 1{,}0\ ms$$

$$65535_{T_{PCA0}} \cdot \frac{x_{PWM}}{21{,}84ms} \triangleq x_{PWM}$$

Formel 5.5 Berechnung der Zwischenwerte für die PWM

Wobei gilt:

- T_{PCA0}: Periode des *PCA0*
- x_{PWM} in ms

Um einen Vergleichswert zu erhalten, muss der aus Formel 5.5 erhaltene Zwischenwert von der Anzahl an PCA0-Zählschritten (65335) subtrahiert werden. Bei der Berechnung für ein PWM-Signal von 1,0 ms ergibt sich der Wert 3003. Wenn die Differenz beider Werte erzeugt wird, ist das Resultat des Vergleichswertes 62532. Dieser wird in hexadezimaler Schreibweise (F444h) in die Vergleichsregister, mit Hilfe des *Configuration Wizard* eingegeben. Diese Pulsweite wird am Port P0.0 ausgegeben und dem ESC übertragen.

Starten der Schubmessung nach Betätigen des Tasters am Port P1.4

Das Vergleichsregister, in dem zum Startzeitpunkt die Pulsweite für den Motorenstillstand eingetragen ist, wird durch das Drücken des Tasters am Port P1.4 mit einem neuen Vergleichswert überschrieben. Dadurch wird eine neue Pulsweite vom ESC interpretiert und dieser überträgt eine Drehzahl an den Gleichstrommotor, während eine Schubmessung mit der Wägezelle durchgeführt wird.

Die digitalisierten Werte der Wägezelle, werden nacheinander vom Softwaretreiber des Wägezellen-Messverstärkers (LCA), mit einer Abtastrate von 80 Hz abgefragt, vgl. [13], und im Zwischenspeicher der MCU gesichert.

5.5.2 Serielle Kommunikation zwischen Wägezelle und LCA

Verstärkung und Timing mit Hilfe des Messverstärkers HX711

Die Messwerte der Wägezelle werden während der Schubkraftmessung aufgezeichnet. Damit die Messwerte von der MCU abgerufen und zwischengespeichert werden können, muss zunächst die Software für den LCA, welche vom Hersteller bereitgestellt wird, in den Gesamtcode im Keil

Projektverzeichnis integriert werden. Bei jedem Aufrufen des Softwaretreibers des LCA in der *main.c*, kann über die beiden Signalpins *DOUT* und *PD_SCK* der letzte digitalisierte Messwert ausgelesen werden. Der HX711-Messverstärker kann analoge Werte mit 80Hz digitalisieren (sprich 80 Werte pro Sekunde). Bei der Überprüfung der Datenrate, ob auch 80 Werte/Sekunde ausgegeben werden, hat sich herausgestellt, dass die Übermittlung eines Messwertes nicht die erwarteten 0,0125 s benötigt, sondern 0,01114725 ms. Durch diese Dauer ergibt sich eine Datenrate von 90Hz. In der Software wurden entsprechend, Schubmessungen von fünf Sekunden und 450 Messwerten, welche an den LCA übertragen werden, eingestellt.

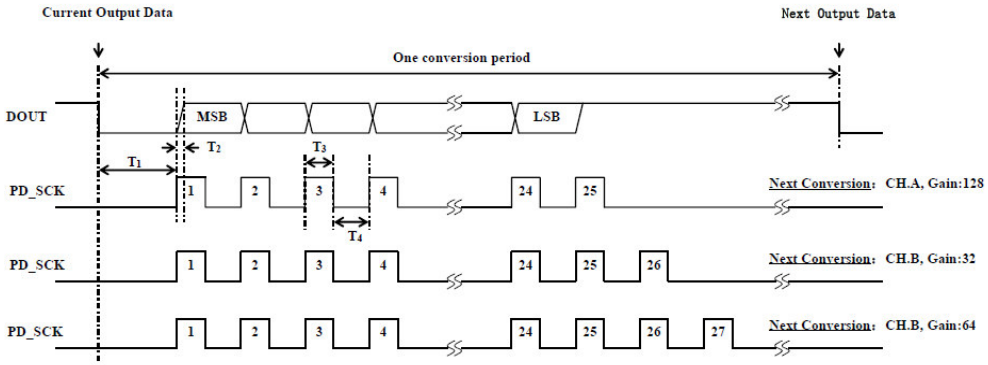


Abbildung 5.15 Daten In- und Output. Auswahl der Verstärkung, Timing und Steuerung

Mit Hilfe des Timing-Diagramms in Abbildung 5.15 wird die Funktionsweise des LCA nach Aufruf in der *main.c*, beschrieben.

Der Messverstärker ist bereit den letzten digitalisierten Wert zu übertragen, sobald der Pin *DOUT* den Low-Pegel erreicht. Der Pegel des Pins *DOUT* wird zu Beginn des Aufrufs auf die Verfügbarkeit des Messwertes überprüft. Jeweils ein Bit des digitalen Messwertes wird in diesem Zustand – dem Low-Pegel – bei jeder High-Flanke am Pin *PD_SCK* über den Pin *DOUT* übertragen.

Jedes Bit wird in einem 24-Bit-Frame einer Messwertvariablen abgespeichert. Dies geschieht durch das sich 24 Mal wiederholende, Setzen und Rücksetzen des Signals am Pin *PD_SCK*.

Der Differenzeingang und der Verstärkungsfaktor des PGA am Pin *PD_SCK* werden mit den Takten 25, 26 und 27 eingestellt. Die Wägezelle wurde an Kanal A angeschlossen, da das Differenzsignal dieser max. 10mV beträgt (bei 2mV/V und 5V LCA-Versorgungsspannung). Somit wird die zulässige Differenzspannung von $\pm$ 20 mV nicht überschritten. Mit dem 25. Takt wird am Kanal A, mit dem PGA ein Verstärkungsfaktor von 128 erzeugt und somit, bei einem Differenzsignal von max. 10 mV, eine maximale Ausgangsspannung von 1,28 V erzeugt. An diesem Kanal ist die Wägezelle angeschlossen.

Anschließend erfolgt die Digitalisierung der PGA-Ausgangsspannung vom A/D-Wandler. Ein Spannungsmessbereich von 5 V ergibt sich bei einer 5 V Versorgungsspannung des A/D-Wandlers. Bei Nennbelastung liegt die Ausgangsspannung des PGA im zulässigen Bereich.

Die Messwertvariable wird als Rückgabewert in der *main.c* durch den LCA Software-Treiber geliefert. Diese Variable wird als Integer im Hauptprogramm interpretiert. Der Messverstärker ist während des High-Pegels des Pins DOUT nicht verfügbar. Das liegt daran, dass der nächste Wert erst nach T_{HX711} =11,1ms ($T_{HX711} = \frac{1}{f_{HX711}} = \frac{1}{90\ Hz}$) abgerufen werden kann.

Tarieren der Wägezelle

Durch ein Aufrufen des Software-Treibers des HX711, ohne die Wägezelle zu belasten, liefert der Sensor einen ganzzahligen Rückgabewert, der 8.834.352 entspricht. Dieser Wert entspricht ungefähr der Hälfte der Auflösung des A/D-Wandlers[5]. Die Variation des Messwertes ist auf die Befestigung des Sensors mit der Konstruktion zurück zu führen. Damit diese leichten Änderungen korrigiert werden, wird bevor Messwerte im externen Speicher der MCU abgespeichert werden, softwaretechnisch eine Mittelung von 10 Messwerten durchgeführt. Hierdurch lassen sich nach Befestigen des Motors mit Luftschraube ebenfalls das Massengleichgewicht herstellen und eventuelle Senkrechtkräfte auf den Sensor kompensieren.

Kalibrieren der Wägezelle

Da während der Schubkraftmessung nur das Gewicht in Gramm interessiert, wird die Kalibrierung für Krafteinheiten vernachlässigt.

Für die Kalibrierung muss ein Kalibrierungsfaktor ermittelt werden. Dieser berechnet sich aus:

- einem bekannten Prüfgewicht welches auf die Wägezelle aufgelegt wird
- dem gemessenen Gewicht des Prüfgewichts
- und dem Tarawert (der im unbelasteten Zustand ermittelt wurde)

[5] 24 Bit = 2^{24} = 16.777.216

Formel 5.6 erläutert den Vorgang nochmals:

$$Prüfgewicht = \left(Messwert_{Prüfgewicht} - Tarawert\right) \cdot k_{gramm}$$

$$k_{Gramm} = \frac{Prüfgewicht}{\left(Messwert_{Prüfgewicht} - Tarawert\right)}$$

Formel 5.6 Berechnung des Prüfgewichtes

Wobei gilt:

- k_{gramm} = Kalibrierungsfaktor

Durch das Auflegen von verschiedenen Gewichten (5g - 2000g) kann gewährleistet werden, dass die Messungen reproduzierbar und zuverlässig sind. Ein Ausschnitt der Messungen mit den verschiedenen Gewichten ist in Tabelle 5.2 aufgeführt.

Tabelle 5.2 *k*-Faktor bei unterschiedlichen Prüfgewichten [1]

Prüfgewicht (g)	Messwert (Prüfgewicht)	Tarawert	*k*-Faktor
5	8.833.016	8.834.352	0,00374
10	8.832.000	8.834.352	0,00425
50	8.823.100	8.834.352	0,00444
1000	8.613.900	8.834.352	0,00454
2000	8.393.100	8.834.352	0,00453

Formel 5.6 wird anschließend softwaretechnisch umgesetzt. Hierbei wird das Prüfgewicht, welches es zu berechnen gilt, in einer Variablen abgespeichert. Mit dem *k*-Faktor 0,00453 kommt man am nächsten auf die zu messenden Prüfgewichte. Für die Berechnung in der Software wurde dieser identisch übernommen.

5.5.3 Datenübertragung an den Computer und Simulink

Für die Visualisierung der Auftriebsmessdaten am PC mit Matlab/Simulink kommt die UART-Schaltung zum Einsatz. Die Konfigurierung der *UART0*-Peripherie des Evaluierungsboards wird mittels *Configuration Wizard 2* durchgeführt. Mit dem Baudratengenerator wird die UART-Peripherie mit einer Baudrate von 9600 bps aktiviert.

Die zu übertragenden Messwerte sind in zwei Bytes (High und Low) aufgeteilt, da die maximale Datenlänge des UART Protokolls nur 8 Bit beträgt, die Datenlänge des Messwertes jedoch 24 Bit ist.

Auf dem Computer wartet Simulink auf die separat übertragenen Bytes und setzt aus diesen den Gewichtswert zusammen (zuerst Low-Byte, dann High-Byte). Bei der Übertragung wird in der Software der UART-*Buffer* beschrieben. Damit sich die Daten nicht überschreiben, wird zwischen jedem Übertragungsvorgang eines Bytes abgewartet, bis es übertragen wurde. Die Übertragung von Low- und High-Byte wird 450-mal (fünf Sekunden) wiederholt, bis die Schubkraftmessung abgeschlossen ist.

In Abbildung 5.16 ist das Konfigurationsmenü des *Configuration Wizard*, für die UART-Schnittstelle des MC-Boards dargestellt. Damit können alle notwendigen Funktionen eingestellt werden.

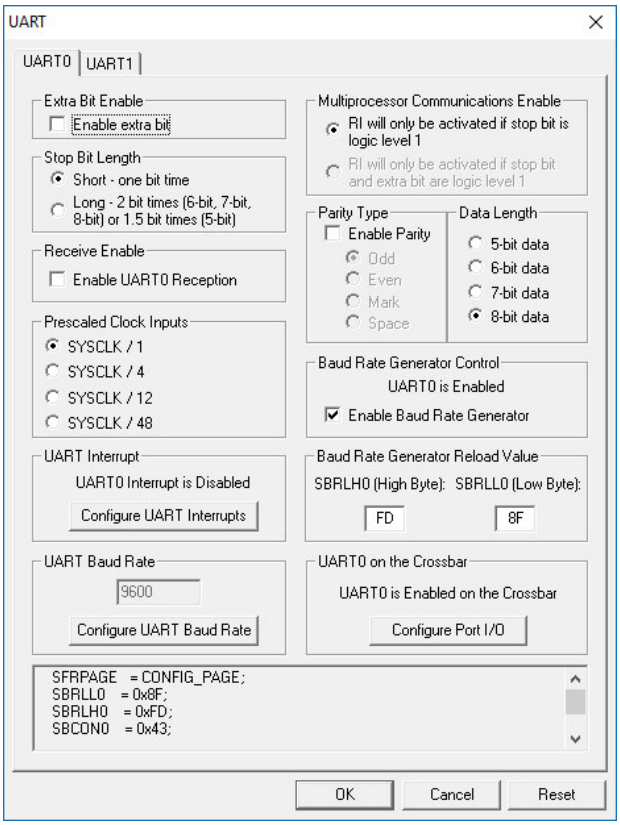


Abbildung 5.16 UART0 Konfiguration

5.6 Simulink Modell

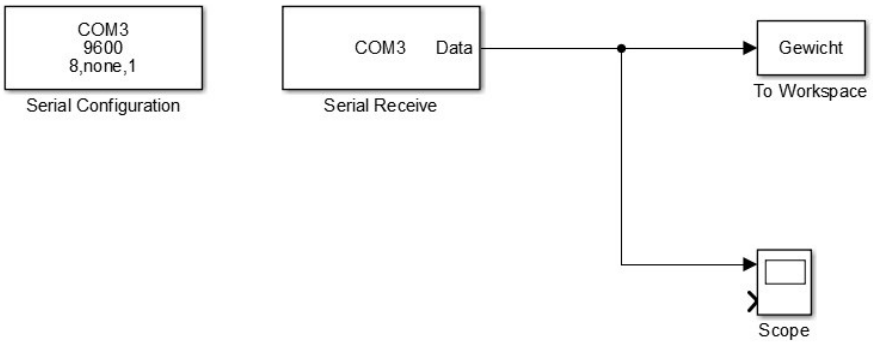


Abbildung 5.17 Simulink Modell für die Übertragung der Schubkraftmessdaten

Abbildung 5.17 Zeigt das Simulink-Modell, welches für die Darstellung und den Empfang der Auftriebsmessdaten verwendet wird. Die Blöcke für die serielle Datenübertragung werden mit folgenden Parametern konfiguriert:

- Baudrate: 9600 Bits/s
- Daten Bits: 8
- Byte-Reihenfolge: *Little Endian*

Die Schubkraftmessung wird initiiert, nachdem das Simulink-Modell gestartet wurde. Die gewonnenen Daten, können nach erfolgreicher Messung im *Scope* ausgelesen bzw. als Diagramm angezeigt werden.

5.7 Maßnahmen für die Erweiterung des Prüfstandes

Der Prüfstand wird dafür verwendet, die Schubkraftmessdaten eines bürstenlosen Gleichstrommotors mit Luftschraube an ein externes Datenverarbeitungs-Programm auf den Computer zu senden. Damit wird die Sprungantwort untersucht. Die Externe Datenverarbeitung wird im neuen Prüfstand von einer Smartphone Applikation übernommen.

Bisher konnte die Drehzahl des Motors von der *JETI-Box* abgelesen werden. Bei der Erweiterung wird diese nicht mehr verwendet. Stattdessen wird mit jeweils einer Reflexlichtschranke die Drehzahl der Motoren gemessen. Diese Daten sollen ebenfalls mit der mobilen Applikation aufgerufen und angezeigt werden können.

Konstruktionstechnisch wird der Aufbau folgender sein:

Ein Rohr aus Karbonfaserverstärktem Kunststoff, zwei BLDC Motoren mit jeweils einer Luftschraube und einer Reflexlichtschranke. In Abbildung 8.2 ist ein Modell

des Aufbaus, welches mit *PTC Creo 3.0* erstellt wurde, dargestellt. Bisher war die Wägezelle zwischen Tragarm und Motor befestigt. Im neuen Aufbau wird der Sensor jedoch zwischen Tragarm und Gestell installiert.

6 AUFGABENSTELLUNG

Ein wichtiger Punkt in der Regelung des Flugverhaltens eines Multikopters ist die Kenntnis über das Sprungverhalten der Schubkraft bei plötzlicher Änderung der Drehzahl. Zum einen aus dem Stillstand heraus, bis zum Erreichen der gewünschten Motordrehzahl beziehungsweise Schub, zum anderen das Verhalten der Sprungantwort bei einem Anstieg der Drehzahl im Schwebeflug.

Die Aufgabe in dieser Arbeit ist die Änderung der Konstruktion eines bereits vorhandenen Prüfstandes, um die Sprungantwort bei Änderung der Motorendrehzahlen zu analysieren.

Auf dem Blockschaltbild in Abbildung 6.1, ist wieder die Aufteilung in zwei Teilsysteme zu erkennen. Mit dem Unterschied, dass für die Veranschaulichung der Daten, unter anderem auch die Drehzahl, eine Applikation für ein Smartphone entwickelt wird. Mit dieser App können die empfangenen Daten in eine externe Datei exportiert und zur Weiterverarbeitung in einer Microsoft Excel Arbeitsmappe verwertet werden.

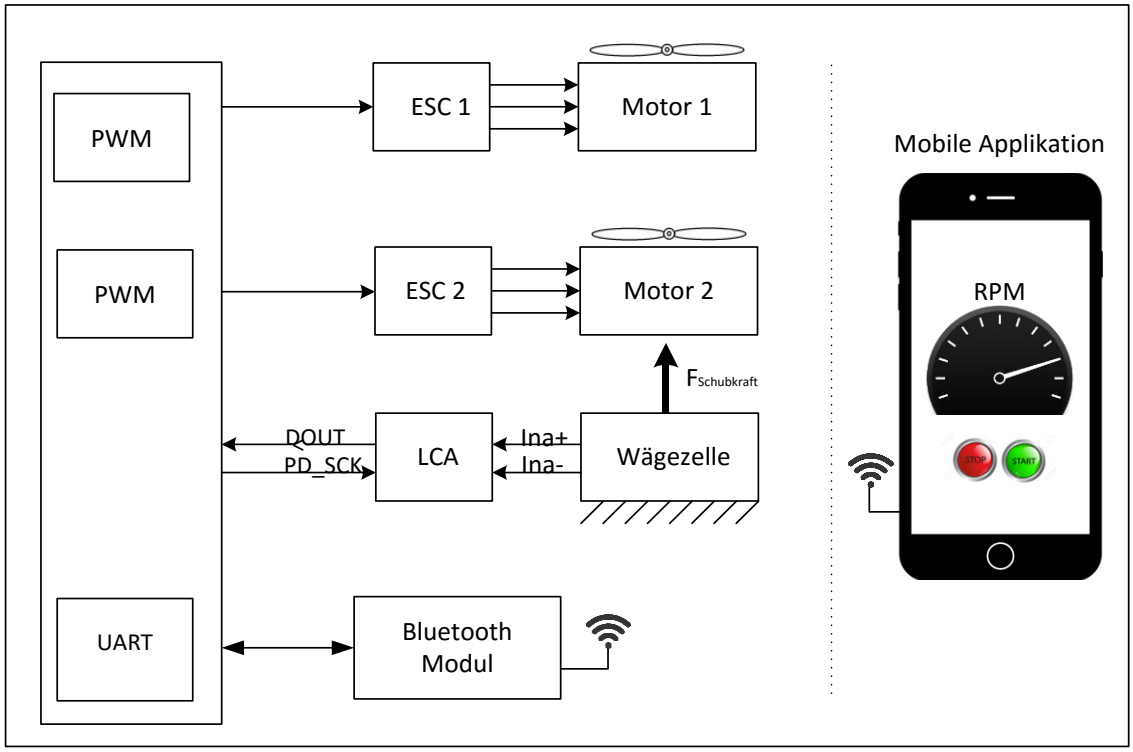


Abbildung 6.1 Blockschaltplan der Aufgabenstellung

Die Wägezelle wird im nächsten Abschnitt genauer betrachtet. Um die Messfähigkeit des Sensors zu überprüfen, sollen wiederholt Messwerte aufgenommen werden.

7 STATISCHE TESTS: MESSFÄHIGKEIT DER WÄGEZELLE

Um die Genauigkeit und die Zuverlässigkeit der Wägezellen-Messwerte zu überprüfen, wird eine Messsystemanalyse nach Methode 1, wie in Abschnitt 0, durchgeführt. Mit verschiedenen Gewichten, soll überprüft werden, ob nach 25 Messungen die Messfähigkeit gegeben ist. Hierfür wird das System von Motor und Luftschraube befreit und es werden statische Tests durchgeführt.

7.1 Messfähigkeit

Für die Bestimmung von geometrischen oder physikalischen Größen werden Messgeräte eingesetzt. Die Qualitätssicherung ist Einsatzgebiet und Ursprung der Bestimmung der Messmittelfähigkeit. In der Beurteilung von Messgeräten werden statistische Methoden verwendet, welche festgelegte Standards für Fertigungsprozesse und Produkte befolgen. Bevor diese Messsysteme verwendet werden können, muss zunächst die Zuverlässigkeit dieser gegeben sein. Bei einer zu großen Varianz der Messwerte besteht die Gefahr, falsche Messdaten zu erhalten, die zu falschen Schlussfolgerungen führen kann.

Bei der Messung sind folgende Aspekte wichtig:

- Genauigkeit:

Ermittelt wird die Differenz zwischen durchschnittlicher Messung und Referenzwert. Dieser Wert ist die systematische Messabweichung (englisch: *bias*)

- Wiederholbarkeit:

Gibt es eine Variation der Ergebnisse bei wiederholtem Messen während die Messbedingungen (gleiche Person, gleiches Messsystem, gleicher Prüfling) identisch bleiben? Die Standardabweichung ist ein Maß für die Wiederholpräzision.

- Reproduzierbarkeit:

Untersuchen, ob es eine Variation der Messergebnisse gibt, sobald ein anderer Bediener dasselbe Messsystem verwendet. Die Beobachtung verschiedener Mittelwerte durch unterschiedliche Bearbeiter, kann die Reproduzierbarkeit zeigen.

- Stabilität:

Alle vorherigen Aspekte müssen über festgelegte Zeitabstände in einem festen Zeitraum gegeben sein. Es werden mehrere Messreihen durchgeführt. Nach jeder Messreihe wird der Mittelwert berechnet und die Differenz der Mittelwerte, welche zu verschiedenen Zeitpunkten beobachtet werden. Hieraus resultiert das Maß der Stabilität.

- Linearität:

Wie für diesen Anwendungsbereich üblich, werden die vorherigen Merkmale mit mehreren unterschiedlichen Prüflingen kontrolliert. Dafür wird ein Mittelwert berechnet und die Differenz zum wahren Messwert ermittelt (in kg), um die Genauigkeit zu untersuchen. Die Linearität ist gegeben, wenn die einzelnen Differenzen zwischen Messwert und Wahrem Wert der verschiedenen Prüflinge klein bleiben.

Messsystem-Analyse (MSA)

Ein MSA kommt zum Einsatz, wenn ein neues Messsystem eingerichtet oder verändert wird. Entsprechend ist eine MSA notwendig, wenn das System an einem neuen Ort aufgestellt wird oder der Einfluss von wichtigen Komponenten sich ändert. Eine MSA wird auch verwendet, sobald bei einer Instandsetzung beziehungsweise einer Überholung, Änderungen vorgenommen werden.

Bevor eine Analyse getätigt werden kann, müssen Vorbedingungen gegeben sein. Eine dieser Bedingungen stellt die Toleranz und die damit verbundene Messmittelauflösung dar. Die Auflösung (RE) eines Messmittels muss RE ≤ 5% der Toleranz der Eigenschaft betragen. Diese Eigenschaft ist hier das Gewicht.

Beispiel:

Im nächsten Abschnitt wird eine MSA mit der Wägezelle durchgeführt. Die Toleranz des Systems liegt bei 200 g (also ± 100 g). Entsprechend bedeuten 10 g, 5 % dieser Toleranz. Hierdurch ist die Auflösung von 10 g des Messsystems gegeben, die für die Gesamtheit der Messwerte gelten muss.

Methode 1:

Dafür wird die Lage und Streuung des Messwertes in einem Toleranzfeld betrachtet. Die Fähigkeitskennwerte *Cg* und *Cgk* legen fest, ob eine Messeinrichtung für das Einsatzgebiet geeignet ist. Diese Werte werden während einer Messreihe, welche 25 Werte beinhaltet, berechnet. Es gibt keinen Bedienereinfluss. Bei der Messung wird ein Normal verwendet, dessen Gewicht bekannt und geeicht ist. Der Ablauf von Methode 1 sieht wie folgt aus:

- Dokumentation erstellen (Teilenummer, Bezeichnung, Merkmal, Toleranz, Prüfmittel, Prüfer, Auflösung, Normal)
- Normal überprüfen
- Mittelwert und Standardabweichung ermitteln
- Berechnung der Fähigkeitskennwerte
- Prüfen, ob Mindestanforderung unter den Fähigkeitskennwerten liegt. Wenn das der Fall ist, ist das Messsystem nicht fähig.

Ein Bearbeiter misst 25 Werte. Das Normal muss nach jeder Messung aus der Vorrichtung entnommen werden und wieder an derselben Position in der Messvorrichtung positioniert sein. Kein Messwert darf verworfen werden.

Mithilfe der Berechnungen der Methode 1 werden der Mittelwert und die Standardabweichung der Messwerte ermittelt. Zusätzlich wird der Abweichungsbetrag des Mittelwertes vom realen Wert des Normals berechnet.

Nach der Auswertung erhält man den C_{gk}- und C_g-Wert gemäß Formel 7.1 undFormel 7.2:

$$C_{gk} = \frac{0{,}1 \cdot T - Bi}{3 \cdot s_g}$$

$$Bi = x_m - x_g$$

Formel 7.1 Berechnung C_{gk}-Wert und Bi

$$C_g = \frac{0{,}2 \cdot T}{6 \cdot s_g}$$

Formel 7.2 Berechnung C_g-Wert

Wobei gilt:

- C_g: Potentieller Fähigkeitsindex
- C_{gk}: Kritischer Fähigkeitsindex
- T: Toleranz
- Bi: Systematische Messabweichung
- x_m: Realer Wert des Normals
- x_g: Mittelwert der Messungen
- s_g: Standardabweichung

Die Herkunft der Fähigkeitswerte ist auf die Automobilindustrie zurück zu führen. Hier liegen die Mindestanforderungen für C_g und C_{gk} bei ≥ 1,33. Diese Anforderungen können jedoch frei gewählt werden. Da die Kenntnisse über die genaue Festlegung der Indizes fehlen, wird 1,33 in dieser Arbeit als Richtwert verwendet.

Bei einem Wert von C_{gk} < 1,33 müssen Maßnahmen für die Reduzierung der Messwertstreuung sowie der Abweichung unternommen werden. Bei einem C_g-Wert < 1,33 kann es sein, dass das Normal nicht richtig vermessen wurde. Dementsprechend muss dies überprüft werden. Sind C_g und C_{gk} kleiner als 1,33, können selbst durch eine Neueinstellung der Messvorrichtung keine Verbesserung der Indizes erreicht werden. Somit ist das Messsystem nicht fähig für die gewünschte Anwendung.

Bei einer MSA kommen, zur Überprüfung von der Fähigkeit eines Messsystems, noch weitere Methoden zum Einsatz. Diese werden vor allem dann angewendet, wenn das Messsystem vollautomatisch ist und es zwischen den Messschritten keinen Einfluss eines Bedieners gibt. An dieser Stelle wird daher auf eine weitere Beschreibung der MSA-Methoden verzichtet.

7.2 MSA der Wägezelle

In diesem Abschnitt wird beschrieben wie die Messfähigkeit der Wägezelle mittels Methode 1 nachgewiesen wird. Dazu wird der Sensor zwischen zwei Aluminiumplatten fixiert, auf die das Gewicht gelegt wird, wodurch eine Druckkraft auf den Sensor ausgeübt wird. Bei den Auftriebsmessungen mit Motor und Luftschraube wird jedoch eine Zugkraft den Sensor beanspruchen. Deshalb wird die Messfähigkeit des Sensors in einem zweiten Schritt mit einem Gewicht untersucht, welches über eine Umlenkrolle und ein Drahtseil eine Zugkraft auf den Sensor ausübt.

7.2.1 Auflegen der Gewichte

Für die Untersuchung der Messfähigkeit, wird die Wägezelle zunächst mit einer Druckkraft belastet. Die Gewichte werden hierzu auf den Sensor gestellt (siehe Abbildung 7.1). Die Platte wurde zuvor mittels Tara[6] aus der Gewichtsmessung ausgeschlossen.

[6] Reset der MCU, siehe Abschnitt 5.5.2

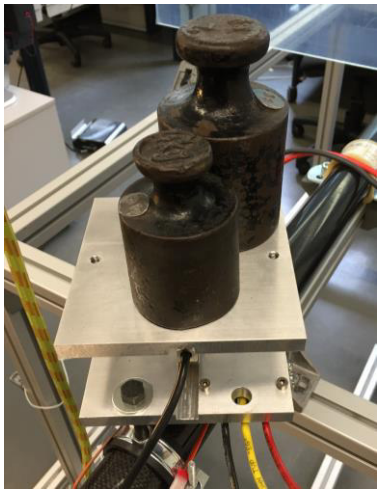


Abbildung 7.1 Gewichte werden auf Motorbefestigung aufgelegt

Um die Messfähigkeit zu überprüfen wird mit Gewichten à ein, zwei und drei kg gemessen. Jede Messwertaufnahme wird 25-mal wiederholt.

Die Durchführung der MSA hat folgende Tabelle 7.1, ergeben:

Tabelle 7.1 Ergebnisse der MSA nach Methode 1 ohne Umlenkrolle und Drahtseil [7]

Gewicht des Normals	Toleranz	Auflösung	Mittelwert	Standartabweichung	C_g	C_{gk}
1000	200	10	999,32	0,6354	2,96	2,86
2000	200	10	1995,96	0,6758	9,87	7,87
3000	200	10	3002,6	0,6455	10,33	8,99

Hierbei ist anzumerken, dass bei der Messung alle Gewichtswerte mit negativem Vorzeichen beaufschlagt werden, da durch Auflegen der Gewichte eine Druckkraft auf die Wägezelle ausgeübt wird. Das ist auf die softwareseitige Implementierung der Aufnahme der Wägezellen-Messwerte zurück zu führen.

Folgende Abbildung 7.2 zeigt die vorhandene Linearität. Das gemessene Gewicht stimmt mit dem geeichten Gewicht, über die gesamte Spanne der Messungen überein, ohne die Toleranz zu verletzen.

[7] Alle Gewichtsangaben in g

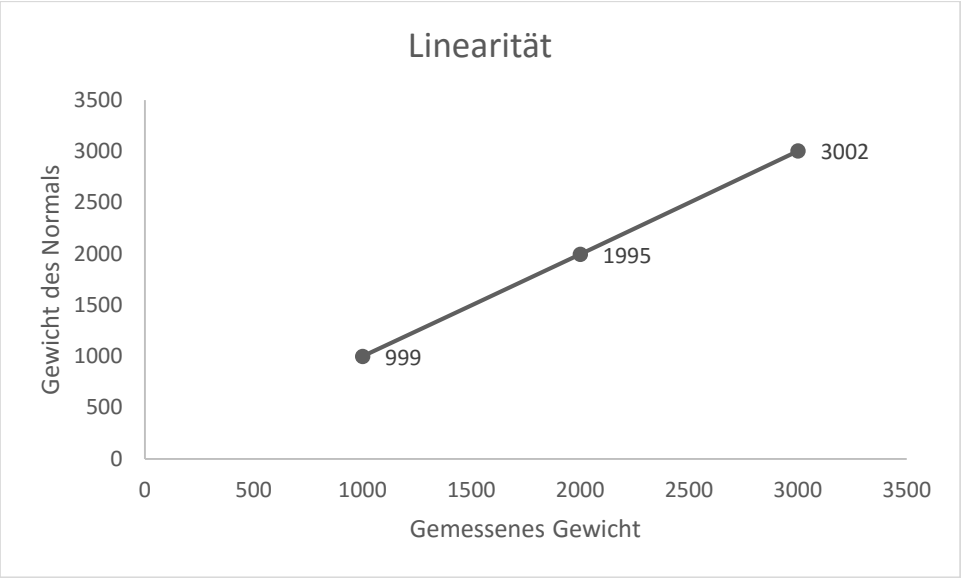


Abbildung 7.2 Linearität des Messmittels

Die Referenzwerte C_g und C_{gk}- Werte von 1,33 wurden nicht unterschritten. Demzufolge hat die Messsystemanalyse ergeben, dass der Sensor fähig ist, Messungen zuverlässig und innerhalb der Toleranzen durchzuführen. Die Linearität ist gegeben, da die Standardabweichung bei allen gemessenen Gewichten etwa 0,6 beträgt. Die vollständige Analyse mit allen Messwerten ist dem Anhang zu entnehmen.

Durch die Verwendung von einem Motor mit Luftschraube wird eine Schubkraft erzeugt, also eine Zugkraft in der y-Achse des Sensors (siehe Abbildung 5.7). Im nächsten Abschnitt wird diese Kraft mit Hilfe eines Gewichtes, welches über eine Umlenkrolle und ein Drahtseil befestigt ist, simuliert.

7.2.2 Aufhängung der Gewichte über eine Umlenkrolle

Bei der Durchführung der Messfähigkeit der Wägezelle wurde für diesen Abschnitt das Gewicht mit Hilfe einer Umlenkrolle und eines Drahtseils an der Wägezelle befestigt. Die Aufhängung des Gewichts, welche in Abbildung 7.3 und Abbildung 7.5 zu sehen ist, spielt dabei eine wichtige Rolle. Die Versuche werden aufgeteilt in *symmetrische Aufhängung* und a*symmetrische Aufhängung*. Anschließend werden die Unterschiede erklärt.

- Nicht symmetrischer Aufhängung

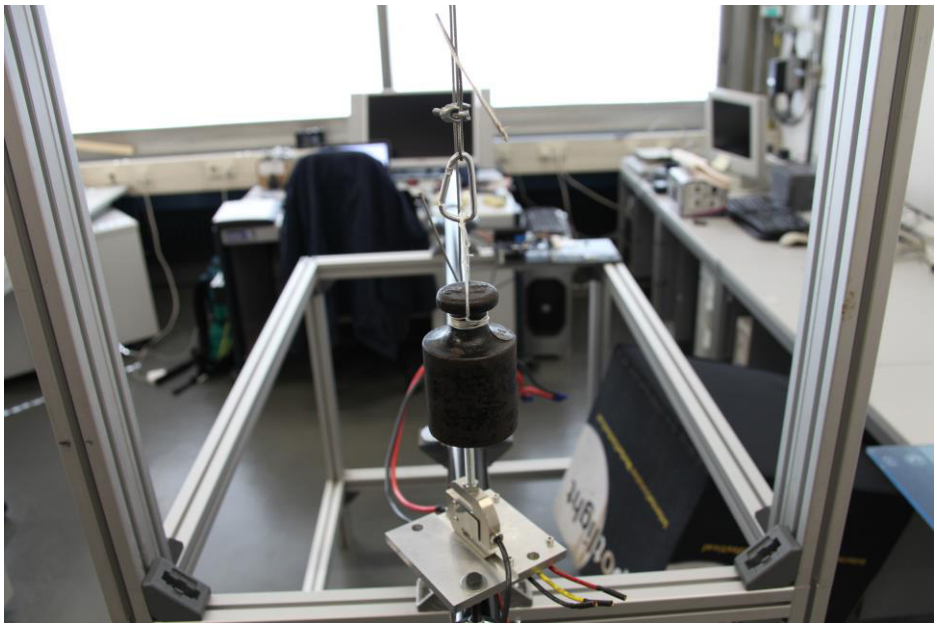


Abbildung 7.3 Aufhängung mit Drahtseil und Umlenkrolle

Bei der Ermittlung einer passenden Befestigung des Gewichtes am Drahtseil wurde zunächst mit einem einfachen Draht das obere Ende des Gewichtes umwickelt. Dadurch wird aber das Gewicht nicht symmetrisch aufgehängt, was bedeutet, dass der Schwerpunkt des Gewichtes verlagert ist. Nach Starten der Messung und Auslösen eines Impulses am Gewicht per Hand, wird im Simulink eine nicht-symmetrische gedämpfte Sinus Form erhalten (siehe Abbildung 7.4). Der Impuls dient dazu, die Schwingungen besser erkennen zu können, und zeigt, dass durch die asymmetrische Aufhängung, das Signal auch im vermeintlich ruhen Zustand schwingt.

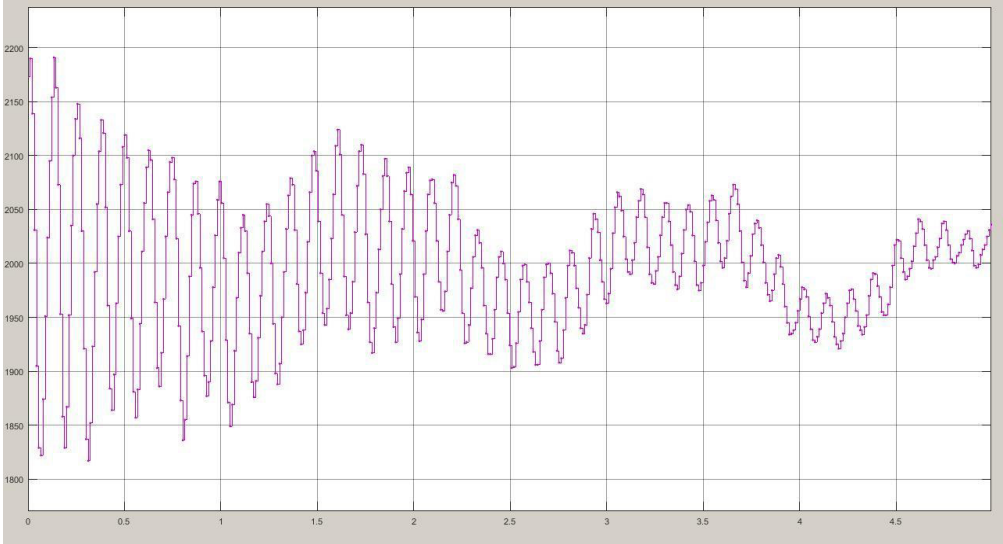


Abbildung 7.4 Asymmetrische, gedämpfte Sinus-Form nach Impuls mit der Hand

- Symmetrischer Aufhängung

Abbildung 7.5 Symmetrische Aufhängung des Gewichts

Um die Symmetrie der Aufhängung zu gewährleisten, wurde anschließend in das Gewicht mittig ein Loch gebohrt und das Drahtseil befestigt.

Nach Starten der Messung und Auslösen eines Impulses am Gewicht per Hand, wird nun im Simulink eine symmetrische gedämpfte Sinus Form erhalten (siehe Abbildung 7.6).

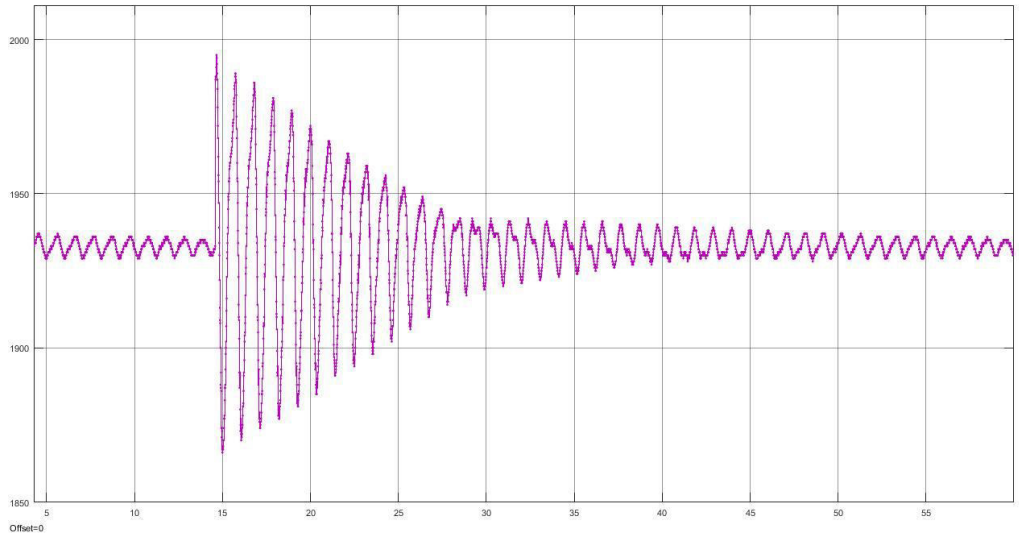


Abbildung 7.6 Symmetrische, gedämpfte Sinus-Form nach Impuls mit der Hand

Bei der Durchführung der Messfähigkeitsanalyse mit genannten Gewichten, wird festgestellt, dass die Werte im Simulink von dem tatsächlichen Gewicht minimal abweichen.

Die Zugkraft wird bei einer festen Rolle nur umgelenkt, also gilt $F_{Zug} = F_{Last}$. Bei dieser Anwendung treten dagegen Reibungen auf, die sich mit Formel 7.3 berechnen lassen:

$$F_{Zug} = F_{Gewicht} + F_{Reibung}$$

Formel 7.3 Reibung berechnen

Bei der Verwendung eines Seils und einer Rolle muss jedoch die spezifischere Seilreibung mit Formel 7.4 berechnet werden:

$$F_{Zug} \leq F_{Halte} \cdot e^{\mu_{H} \cdot \alpha}$$

Formel 7.4 Seilreibung

Wobei gilt:

- α: Umschlingungswinkel (hier 180°)
- μ_H: Haftreibungskoeffizient (Stahl auf Kunststoff = 0,3)

Bei der mehrmaligen Wiederholung von Messreihen treten jedoch keine konstanten Differenzen auf.

Im nächsten Abschnitt wird etwas näher auf die Temperaturmessung eingegangen. Mit dieser soll bewiesen werden, dass unter Verwendung einer Vollbrücken-Schaltung in der Wägezelle, die Temperatur nur einen geringen Einfluss auf die Messwerte hat.

7.3 Temperaturtest über 24h

Um zu beweisen, dass die Temperatur, die Messwerte der Wägezelle nicht beeinflussen, wird eine 24 Stunden-Test durchgeführt. Hierbei wird das C-Programm so verändert, dass 24 Stunden lang Wägezellen-Messwerte von der MCU aufgenommen und mittels Simulink gespeichert werden.

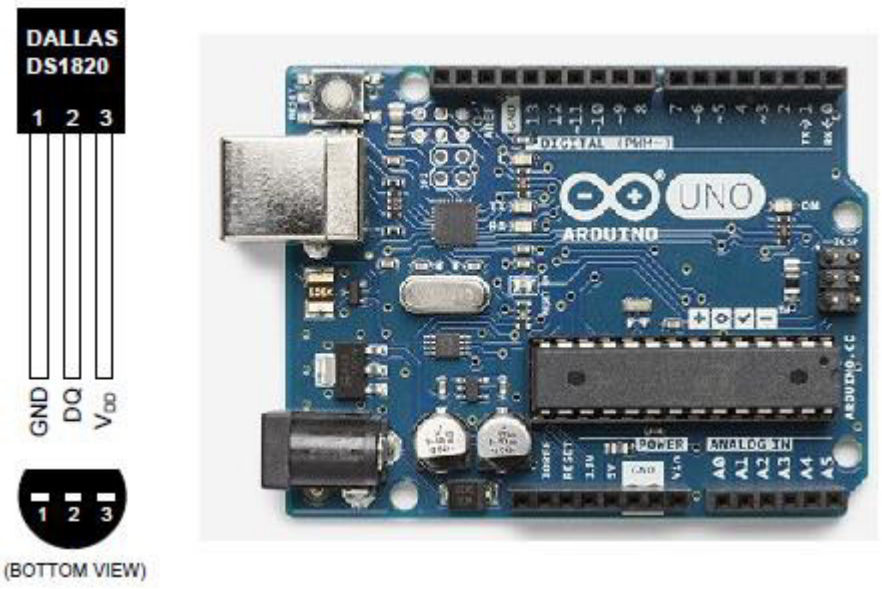


Abbildung 7.7 Temperatursensor DS18s20 von DALLAS [14] und Arduino Uno [15]

Für die Messung der Temperatur wurde der Temperatursensor *DS18s20* von *DALLAS* (links in Abbildung 7.7) verwendet, welcher über einen *One-Wire* Bus mit einem *Arduino Uno* Board (rechts in Abbildung 7.7) kommuniziert. Die Daten werden über eine serielle Schnittstelle an Simulink weitergeleitet. Ein Beispiel-Code und die verwendete *One-Wire* Bibliothek für die Temperaturmessung liegt dem Datenblatt des Sensors bei und wurden als solche verwendet. Die Software für die Datenverarbeitung der Gewichtsdaten mit Simulink und Excel sind ebenfalls dem Anhang zu entnehmen.

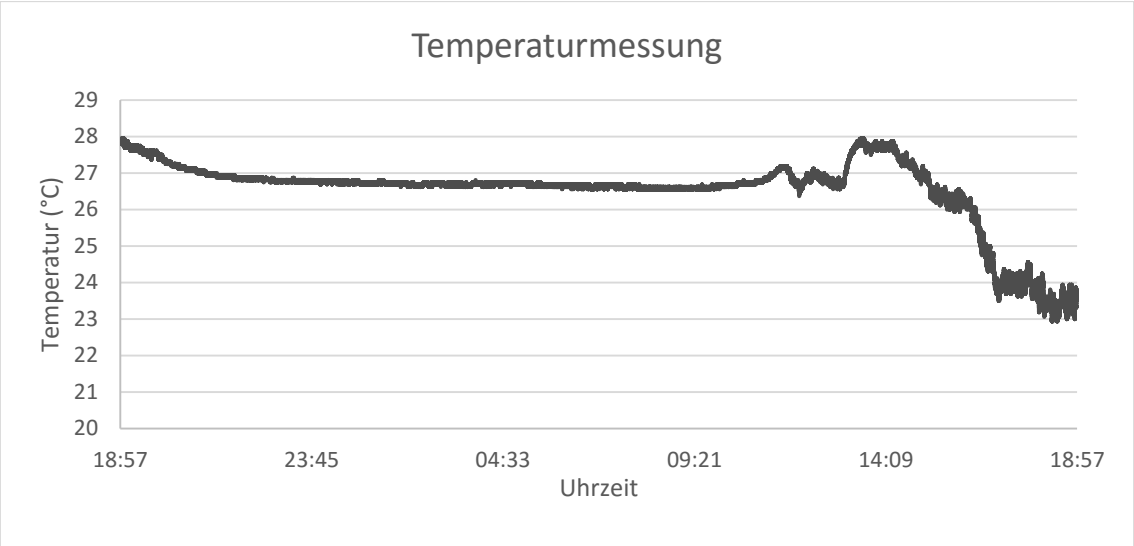


Abbildung 7.8 Ergebnis einer 24h Temperaturmessung mit dem DS18s20

Die Aufzeichnung der Temperatur über einen Zeitraum von 24 Stunden ist in Abbildung 7.8 dargestellt. Die Temperatur nimmt während der Nacht leicht ab. Die Abnahme der Temperatur am Anfang des Tages (etwa um 10:00) sind auf das Öffnen der Fenster und Türen im Raum zurück zu führen.

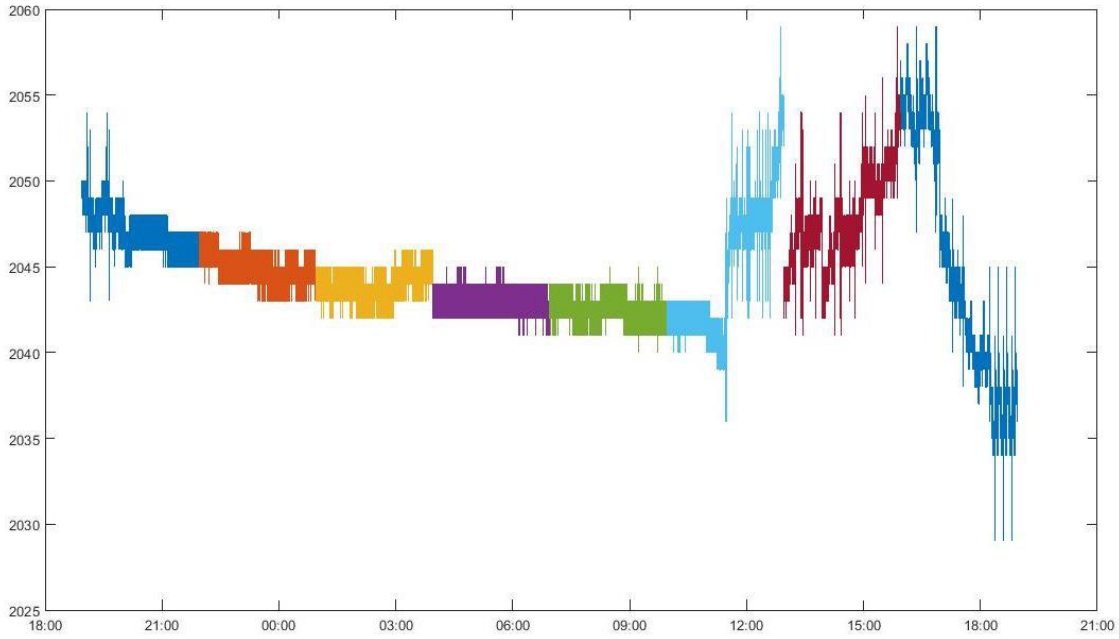


Abbildung 7.9 Ergebnis einer 24h (x-Achse) Gewichtsmessung (y-Achse) mit der Wägezelle.

Die Aufnahme der Gewichtsmessdaten über einen Zeitraum von 24 Stunden in Abbildung 7.9 hat einen ähnlichen Verlauf. Die Berechnung der Korrelation wird

zeigen, dass der Zusammenhang zwischen Temperatur und Gewicht dennoch nicht ausgeprägt ist.

Die Korrelation ist „ein dimensionsloses Maß für den Grad des Linearen Zusammenhangs zwischen mindestens zwei intervallskalierten Merkmalen“, [16]. Der zu untersuchende Zusammenhang besteht zwischen der Temperatur und dem Gewicht. Der Korrelationskoeffizient kann Werte zwischen -1 und 1 annehmen. 1 entspricht einem vollständigen, positiv-linearem Zusammenhang. Ist der Wert zwischen 0 und 0,5 besteht ein schwacher Zusammenhang der Werte.

Der Korrelationskoeffizient berechnet sich zu:

$$r = \frac{Cov(x,y)}{S_x \cdot S_y}$$

Formel 7.5 Korrelationskoeffizient

Wobei:

- Cov(x,y): Kovarianz der beiden Messwertreihen
- S_x: Standartabweichung von x
- S_y: Standartabweichung von y

Wie vermutet hat ein 24 Stunden Test bewiesen, dass die Gewichtsschwankungen und die Temperatur schwach korrelieren (r = 0,28). Der Einfluss der Temperatur auf die Wägezellen-Messwerte wird durch Verwendung einer Vollbrücke in der Wägezelle verhindert (näheres hierzu in Abschnitt 5.3.2).

Die Durchführung von Schubkraftmessungen mit dem Prüfstand haben jedoch gezeigt, dass die Schwankungen[8], welche bei einem 24 Stunden-Test auftreten, nicht relevant sind. Der Prüfstand wird durch die von den Luftschrauben verursachten Vibrationen dennoch beeinflusst. Somit sind die externen Einflüsse im Raum bei der Schubkraftmessung zu vernachlässigen. Diese Tests haben trotzdem bewiesen, dass der verwendete Sensor zuverlässige und genaue Daten liefert.

[8] Mit einer Amplitude der Schwankungen von etwa 30 g

8 AUSBAU DES PRÜFSTANDES

Im Rahmen einer Bachelorarbeit[9] wurden, zur Auslegung von Regelparametern für eine Flugsteuerungseinheit, das Sprungverhalten eines BLDC Motors mit Luftschraube untersucht. Dabei wird eine Schubkraftmessung bei sprunghafter Änderung der Motordrehzahl durchgeführt. Der in Abbildung 8.1 dargestellte, Zusammenhang zwischen Drehzahl und Schubkraft, spiegelt sich in einer parabolischen Kurve wieder. Diesen Zusammenhang hat die Realisierung von Schubkraftmessungen im Rahmen oben genannter Bachelorarbeit unterstrichen.

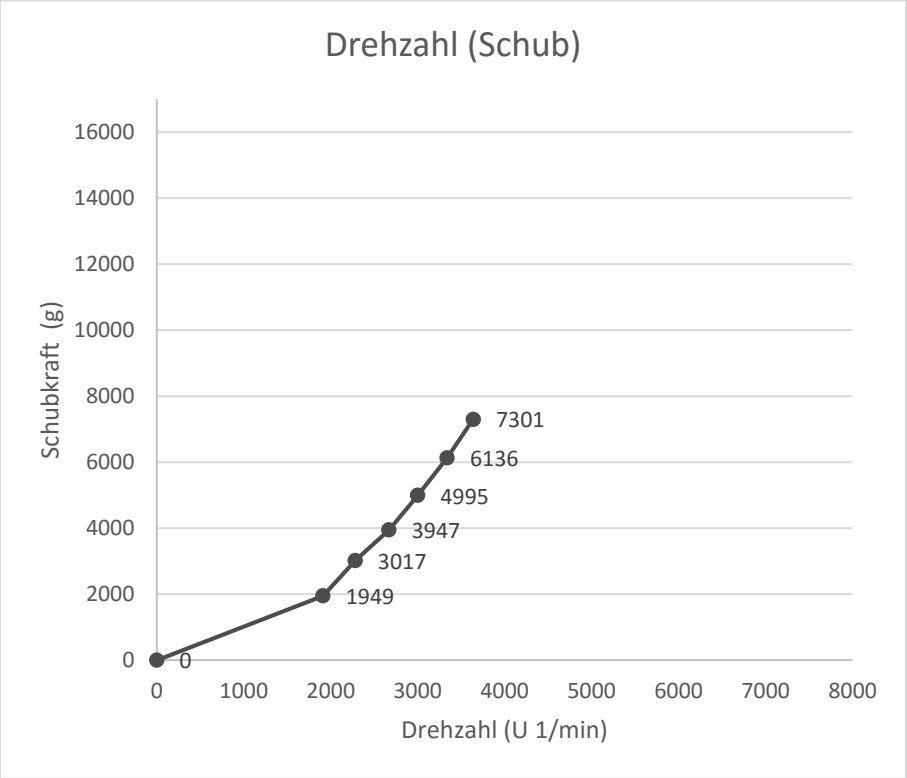


Abbildung 8.1 Auftriebsmessung für die Luftschraube XOAR 1412-25X12

Für den Prüfstand mit zwei Multikopterantrieben sind Änderungen, sowohl konstruktions-, hardware- als auch softwaretechnisch, notwendig. So soll beispielsweise mit einer Reflexlichtschranke die Drehzahl gemessen werden. Es wird ein weiterer Motor mit Drehzahlsteller angebaut, und die Konstruktion soll an die des manntragenden Multikopters angepasst werden. Zusätzlich wird die Software optimiert und hinsichtlich der Verwendung von Interrupts verändert. Der neue Testablauf sieht eine Messung von zehn Sekunden vor. Gleichzeitig werden

[9] Nermin Dedic, *Entwicklung und Aufbau eines Prüfstandes zur Auftriebsmessung von Luftschrauben*

die Motoren für fünf Sekunden gestartet und nach Ablaufen der Zeit wieder abgeschaltet. Hiermit kann das Schubverhalten bei langsamem Auslaufen der Motoren erfasst und untersucht werden.

8.1 Konstruktion

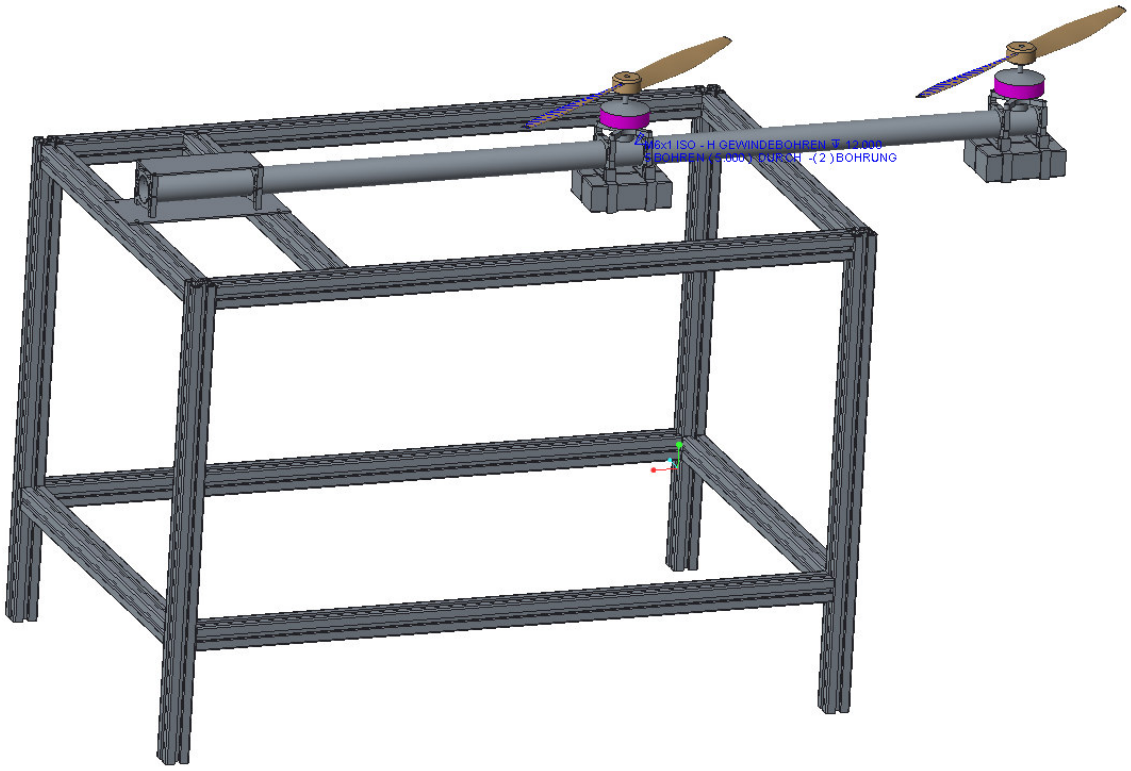


Abbildung 8.2 Aufbau des neuen Prüfstandes, erstellt mit *PTC Creo Parametric*

Um für die Regelparamater der Flugsteuerung ein genaues Sprungverhalten messen zu können, müssen für den Prüfstand realitätsnahe Bedingungen herrschen. Entsprechend muss die Kohlefaser-Leichtbau-Konstruktion vgl. [17] [10] und Abbildung 8.3 nachgebaut werden. Der manntragende Multikopter wird mit zwölf Tragarmen mit je zwei Luftschrauben ausgestattet. Für den Prüfstand wird jedoch nur ein Tragarm benötigt.

[10] Die Konstruktion des manntragenden Multikopters wurde in der Masterarbeit von M.Sc. Tobias Kuentzle entwickelt.

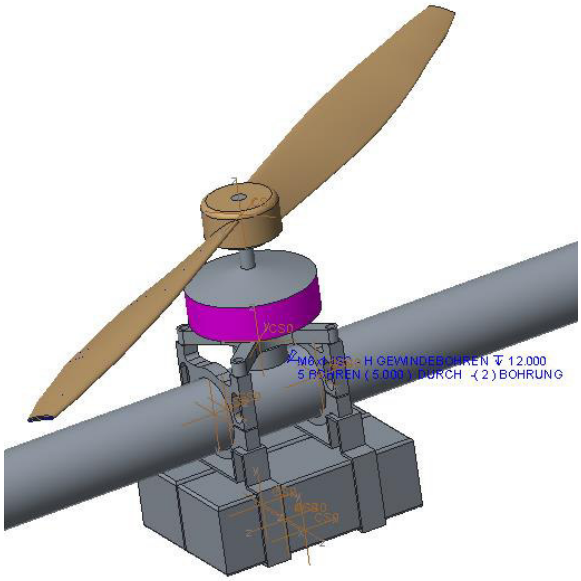


Abbildung 8.3 Befestigung des Motors mittels Rohrschellen am Tragarm

Wie auch beim VC5, wird die Sandwichbauweise für die Rahmenplatten verwendet. Diese dienen auch der Aufnahme des Pilotensitzes und müssen besonderen Belastungen standhalten. Darüber hinaus ist bei solch einer Bauart eine hohe Steifigkeit, bei dennoch geringem Gewicht gegeben. Den Prüfstand Betreffend, welcher konstruktionstechnisch aus *Minitec*[11]-Profilen und einem einzigen CFK Ausleger besteht, ist die Sandwichbauweise mit CFK-Platten vernachlässigt und stattdessen mit Aluminiumplatten gegeben (siehe Abbildung 8.2).

Der Ausleger nimmt die beiden Motoren auf. Diese sind mittels Rohrschellen (zwei Stück je Motor) befestigt und aus Aluminium Vollmaterial gefräst. Um auch hier Gewicht zu sparen, werden die Rohrschellen an nicht belasteten Stellen ausgespart.

8.2 Hardware

8.2.1 Inbetriebnahme des zweiten Motors

Der manntragende Multikopter besteht aus einem System mit 12 Rotoren, welche auf zwei Kreisbahnen punktsymmetrisch angeordnet sind. Bei der Konfiguration der Motoren muss dementsprechend darauf geachtet werden, dass der Schub eines jeden Propellers ein Drehmoment um die x- und y-Achse proportional zur Länge des Tragarms erzeugt. Werden diese Momente durch identischen Schub der

[11] Aluminiumprofile-Hersteller

Rotoren aufgehoben, schwebt der Multikopter. Ein Augenmerk ist auch auf das Moment um die z-Achse zu legen: damit dieses Moment sich ausgleicht, muss jeder benachbarte Rotor in entgegengesetzter Richtung drehen. Die Drehzahl der Motoren wird dann verändert, um den Multikopter zu gieren.

Für den Prüfstand mit einem einzigen Ausleger müssen dieselben Bedingungen, in Bezug auf die umgekehrte Drehrichtung, herrschen. Mit der Verwendung des zweiten Motors muss auch ein zweiter ESC verbaut werden.

Die JETI-box wird in der Erweiterung des Prüfstandes ausschließlich für die Konfiguration der Motoren verwendet. Die Drehrichtung kann zum Beispiel beim aktuellen Prüfstand nur mit der JETI-Box eingestellt werden.

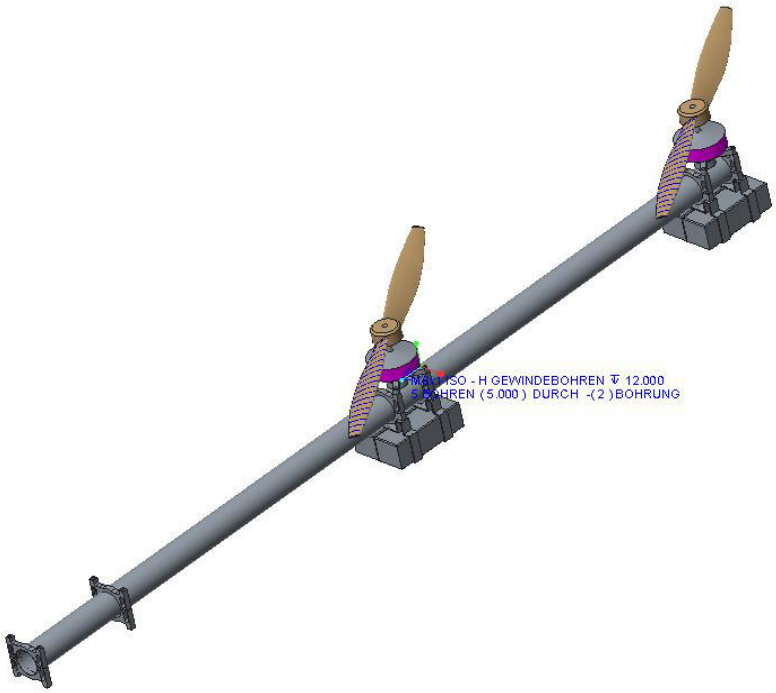


Abbildung 8.4 CFK-Ausleger mit zwei Motoren

Der ESC generiert für den zweiten Motor ein PWM-Signal. Die Peripherie der MCU ermöglicht es mit dem *PCA0*, welcher sechs Module hat, sechs verschiedene PWM-Signale zu erzeugen. Modul eins erzeugt am Port P0.1 (*CEX1*) ein 16-Bit PWM-Signal für Motor eins. Für den zweiten Motor wird Modul zwei im selben Modus verwendet um am Port P0.2 (*CEX2*) ein PWM-Signal zu erzeugen.

8.2.2 Wägezelle

Beide Motoren erzeugen eine maximale Schubkraft von je 17 kg. Bei einem resultierenden Schub von 34 kg ist die bisher verwendete Wägezelle mit einer Nennlast von 20 kg nicht mehr geeignet. Sie wird durch einen identischen S-förmigen Sensor[12] mit einer Nennlast von 50 kg ersetzt.

Um die Schubkraft messen zu können, welche beide Motoren am Ausleger ausüben, wird auch die Position der Wägezelle geändert und unterhalb des Tragarms befestigt (siehe Abbildung 8.5). Dabei wird darauf geachtet, dass die Befestigung der Wägezelle am Ausleger keine Reaktionskraft in entgegengesetzter Richtung der Schubkraft ausübt, um die Messergebnisse nicht zu verfälschen.

Abbildung 8.5 Befestigung der Wägezelle

8.2.3 Drehzahlmessung

Sensor

Für die Drehzahlmessung werden zwei Reflexlichtschranken (eine pro Motor) des Typs MRL601 verwendet.

[12] Ebenfalls der Firma Bosche GmbH & Co. KG

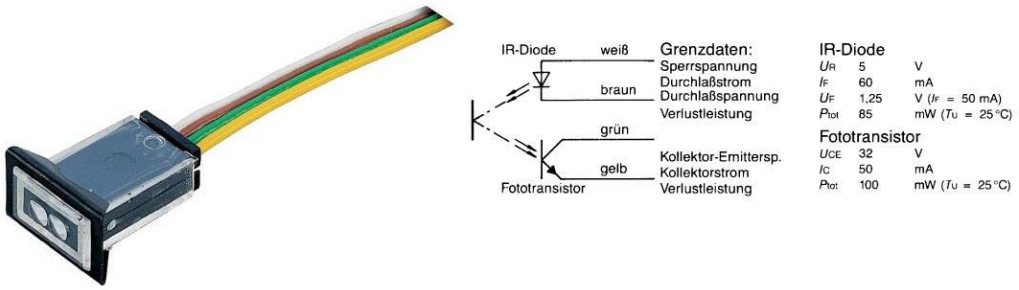


Abbildung 8.6 Reflexlichtschranke MRL601 und Datenblatt

Sie besteht aus einer IR-Sender-Diode und einem Fototransistor, welcher als Empfänger fungiert. Bei Reflexion wird der Strom über den Fototransistor geleitet. Wenn der Sensor nicht reflektiert, wird kein Strom geleitet und es wird kein Signal ausgegeben. Um den Sensor zu testen wurde zuvor ein Schaltkreis mit zwei Widerständen erzeugt. Das erzeugte Signal ist auf Abbildung 8.7 zu erkennen. Demnach ist aber kein Rechteck-Signal zu verzeichnen, was aber für die Verarbeitung mit dem MC-Board benötigt wird.

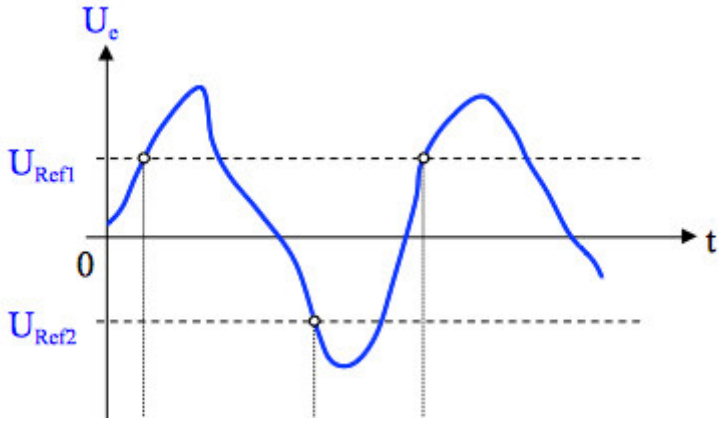


Abbildung 8.7 Lichtschranken-Signal ohne Schmitttrigger

Die Lösung hierfür ist ein Schmitt-Trigger. Der Vorteil dieses Bauteils ist die Ausgabe eines wohl definierten Ausgangspegels, welcher abhängig vom Eingangspegel ist. So wird ein High-Pegel erreicht, wenn am Eingang eine Spannung U_H überschritten wird. Ein Low-Pegel wird ausgegeben, wenn die Spannung U_L am Eingang unterschritten wird. Durch diese Hysterese bleibt der Pegel in den Übergangsphasen bestehen.

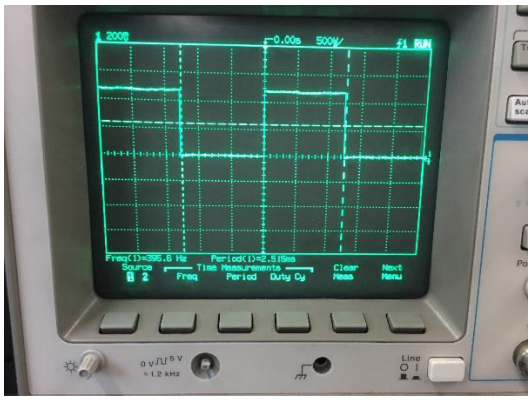


Abbildung 8.8 Signal der Drehzahlmessung am Oszilloskop

Das Oszillogramm gibt ein deutliches Rechteck-Signal zu erkennen. Bei der Verwendung mit einer MCU und entsprechender Peripherie, wird ein High- oder Low-Pegel benötigt (5 V/0 V), damit die Programmierung mit einer Flankenerfassung erfolgen kann.

Die Schaltung der Drehzahlmessung stellt sich wie in Abbildung 8.9 dar:

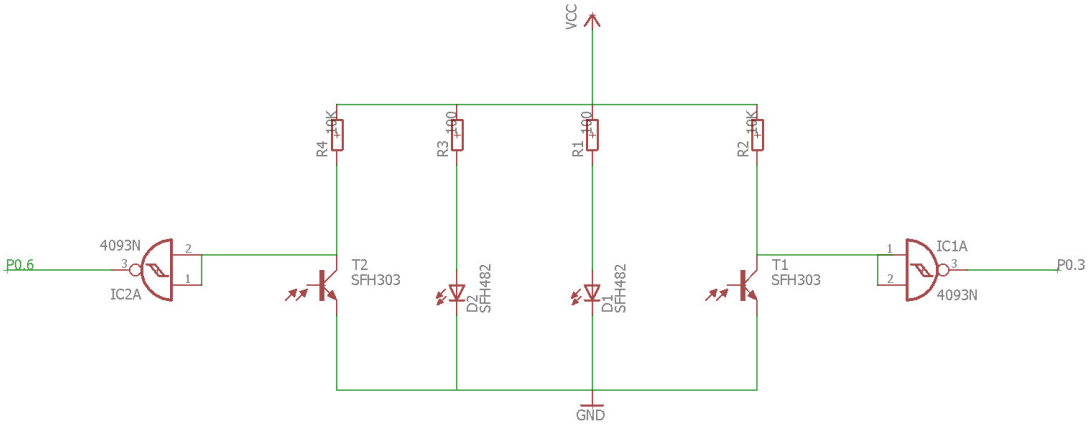


Abbildung 8.9 Schaltplan der Drehzahlmessung mit Ausgang zu den Ports P0.6 und P0.3 der MCU

Messung mittels Bit-Muster

Die Messung der Drehzahl mit einer Reflexlichtschranke funktioniert über Reflexion und Nicht-Reflexion auf einer ausgewählten Oberfläche. Der Motor wird mit Hilfe eines schwarz-weiß gestreiften Bandes (4 weiße und 4 schwarze Streifen) markiert. Bei jeder Reflexion wird auf dem Oszilloskop eine Flanke sichtbar. Diese Flanke wird mit dem Evaluierungsboard verarbeitet und in der Software kann die Anzahl der Umdrehungen pro Minute berechnet werden.

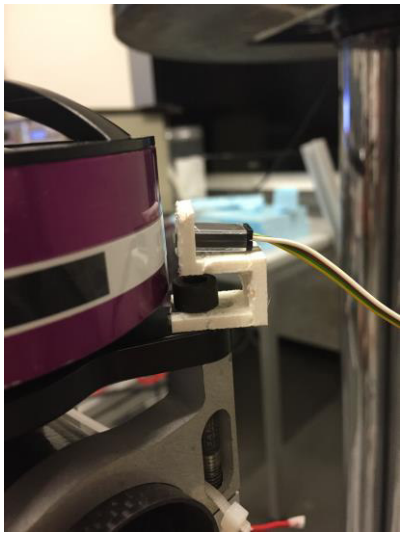


Abbildung 8.10 Halter mit Lichtschranke und Bit-Muster am Motor

Vier weiße Streifen stellen vier Flanken und demnach eine Umdrehung dar. Um die Drehzahl zu berechnen wird ebenfalls eine Zeitkomponente benötigt. Das heißt, wenn die Zeit zwischen zwei Flanken und die Anzahl dieser pro Umdrehung bekannt ist, kann die Drehzahl berechnet werden. Diese stellt den Kehrwert der Umlaufdauer $n = \frac{1}{T}$ dar und somit gilt:

$$RPM = \frac{60\ s}{4 * T}$$

Formel 8.1 Berechnung der Drehzahl

mit:

- T: Zeit zwischen zwei Flanken
- RPM in Umdrehung 1/min

PCA1: Edge-Triggered Capture-Modus

Der *Programmable Counter Array* kann in sechs verschiedenen Modi betrieben werden. Zwei dieser Modi regeln bereits die Pulsweitenmodulation. Für die Berechnung der Drehzahl über die Evaluation der Zeit zwischen zwei Flanken wird der PCA1 im Capture Modus konfiguriert.

Im Capture Modus kann der PCA eine vom Sensor erzeugte Taktflanke erfassen und in ein Capture-/Compare-Register schreiben (*PCA1CP6*: Motor 1 und *PCA1CP7*: Motor 2). Durch Einstellung des *Control*-Registers kann definiert werden, welche Zeitbasis der *Capture*-Modus anwenden soll und bei welchem Typ Flanke dieser aktiv wird. In der vorliegenden Arbeit wird als Zeitbasis die des Timer 4 verwendet und bei einer *Low-to-High* Transition wird die Flanke erfasst.

Sobald eine Flanke erfasst ist, wird das Capture-/Compare-Flag auf logisch 1 gesetzt und ein Interrupt wird ausgelöst. In der entsprechenden Interrupt Service Routine kann anschließend die Drehzahl, anhand der im Timer 4 definierten Zeitbasis und der Formel 8.1 berechnet werden.

Das Oszillogramm in Abbildung 8.8 zeigt eine Frequenz von 395,6 Hz. Mit Formel 8.1 errechnet sich damit eine Drehzahl von 5934 Umdrehungen pro Minute.

8.2.4 Notausschalter

Um das System zu jedem Zeitpunkt stoppen zu können und in einen sicheren Zustand zu bringen, wird ein Notausschalter, wie sie auch in der Industrie eingesetzt werden, verwendet.

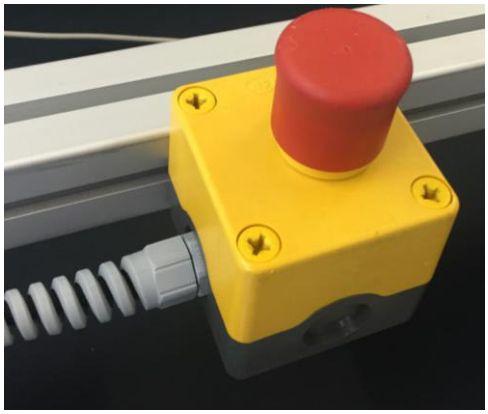


Abbildung 8.11 Notausschalter

Abhängig vom Gebiet werden verschiedene Strategien eingesetzt, um ein System in einen sicheren Zustand zu versetzen. Eine Lösung wäre die Unterbrechung der Stromzufuhr des ESC, welcher unverzichtbar für die Motoransteuerung ist. Bei der Auswahl eines Schalters muss jedoch beachtet werden, wieviel Strom durch die Leitungen am ESC fließt. Der Dauerstrom des ESC beträgt 125 A, vgl. [8]. Bei einer derart großen Stromstärke kann es, bei abrupter Unterbrechung des Stromkreises durch den Notausschalter, zu einem Lichtbogen kommen. Da die genauen Eigenschaften des Schalters nicht bekannt sind, wird ein Notaus über die Software und über die Hardware des Mikrocontrollerboards gelöst.

Vorgehensweise

Die ESCs benötigen ein PWM Signal, um die Motoren mit einem kommutierten Strom von 8 bis 32 kHz ansteuern zu können. Somit kann ein Motorstopp auch über die Motoransteuerung der ESCs herbeigeführt werden.

Software- und Hardwaretechnisch wird das Auslösen eines Notaus mit einem externen Interrupt gelöst. Mit dem *Configuration Wizard 2* wird das externe Interrupt (*INT0*) konfiguriert. Mit diesem kann ein gewünschter Port der MCU überwacht werden und je nach Einstellung wird überprüft, ob an diesem Port eine *active-low*- oder *active-high*-Polarität besteht. Sobald der Schalter betätigt wird, öffnet sich der Stromkreis und eine *active-low* Situation tritt ein. Dadurch wird das Interrupt-Flag auf logisch 1 gesetzt und eine *Interrupt Service Routine* (ISR)wird aufgerufen. In dieser wird dem ESC ein 1,0ms PWM-Signal übermittelt, bis der Schalter wieder gelöst wird und kein *active-low* mehr besteht. So lange dieses PWM-Signal übertragen wird, können die Motoren nicht gestartet werden. Anschließend kann mit dem Messbetrieb fortgefahren werden. Auf die Funktionsweise der Software des Notausschalters wird genauer in Abschnitt 8.3.1 eingegangen.

8.2.5 Port-Belegung

Die Belegung der Ports der Peripherie des MC-boards können nicht individuell belegt werden, sondern werden bei Benutzung des *Configuration Wizard 2* automatisch zugewiesen, sobald ein Peripherie-Baustein mit diesem konfiguriert wird.

Peripherie	MCU-Port	Bezeichnung	Funktion
PCA0	P0.1	CEX1	PWM Motor1
	P0.2	CEX2	PWM Motor2
PCA1	P0.3	CEX6	Capture Reflexlichtschranke 1
	P0.6	CEX7	Capture Reflexlichtschranke 2
UART0	P0.4	UART0_TX	Bluetooth-Receive: EGBT-046S RX
	P0.5	UART0_RX	Bluetooth-Transmit: EGBT-046S TX
	P1.7	INT0	Externes Interrupt für Not-Aus
	P2.0	ADSK	LCA: PD_SCK Port
	P2.2	ADDO	LCA: DOUT Port

8.3 Software[13]

In diesem Abschnitt werden die an der Software geänderten Funktionen erklärt. Die Verwendung von Interrupts spielen dabei eine wichtige Rolle.

8.3.1 Programmablauf

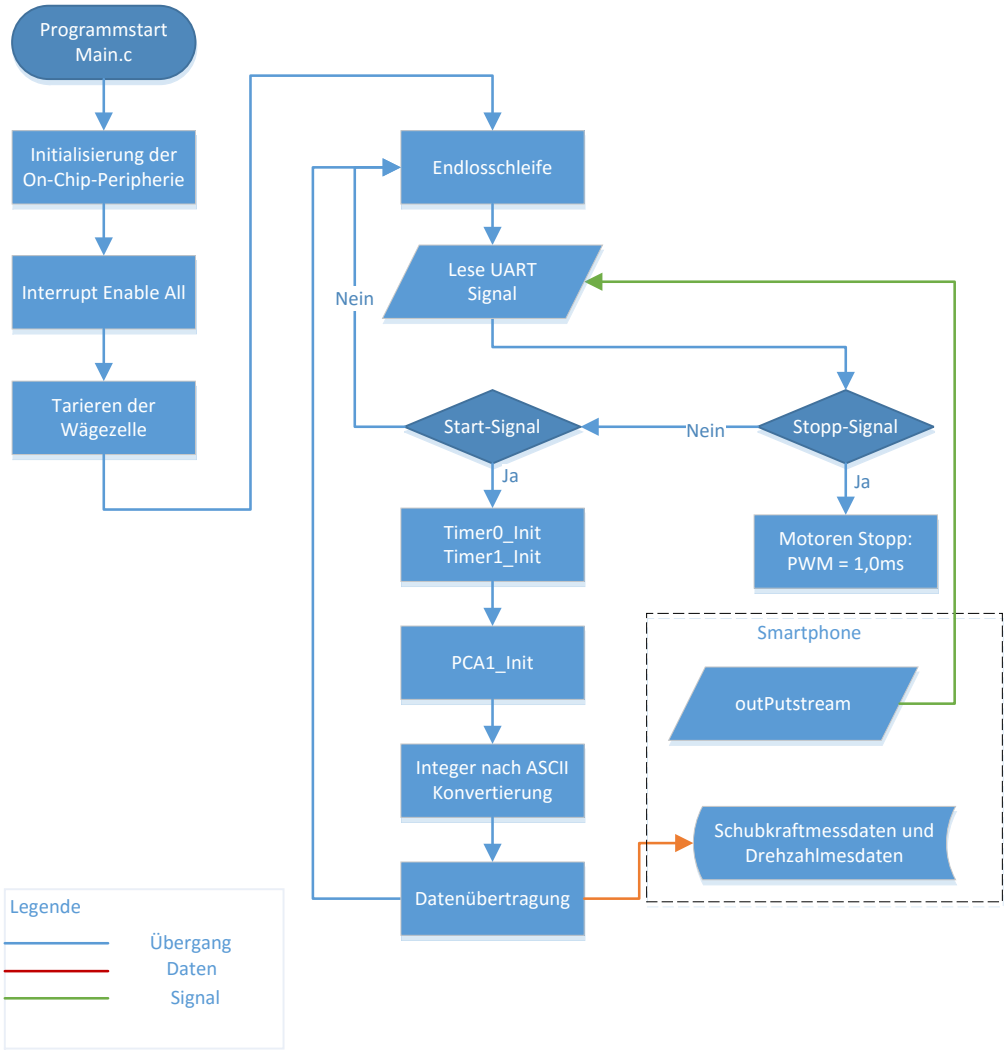


Abbildung 8.12 Programmablaufplan Main.c

Gemäß im Programmablaufplan in der Abbildung 8.12 dargestellt, unterscheidet sich der Programmstart nicht von der Vorgängerversion der Software[14]. Nach der Initialisierung der Peripherie wird die Wägezelle tariert. Änderungen im Programm finden für die Ansteuerung der Motoren und die Durchführung der

[13] Der Quellcode kann im Anhang betrachtet werden.
[14] In der in Abschnitt 8 erwähnten Bachelorarbeit

Schubkraftmessungen statt. Da der Vorgang künftig per mobiler Applikation gestartet werden soll, wurde das Programm entsprechend angepasst. Eine Möglichkeit ist die Übermittelung eines *ASCII*-Zeichens über die UART-Schnittstelle, welches das C-Programm in einer *If*-Abfrage vergleicht. Sobald eine Übereinstimmung des empfangenen Zeichens mit dem festgelegten Zeichen erfolgt, können die von der Abfrage abhängigen Funktionen starten.

Für das Starten der Motoren und der Schubkraftmessungen wird in der *if*-Abfrage zum Abgleich mit dem empfangenen Signal am UART, ein ‚a' festgelegt. Anschließend startet die Initialisierung der Timer 0 und 1. Beide Timer sind so konfiguriert, dass jede 11,1 Millisekunden, je ein Interrupt ausgelöst wird. In der Interrupt Service Routine *ISR_Timer0* wird die Funktion für das Aufzeichnen der Wägezellen-Messwerte ausgeführt. In der *ISR_Timer1* erzeugt der PCA0 ein PWM Signal für die Ansteuerung der Motoren. Nahezu gleichzeitig wird der PCA1 initialisiert und startet die *ISR_Capture* mit der Flankenerfassung, für die Drehzahlmessung.

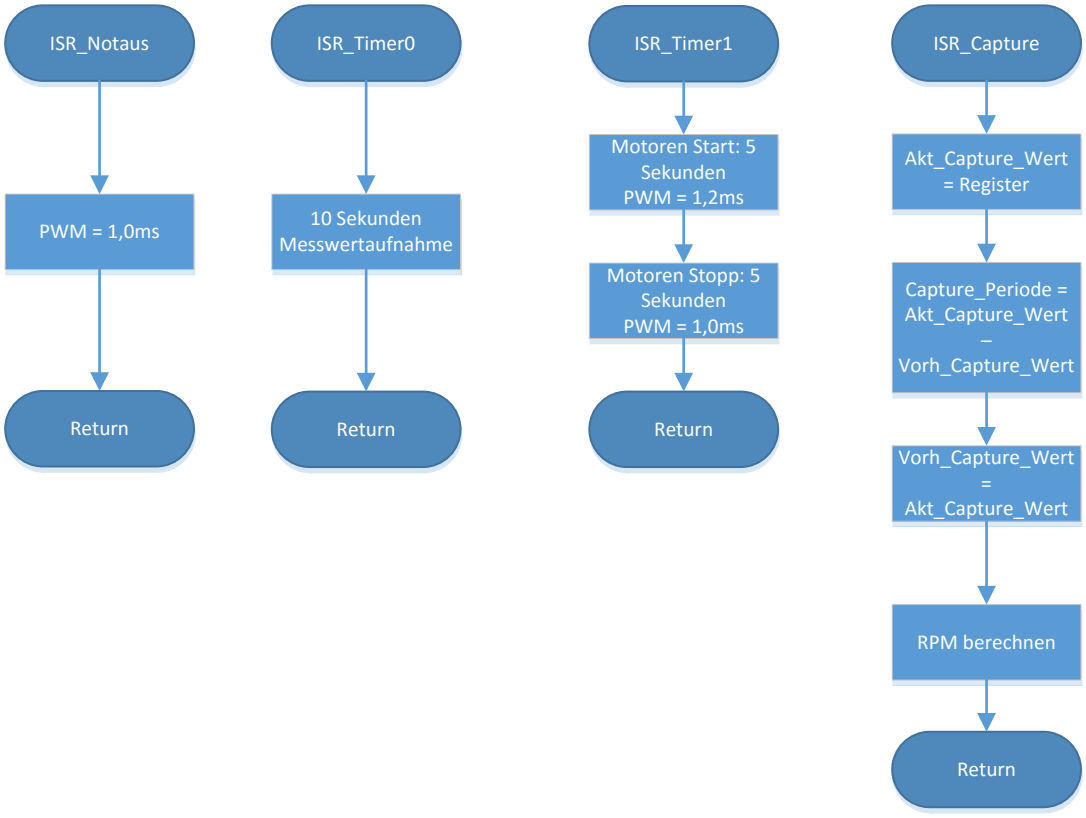


Abbildung 8.13 Programmablauf Interrupt Service Routinen

Timer 0 und 1 / ISR_Timer0 und ISR_Timer1

Beide Timer sind im 16-Bit Modus konfiguriert. Entsprechend werden in ein High und Low-Byte[15] die Initialisierungswerte des Timers eingegeben.

Die Initialisierungswerte werden wie folgt ermittelt folgende:

- Ein Maschinenzyklus stellt $\frac{1}{12}$ der eingestellten Prozessor-Taktfrequenz von 12 Mhz dar, also 1 MHz entsprechen 1 µs.
- Im Initialzustand zählt der Timer somit von 0000h bis FFFFh (also von 0 auf 65535) in 65,535ms.
- Um die Überlaufzeit des Timers einstellen zu können, muss diese dem Wert 65535 subtrahiert werden, damit der Timer nicht mehr von 0 auf 65535 zählt, sondern von 65535 – 11111 = 54424 auf 65535.
- Der Wert 54424 kann nun in hexadezimaler Form in High- und ein Low-Byte des Timers geschrieben werden und dieser zählt bis 11,1 ms.

Die Interrupt Service Routine wird demnach gestartet, sobald das Interrupt-Flag durch einen Timer-Overflow ausgelöst wurde. In der ISR werden zunächst die Timer wieder initialisiert, da im 16-Bit Modus keine *Auto-Reload*-Funktion verfügbar ist, vgl. [18].

Die Wägezellen-Messwertaufnahme wird mit der ISR von Timer 0 gesteuert. Um die gewünschte Messdauer von zehn Sekunden zu erreichen, wird ein Zähler inkrementiert. Durch das Inkrementieren des Zählers, kann die ISR eine definierte Anzahl von Durchgängen absolvieren. Bei einer Überlaufzeit von 11,1 ms sind somit 901 Durchgänge notwendig, um auf etwa zehn Sekunden zu kommen.

Dasselbe Verfahren gilt für Timer 1 und dessen ISR, um je fünf Sekunden lang zwei verschiedene PWM-Signale zu senden. Ist jeweils die erstrebte Dauer erreicht, wird der Timer gestoppt und das Interrupt-Flag auf logisch 0 gesetzt, worauf die ISR gestoppt wird. Danach wird der Programmablauf in der *main.c* fortführt.

Für diese Anwendung, ist das Verwenden von Interrupts dem *Polling* gegenüber, vorteilhafter. Im vorigen Programmablauf wurden die Messwerte im Speicher der MCU gespeichert, bevor sie über UART gesammelt gesendet werden. Das ermöglicht keine Entkoppelung von Messwertaufnahme und Betrieb der Motoren. Durch diese Entkoppelung ist es möglich Messungen durchzuführen, auch wenn die Motoren nicht mehr laufen. Damit sind die Schubkraftmessung und entsprechend die Sprungantwort aussagekräftiger. Darüber hinaus ist *Polling* ungünstig, beim wiederholten Abfragen eines externen Signals und in Hinblick auf die Effizienz eines Programmes.

[15] Timer 0: TL0, TH0 und Timer 1: TL1, TH1

Datenübertragung an die Smartphone-App

Am Ende eines Durchlaufes der Endlosschleife werden die Daten der Schubkraft und der Drehzahl an die Applikation gesendet. Damit die Daten auch vom Smartphone verarbeitet werden können, werden die erhaltenen Messwerte aus dem *Integer*-Format in *ASCII*-Zeichen konvertiert und als eine Daten-Zeichenkette versendet. Damit die Applikation später erkennt, welche Stellen in dieser Zeichenkette der Schubkraft und welche der Drehzahl entsprechen, werden Trennzeichen verwendet. Zusätzlich werden Zeichen verwendet, um den Anfang und das Ende einer einzelnen Kette zu markieren. Die genaue Verarbeitung der Daten mit der Applikation und deren Darstellung werden in Abschnitt 9.3.2 erklärt.

Motoren stoppen

Das Stoppen der Motoren übernimmt eine separate Funktion, welche unabhängig von den Interrupts ist. Hier wird ebenso ein *If*-Abgleich in der Endlosschleife des Hauptprogrammes durchgeführt. Sobald ein anderes Symbol als *‚a‘* vom MC-Board empfangen wird, wird ein 1,0 ms PWM-Signal übermittelt und die Motoren stoppen. Bei jedem Stoppvorgang werden alle Timer und Interrupts zurückgesetzt, damit ein Neustart möglich ist, ohne die MCU zurücksetzen[16] zu müssen.

8.3.2 Änderungsprotokoll der Software

Im Laufe der Weiterentwicklung der Software haben sich mehrere Versionen ergeben. Damit ein Überblick über die Änderungen vorhanden bleibt, wurde bei jeder maßgebenden Optimierung eine neue Version erzeugt.

Tabelle 8.1 Änderungsprotokoll der Software

Version	Änderungen
1.0	Einführung Not-Aus
2.0	Einführung von Interrupts
2.1	Verschiedene Optimierungen in der Organisation des Codes
2.2	Einführung der Drehzahlmessung
3.0	PWM-Signal für 2 Motoren
3.1	Vorbereitung für die App
3.2	Verschiedene Optimierungen in der Organisation des Codes

[16] Also entweder durch ein Reset oder durch Trennen der Versorgungsspannung des MC-Boards

9 SMARTPHONE APPLIKATION – CONTROL CENTER

Um die Sicherheit des Benutzers zu gewährleisten und die Bedienbarkeit zu vereinfachen, wird das System über eine Smartphone Applikation gesteuert und ausgewertet. Somit kann sich der Nutzer außerhalb des Gefahrenbereichs der Motoren begeben, um eine Messung durchzuführen. Zu beachten ist jedoch die Übertragungsdistanz von Bluetooth von etwa zehn Metern.

Die Funktionen der App sind folgende:

- Verbindung mit MC-Board aufbauen
- Steuern der Motoren und simultane Messwertaufnahme
- Messdaten verwalten

Zuvor wurde die Aufgabe des Startens von einem Taster auf dem Mikrocontroller-Board übernommen. Stoppen war damit nicht möglich. Es musste gewartet werden, bis die Motoren das nach fünf Sekunden ausführen.

Mit Hilfe der UART-Schnittstelle und einer *if*-Funktion, können Daten ausgewertet werden und auf ein gesendetes Zeichen überprüft werden. Sobald per UART das Start-Signal von der Applikation erhalten wird, kann eine Messung mit gewohntem Ablauf durchgeführt werden.

9.1 Auswahl der Software-Entwicklungsumgebung – SDK

Zur Implementierung von mobilen Applikationen gibt es eine Vielzahl von Entwicklungsumgebungen. Bei der Auswahl wurden verschiedene Betriebssysteme und deren Endgeräte, sowie die Art der zu programmierende App betrachtet.

Es gibt unterschiedliche Arten von Applikationen:

Native Applikationen

Native Apps können direkt mit der Entwicklungssprache und Schnittstelle der jeweiligen Plattformt programmiert werden. Bei Android ist es *Java* und *Android Studio* mit dem Android *Software Development Kit* (SDK). Der wesentliche Vorteil ist die im Vergleich zu anderen, leistungsfähigere Entwicklungssprache. Zum Beispiel kann mit *Java*, *Android Studio* und den integrierten *Android-Plug-Ins* auf schnelle Weise, auf die Bluetooth-Schnittstelle des Smartphones zugegriffen werden. Ein großer Vorteil von *Android Studio* und *Java* ist ebenfalls die

vorhandene Online-Gemeinschaft, in der umfangreiche Hilfe gefunden werden kann. Dadurch wird der Einstieg in eine neue Programmiersprache und Entwicklungsumgebung erleichtert.

Hybride Applikationen

Mit der SDK von Intel[17], können hybride Apps programmiert werden, welche auf allen gängigen Geräten laufen. Diese benötigen nur einen Quelltext, welcher in *Javascript* und *HTML5* bzw. *CSS* implementiert wird. Das Grundgerüst solcher Apps besteht aus einem Internet-Browser, welcher den *HTML5*-Code lädt. Durch die Schnittstelle des Browsers kann ein *Script* auf die Hardware und auf betriebssystem-spezifische Funktionen zugreifen. Der Nachteil bei hybriden Apps ist die Leistungsfähigkeit und das notwendige Integrieren von *Plug-ins*, um auf alle Funktionen und Peripherien der jeweiligen Plattform zugreifen zu können. Des Weiteren ist die Unterstützung mit Tutorien und vorhandenen Beispielen bei *Intel XDK* sehr gering und unübersichtlich.

Android Studio

Zur Implementierung der Mobilen Applikation wurde entschieden, *Android Studio* von *Google Inc. z*u verwenden. Die Programmiersprache ist Java für die Implementierung der Programmfunktionen, und XML für die Erstellung des Layouts, also der Benutzeroberfläche. Mit dem integrierten Layout-Editor ist es möglich, per *drag-and-drop*[18], *Userinterface*-Komponenten in eine Vorschau-Ansicht zu schieben. Das vereinfacht die Gestaltung der App in Hinsicht auf die Funktionalitäten und das Design. Das Grundgerüst einer App, welche mit Android Studio programmiert wird, sind *Activities*, auch Java-Klassen genannt. Eine *Activity* ist im Grunde eine Bildschirmseite der App, auf der die Benutzeroberfläche repräsentiert ist, und in dessen Hintergrund implementierte Programmabschnitte ausgeführt werden.

Android ist ein *Multithreading*- Betriebssystem, mit dem mehrere *Threads* parallel ausgeführt werden. Das sind Teilprozesse, welche parallel innerhalb eines Hauptprozesses laufen. Der Haupt-*Thread* steuert die Ereignis-Steuerung und Interaktion mit der Benutzeroberfläche. Alle anderen zusätzlichen Aufgaben, welche in der App gestartet werden sollen, starten im selben Haupt-*Thread*.

Unter keinen Umständen darf ein *Thread* andere Aufgaben in deren Ausführung im Haupt-*Thread*, beispielsweise durch eine zeitkonsumierende Teilaufgabe, aufhalten. Dadurch stürzt die Applikation ab und muss neu gestartet werden. Diese Aufgabe sollte in einem separaten *Thread* gestartet werden. Bei einer Datenübertragung zum Beispiel, kann während der Ausführung des Haupt-*Threads* in der Applikation nicht auf einen Dateneingang gewartet werden. Dafür

[17] *Intel XDK*
[18] Zu Deutsch: „Ziehen und Ablegen“

muss ein neuer *Thread* gestartet werden in dem ein *Handler* neue Daten verarbeitet, sobald diese über den Datenstrom ankommen.

Der *Handler* ist dazu da diese Aufgaben im separaten *Thread* zu steuern, sodass dieser bei dessen Start nicht das Userinterface aktualisieren kann. Das Userinterface sollte immer vom Haupt-*Thread* aktualisiert werden. Der mit dem Userinterface interagierende *Thread* muss also mit dem Haupt-*Thread* synchronisiert werden, was der *Handler* übernimmt. Dieser koordiniert die erhaltenen Daten oder Befehle, sodass es keinen Konflikt mit den im Haupt-*Thread* ausgeführten Aufgaben gibt.

Zusammengefasst wird der *Handler* im Haupt-*Thread* eingesetzt indem er die Aktualisierungen der Benutzeroberfläche, in Abhängigkeit der erhaltenen Nachrichten, aus einem separaten *Thread* koordiniert.

Das Manifest

Im Dateienverzeichnis jeder Applikation ist eine Datei namens *Manifest.xml* Voraussetzung. In dieser sind grundlegende Informationen über die Applikation für das Ausführen im Android-Betriebssystem hinterlegt. Ohne diese kann die App nicht gestartet werden. Unter anderem beinhaltet sie Informationen über die Mindest-Android-Version[19] die ein Zielgerät haben muss. Anzeigename sowie -symbol, von der App definierte Komponenten wie *Activities* und auch Hintergrund-Dienste und Inhaltsanbieter werden im Manifest definiert.

Ein Wichtiger Teil der Manifest-Datei sind Berechtigungen, welche der Applikation erteilt werden. Diese werden benötigt um beispielsweise mit der Applikation Zugriff auf das Internet oder auch auf Bluetooth des verwendeten Gerätes zu haben.

9.2 Kommunikation zwischen MC-Board und App

Für die Übermittlung der Schubkraftmessdaten an Simulink wurden bisher zwei Bluetooth-Module verwendet, welche die UART Kommunikation zwischen Computer und MC-Board ermöglichen. Die Bluetooth-Module dienten nur der drahtlosen seriellen Kommunikation zwischen Computer und MC-board.

Die Übertragung der Daten an ein Smartphone kann auch über Bluetooth geschehen. Der Bluetooth-Adapter des mobilen Gerätes, sei es ein Tablet PC oder ein Smartphone, kann eine Verbindung mit dem EGBT-Bluetooth-Modul herstellen. Hierzu muss auf dem Empfangsgerät ein Bluetooth Socket eingerichtet werden. Damit die Bluetooth Schnittstelle von der App angesprochen werden kann, ist im Manifest eine Zugriffs-Erlaubnis zu hinterlegen.

[19] Zum Beispiel Android 1.6, Donut. Eine der ersten Versionen.

Für die Benutzung von Bluetooth gibt es zwei Berechtigungen:

- *BLUETOOTH*: Erlaubt die grundlegenden Funktionen der Bluetooth-Kommunikation zu verwenden. Das sind die Verbindungsanfragen, das Akzeptieren einer Verbindung und die Datenübertragung.
- *BLUETOOTH_ADMIN*: Hiermit lässt sich die Suche nach anderen Endgeräten starten und die Einstellungen verändern.

Verbindung herstellen und Datenübertragung

Für die Bluetooth-Kommunikation muss das Endgerät über einen Bluetooth-Adapter verfügen, was beim Starten der App überprüft wird. Ist die Überprüfung erfolgreich, kann im nächsten Schritt kontrolliert werden, ob der Adapter aktiv ist. Ist das nicht der Fall, wird der Benutzer dazu aufgefordert, Bluetooth zu aktivieren, damit der Anschluss verwendet werden kann.

Anschließend sucht das Smartphone nach Bluetooth-Geräten. Sobald ein verfügbares Gerät ausgewählt wird, kann sich mit dem gewünschten Gerät gepaart und dessen Informationen abgerufen werden. Darunter befinden sich der Gerätename und dessen Hardware-Adresse, auch *Media Access Control*-Adresse (MAC) genannt. Mit dem Koppeln werden diese Informationen zwischen den Endgeräten ausgetauscht und eine Authentifizierung mit PIN, falls notwendig, durchgeführt. Die Informationen des gepaarten Gerätes sind auf dem Smartphone gespeichert und können abgerufen werden. Die bekannte MAC-Adresse kann in der Applikation zwecks Verbindung verwendet werden ohne, dass jedes Mal ein neuer Kopplungsvorgang erfolgen muss.

Bluetooth Socket

Ein Kommunikationsendpunkt eines Netzwerkes, auch *Socket* genannt, ist eine Adressen-Struktur und bezeichnet den logischen Endpunkt einer Verbindung. Ein Socket ist mit einer definierten Datenstruktur und dem *Universal Unique Identifier*[20] (UUID) definiert. Die Adresse und das Transportprotokoll sind ebenfalls wichtige Informationen für die Konfiguration eines *Sockets* für ein System. Die UUID „ist ein Standard für Identifikationen, der in der Softwareentwicklung verwendet wird“, [19]. Die meisten *Sockets* werden heutzutage basierend auf dem Internetprotokoll (*TCP/IP*) verwendet und werden daher auch als Internet Sockets bezeichnet.

Ein *Socket* ist im weiteren Sinn ein Software-Werkzeug, mit dem ein lokales Programm Befehle an eine Programmierschnittstelle (API[21]) geben kann, um eine Verbindung zu verwenden. Zum Beispiel können Sockets Befehle geben wie: „sende diese Daten an diesen bestimmten Socket“. Dementsprechend gibt es

[20] Auch der Port genannt
[21] *Application Programming Interface*

einen *Server Socket* und einen *Client Socket*. Der Server wartet auf Daten und stellt auch Daten für die Socket-Clients bereit. Der Socket-Client hingegen steuert und initiiert ein- und ausgehende Bluetooth-Verbindungen. Die Funktionsweise ist in Abbildung 9.1 dargestellt:

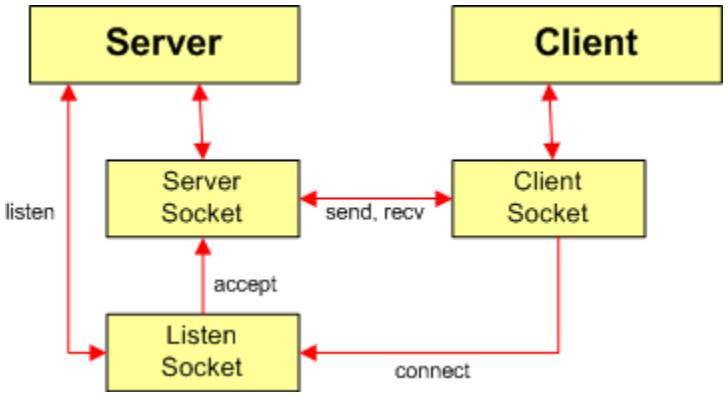


Abbildung 9.1 Funktionsweise eines Sockets, [20]

Der bekannteste Bluetooth-*Socket* ist die *RFCOMM* (*Radio Frequency Communication*), welcher auch von *Android* unterstützt wird. Da die Daten, welche vom MC-Board kommen, über eine serielle UART-Schnittstelle gesendet werden, eignet sich das *Bluetooth Serial Port Protokoll* (SPP), welches auf der RFCOMM basiert, besonders. Die UUID, muss für beide Socket-Teilnehmer dieselbe sein. Bei der Verwendung des Bluetooth SPP kommt üblicherweise eine allgemein verwendete UUID zum Einsatz, die bei der Programmierung der Android-Applikation bereits festgelegt werden kann. Sobald eine Verbindung mittels *RFCOMM-Socket* hergestellt ist, können Informationen über diesen gesendet und empfangen werden.

Datenübertragung

Sobald eine Verbindung steht können Daten, oder wie in diesem Fall benötigt, einzelne Zeichen übertragen werden. Um mit Hilfe des erstellten Sockets Daten lesen und auch schreiben zu können, müssen ein *Input*- und *Outputstream* softwaretechnisch implementiert werden. Mit den Funktionen *read ()* und *write ()* kann der Eingangsdatenstrom gelesen oder Daten geschrieben werden. Gelesene Daten werden in einem Puffer gespeichert und können in eine Variable, zum Weiterverarbeiten geschrieben werden. Wie die Daten zur Visualisierung in der Applikation verarbeitet werden ist in Abschnitt 9.3.2 erläutert.

9.3 Klassen der Applikation – *Activities*

Activities sind Bildschirmseiten einer Android-Applikation. Das Android Betriebssystem verwaltet diese Bildschirmseiten auf einem Stapel. Wenn eine neue Seite gestartet wird, wird diese auf den Stapel gelegt. Sobald die *Zurück*-Taste betätigt wird, wird die zuletzt aktive Bildschirmseite aus dem Stapel angezeigt und aktiv.

Die *Activities* stellen mehrere Methoden zur Implementierung der Software bereit, die je nach Phase im Lebenszyklus der *Activity* relevant sind. Die Methoden können dem Flussdiagramm im Anhang 14.2.3 entnommen werden. O*nCreate ()* ist zum Beispiel wichtig zum Laden der Benutzeroberfläche und dessen Hintergrundprozesse, sobald die Seite geöffnet wird. In ihr sind alle Funktionen und Layout-Komponenten zu definieren, die für die Anwendung geladen werden sollen.

9.3.1 Device List Activity

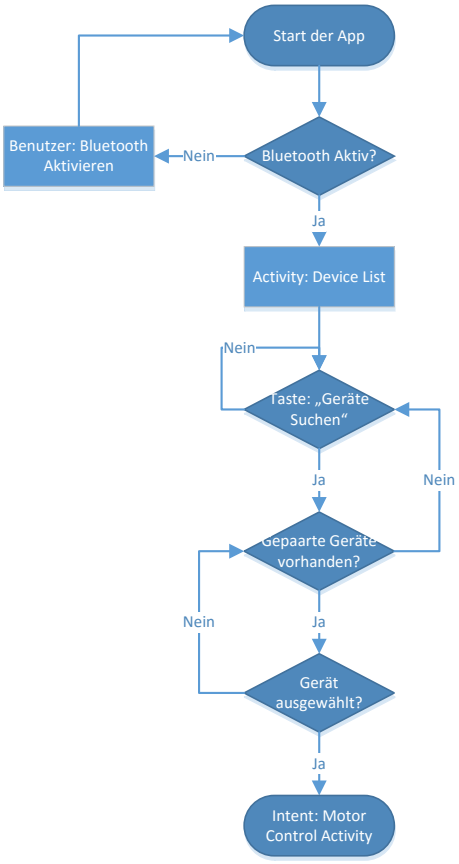


Abbildung 9.2 Programmablaufplan Device List Actitvity

Durch das Betätigen der Taste *Scan Devices* wird im *Device-List*-Bildschirm (siehe Abbildung 9.3) dem Benutzer angezeigt, welche gekoppelten Geräte derzeit verfügbar sind. Wenn ein Gerät angezeigt wird, kann eine Verbindung mit dem Gerät, hier das Bluetooth-Modul EGBT, hergestellt werden. In dieser *Activity* wird auch kontrolliert, ob der Bluetooth-Adapter aktiv ist und wenn nicht, wird der Benutzer aufgefordert es zu aktivieren. Das Verbinden leitet den Übergang zum Motor-Control-Bildschirm ein. Dieser Übergang ist mit der *Intent*-Methode möglich. Sie lädt und öffnet den App-Kontext der *Activity* in ein neues Fenster, sobald eine Taste gedrückt wird.

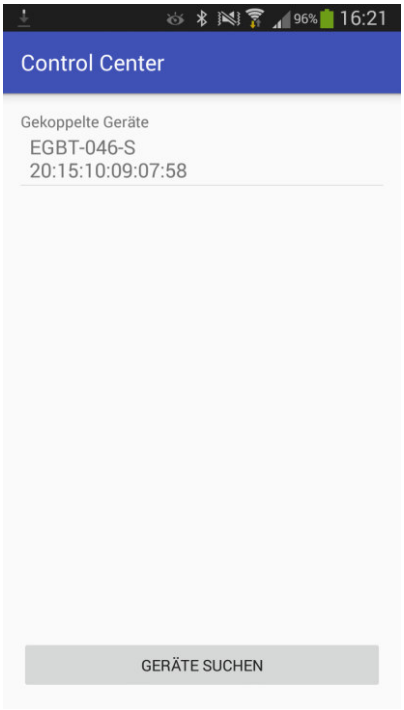


Abbildung 9.3 App-Bildschirm1: Device List

9.3.2 Motor Control Activity

Nachdem in der *Device-List-Activity* eine Verbindung mit einem Gerät angestoßen ist, geht die Applikation in den *Motor-Control*-Bildschirm über. Hier wird der Bluetooth Socket erstellt und der *Inputstream* dieser Verbindung ausgelesen. Mit dem Schreiben in den *Outputstream* können Daten bzw. einzelne Zeichen an die UART-Schnittstelle des MC-Boards gesendet werden. Der Ablauf des Programmes ist in Abbildung 9.4 dargestellt.

Die Funktionen für die Verwendung und Auswertung der *Activity* sind folgende:

- Start-Verriegelung
- Start/Stopp der Motoren
- Empfangen der Daten als String und Unterteilung in Substrings, zur Darstellung von momentaner Schubkraft und Umdrehungszahl
- Speichern der Daten in einem Array und Export aus diesem in eine CSV-Datei
- Bluetooth-Verbindung trennen

Ein wichtiges Werkzeug ist die Methode *onClickListener()*. Er überwacht während der Verwendung der App, ob eine Taste betätigt wird, und führt die der Taste hinterlegte Funktion aus.

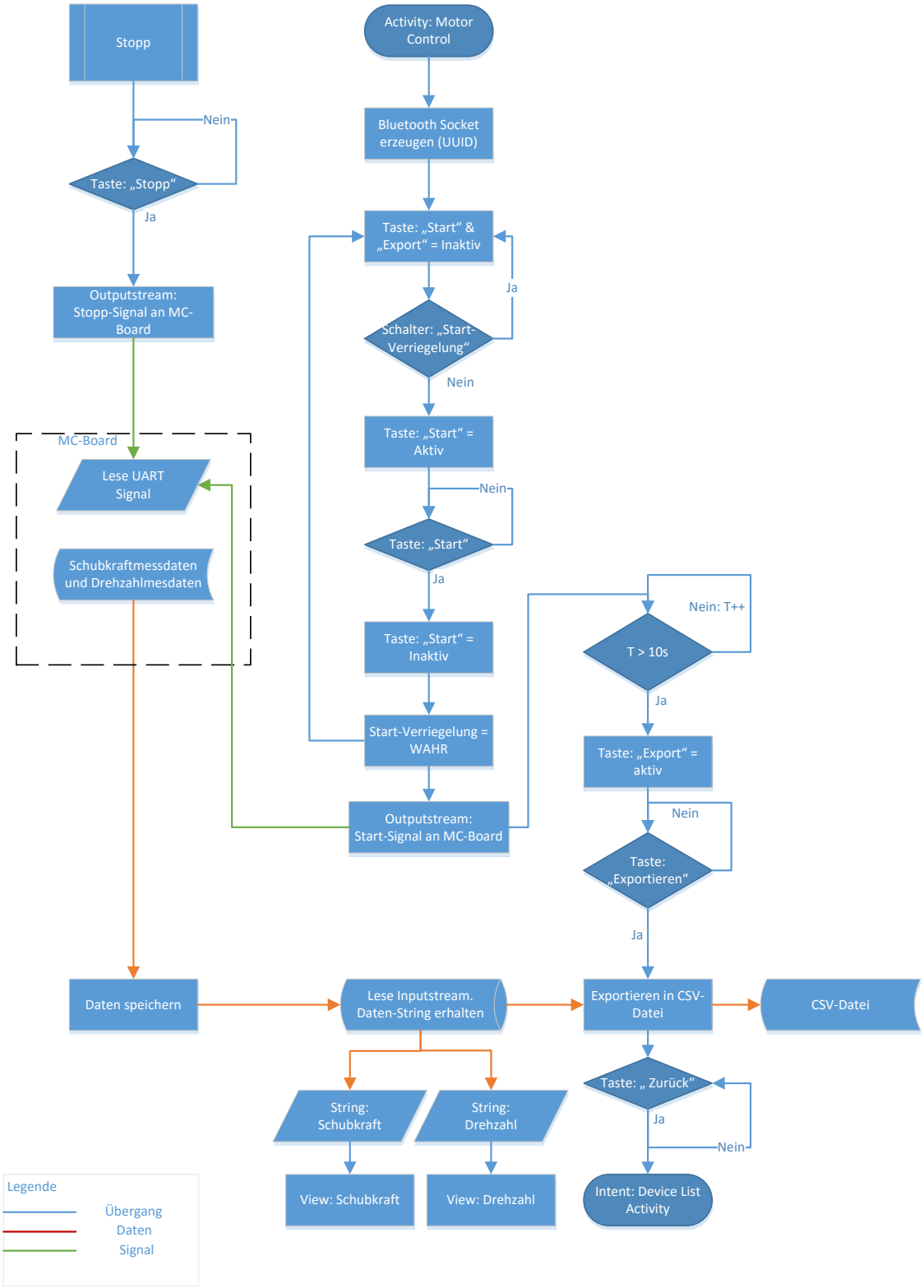


Abbildung 9.4 Programmablaufplan Motor Control Activity

Start-Verriegelung

Sobald die Motor Control *Activity* geöffnet ist, können die Motoren gestartet werden. Da die Start-Taste zu Beginn noch nicht aktiv ist (siehe Abbildung 9.5), wird verhindert, dass aus Versehen ein Motorenstart ausgelöst wird. Die taste wird aktiv, sobald die Start-Verriegelung, mit einem *Switch* dargestellt, gelöst wird. Softwaretechnisch ist diese Verriegelung mit einer *Boolean*-Variable und einem *onClickListener()* gelöst. Ist diese Variable *false*, ist die Funktion der Start-Taste inaktiv und ein Warnhinweis wird angezeigt. Ist die Variable *true*, sind die Funktionen dieser *Activity* verfügbar (siehe Abbildung 9.6).

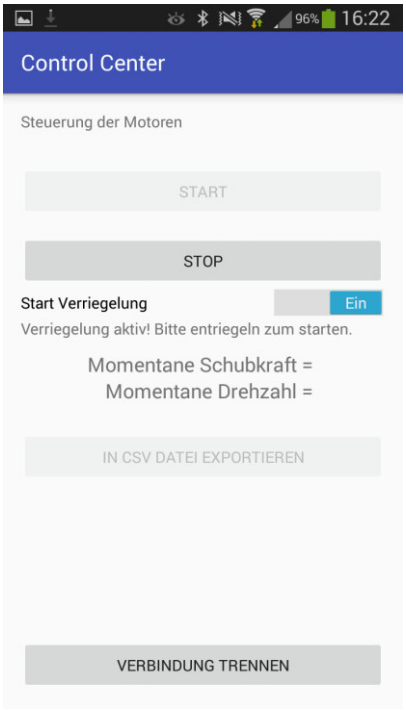


Abbildung 9.5 App-Bildschirm 2: Motor Control, Start-Sperre

Start/Stopp der Motoren

Mit dem Auslösen eines Starts wird mit der *write()*-Funktion über den *Outputstream* das Zeichen *‚a‘* an die UART-Schnittstelle des MC-Boards gesendet. Damit nicht irrtümlich ein zweites mal gestartet werden kann, wird die Start-Taste wieder inaktiv und die Start-Verriegelung auf „EIN“ gestellt. Dasselbe Prinzip gilt für das Stoppen, bei dem über den *Outputstream* das Symbol *‚b‘* an das MC-Board geschickt wird. Stoppen ist zu jedem Zeitpunkt möglich.

Bei jeder Tasten-Verwendung wird kurzzeitig ein Hinweistext ausgegeben, der dem Benutzer die Auswirkung jener Taste mitteilt.

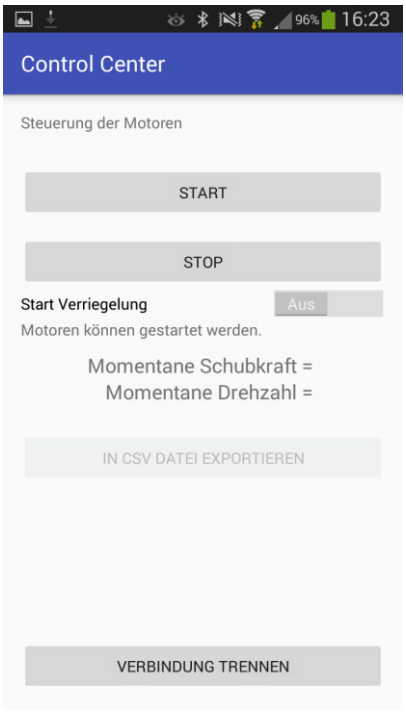


Abbildung 9.6 App-Bildschirm 2: Motor Control, Start-Freigabe

Messdatenaufnahme, -anzeige und -export

Bei Auslösen der Start-Taste wird gleichzeitig die Aufnahme der Messdaten, durchgeführt, welche vom MC-Board gesendet werden. Die Daten werden vom MC-Board, nach einer Konvertierung aus dem Integer-Format, als *ASCII*-Zeichenkette mit 16 Stellen verschickt. Vier Stellen sind jeweils für die Schubkraftwerte und die Drehzahlwerte beider Motoren gedacht. Damit die Applikation erkennt, wann eine Zeichenkette beginnt und endet, werden jeweils ein ‚*#*'-Zeichen und ein ‚~' -Zeichen am Anfang und Ende des *Strings* geschickt. Um die Messwerte voneinander zu trennen, sendet das MC-Board zwischen den jeweils vier Messwertstellen ein ‚*+*' -Zeichen.

Mit Hilfe dieser Trennzeichen können in der Applikation, zur Differenzierung der empfangenen Daten, unter Verwendung von *Substrings*, einzelne Daten-Zeichen-Ketten erzeugt werden. Diese werden zunächst in separaten Arrays gesichert. Im Hintergrund werden, die aus dem *Inputstream* gelesenen und in Arrays gespeicherten Messwerte, in einem separaten *Thread* abgerufen. In diesem werden von einem *Handler* die Anzeigen der Schubkraft und Drehzahlen in der Benutzeroberfläche aktualisiert.

Die drei Zeichen-Ketten-Arrays (für Schubkraft und Drehzahlen beider Motoren) werden je in eine CSV-Datei in ein Dateienverzeichnis der Dropbox auf dem Smartphone exportiert. Von diesem Verzeichnis aus kann Microsoft-Excel auf die Datei zugreifen, mit der die Daten in einem Diagramm zur Analyse der Sprungantwort visualisiert werden können.

Damit nach Ende der Schubkraftmessung die Daten in eine CSV-Datei Exportiert werden können, wird die *Export*-Taste in der Benutzeroberfläche erst nach 10 Sekunden aktiv. Das entspricht der Dauer einer Messung, welche im Programm für das MC-Board festgelegt ist. Sobald die Daten übertragen sind, wird der Benutzer nach Bestätigen der *Zurück*-Taste wieder zur *Device List Activity* geleitet.

Abbildung 9.7 App-Bildschirm 2: Daten exportiert

Verarbeitung der Daten mit Excel

Die exportierte CSV-Datei wird nach erfolgreichem Abschließen einer Schubkraft- und Drehzahlmessung mit Excel verarbeitet. Auf dem unten angegebenen Diagramm ist testweise dargestellt, dass die Übertragung der Daten funktioniert. Da die Übertragungsrate und die Anzahl an übermittelten Werten bekannt sind, können die Datenpunkte für die Zeitachse manuell in die Excel-Liste eingetragen werden.

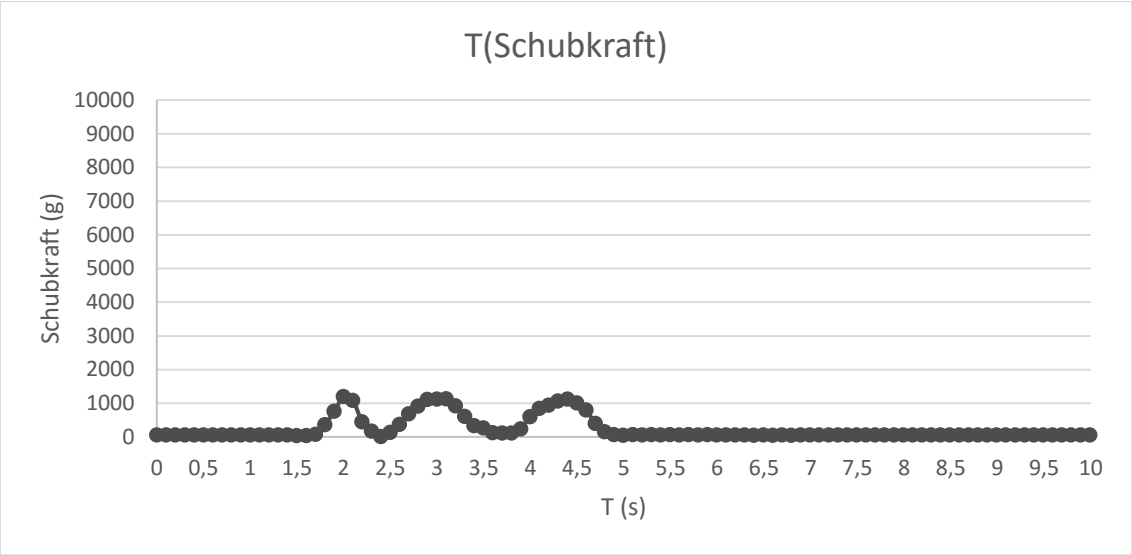


Abbildung 9.8 Test: Schubkraft in Abhängigkeit der Zeit

Die Weiterentwicklung des Prüfstandes in Hard-und Software ist mit der funktionsfähigen Smartphone Applikation abgeschlossen. In den nächsten Abschnitten geht es um die Ermittlung der Sprungantwort und dem Vergleich mit einer zuvor erstellten Simulation.

10 DYNAMISCHE TESTS

In der Masterarbeit „*Entwicklung eines manntragenden Multikopters in Hard-und Software*“[22], wird der Leistungsbedarf ermittelt, welcher für den Schwebeflug des Multikopters notwendig ist. Für eine Simulation wird ermittelt, bei welcher Drehzahl die erforderliche Schubkraft erreicht wird. Nach Durchführung von Schubkraftmessungen mit dem Prüfstand werden die Ergebnisse, mit denen der Simulation verglichen und überprüft, ob der Prüfstand nachvollziehbare Werte wiedergibt.

Schließlich wird die Ausprägung der Sprungantwort nach plötzlicher Änderung der Drehzahl aus dem Stand heraus ermittelt. Ein weiterer wichtiger Punkt bei der Leistungsauslegung des manntragenden Multikopters, ist die Untersuchung des Sprungverhaltens im Schwebeflug nachdem die Drehzahl erhöht wurde.

10.1 Simulation

Leistungsauslegung

Das System muss gewisse Anforderungen erfüllen. Die folgenden Eigenschaften beschreiben die Basis, zur Dimensionierung des Systems[23]:

- Gewicht:

Das Gesamtgewicht des Multikopters beläuft sich auf 50 kg, ohne Pilot und ca. 120 kg mit Pilot. Dabei werden alle für die Konstruktion benötigten Bauteile berücksichtigt.

- Luftschraube:

Die Luftschraube *Xoar Beechwood* 25x12 Zoll, der Firma *Hacker GmbH* hat einen Durchmesser von umgerechnet 63,5 cm und eine Steigung von 30,48 cm. Bei einer Umdrehung würde dieser Propeller theoretisch 30,48 cm zurücklegen. Die Beschleunigung der durch den Rotor strömenden Luftmassen, erzeugt den notwendigen Schub. Die Strahltheorie von Bernoulli, auf der die Leistungsauslegung der Luftschraube beruht, wurde bereits in einer Studienarbeit für die Entwicklung des VC05 Hexakopters ausführlich beschrieben. Ein Versuchsaufbau konnte eindeutige Werte über den Propellerwirkungsgrad liefern.

[22] von M.Sc. Tobias Kuentzle, [17]
[23] Eigenschaften werden in zuvor genannter Masterarbeit genauer dargestellt.

Leistungsauslegung für den Schwebeflug

Die Randbedingungen werden hier nochmals aufgeführt:

Tabelle 10.1 Parameter zur Leistungsauslegung, [17]

Gesamtgewicht (kg)	120
Gewichtskraft (N)	1177
Gewichtskraft pro Motor (N):	98
Rotordurchmesser (m):	0.635
Anzahl Rotoren:	12
Rotorkreisfläche (m²):	3.8
Luftdichte 20°C (kg/m²):	1.293

Die ideale Gesamtleistung des Systems wurde zu 12,9 kW berechnet. Die Motoren müssen entsprechend eine ideale Rotorleistung von 1,07 kW aufbringen. In der Realität müssen jedoch die Verlustquellen, das heißt der Profilwiderstand, die Gleichförmigkeit des Durchflusses, der Restdrall im Luftstrom und die Blattspitzenverluste mit einbezogen werden.

Unter Berücksichtigung des mechanischen Wirkungsgrades von Multikopter-Rotoren (η_{Rotor} = 0,5) beträgt somit die benötigte mechanische Leistung eines einzelnen Motors, mit einem Schub von 98 N, ca. 2,1 kW.

In Bezugnahme auf den Motor- und Drehzahlsteller-Wirkungsgrad ($\eta_{Motor}*\eta_{ESC}$ = 0,9) kann die benötigte elektrische Eingangsleistung des Motors berechnet werden und ergibt 2,39 kW, vgl. [17].

Die Annahmen der Wirkungsgrade stammen aus der zuvor erwähnten Masterarbeit.

Simulation

Für die Simulation wurde zunächst die Luftschraube vermessen und dessen Koordinaten in eine Simulations-Software eingetragen. Mit der Auswahl eines Simulations-Profils und den eingetragenen Koordinaten wurde von der Software ein Luftschrauben-Modell erzeugt, was als Grundlage der Simulation dient. Die Simulation soll eine Auskunft über den notwendigen Motorschub liefern.

Bei einem Drehzahlbereich von 0 bis 5000 1/min werden folgende Resultate erhalten:

- Bei 4700 U 1/min ist der Schub gleich 98 N (= 9,81 m/s² * 9,98 kg)
- Bei 4700 U 1/min ist die mechanische Antriebsleistung 2,1 kW

Die erhaltenen Werte bestätigen die zuvor berechneten Kennwerte der benötigten mechanischen Antriebsleistung des Motors samt ESC und des benötigten Schubs für den Schwebeflug.

Diese Werte gilt es nun in einem realen Test mit dem Prüfstand zu verifizieren.

10.2 Schubmessung und Sprungverhalten

Ausgangslage für die Schubkraftmessungen

Simulationsergebnisse:

- Je Motor benötigter Schub für den Schwebeflug: 98 N (9,98 kg)
- Benötigte Motorleistung: 2,1 kW. Diese ist durch die verwendeten Q80-Motor bereits gegeben.

Der Prüfstand mit Wägezelle ist einsatzbereit und kann mit der Android Smartphone Applikation gesteuert und ausgewertet werden. Nachdem die Messwerte von der Applikation aufgenommen wurden, können diese nach Export in die *Dropbox,* mit *Microsoft Excel* ausgewertet werden[24].

Erwartete Ergebnisse

Die erwarteten Ergebnisse sollen eine Auskunft über das Sprungverhalten bei Änderung der Drehzahl geben.

Es wird getestet, wie sich die Motoren verhalten, wenn aus dem Stand heraus die Drehzahl geändert wird. Damit wird die Totzeit zwischen dem Start-Zeitpunkt der Motoren und dem Zeitpunkt, wenn eine Schubkraft zu verzeichnen ist, ermittelt.

Das ist die Zeit die der Motor benötigt, um die anfangs maximale Haftreibung zu überwinden, um in Gleitreibung überzugehen. Dieser Übergang wird auch *Stick-Slip*-Effekt genannt, vgl. [21] . Nach Überwindung der Haftreibung setzt sich der BLDC Motor in Bewegung. Resultierend kann ein sprungartiger Anstieg der Schubkraft erkannt werden, sobald die Drehzahl erhöht wird. Einen großen Einfluss auf das verzögerte Erreichen der Schubkraft hat ebenso das Trägheitsmoment.

Im zweiten Testszenario soll ermittelt werden, wie sich die Motoren verhalten, wenn im laufenden Betrieb die Drehzahl erhöht wird.

[24] Die Erzeugten Microsoft Excel-Diagramme können auf beigefügter CD betrachtet werden

Messen der Sprungantwort bei einer Drehzahl von ca. 1900 U 1/min

Nach Einstellen eines PWM-Signals von 1,2 ms werden die Motoren gestartet. Folgendes Diagramm in Abbildung 10.1 stellt das Ergebnis dar:

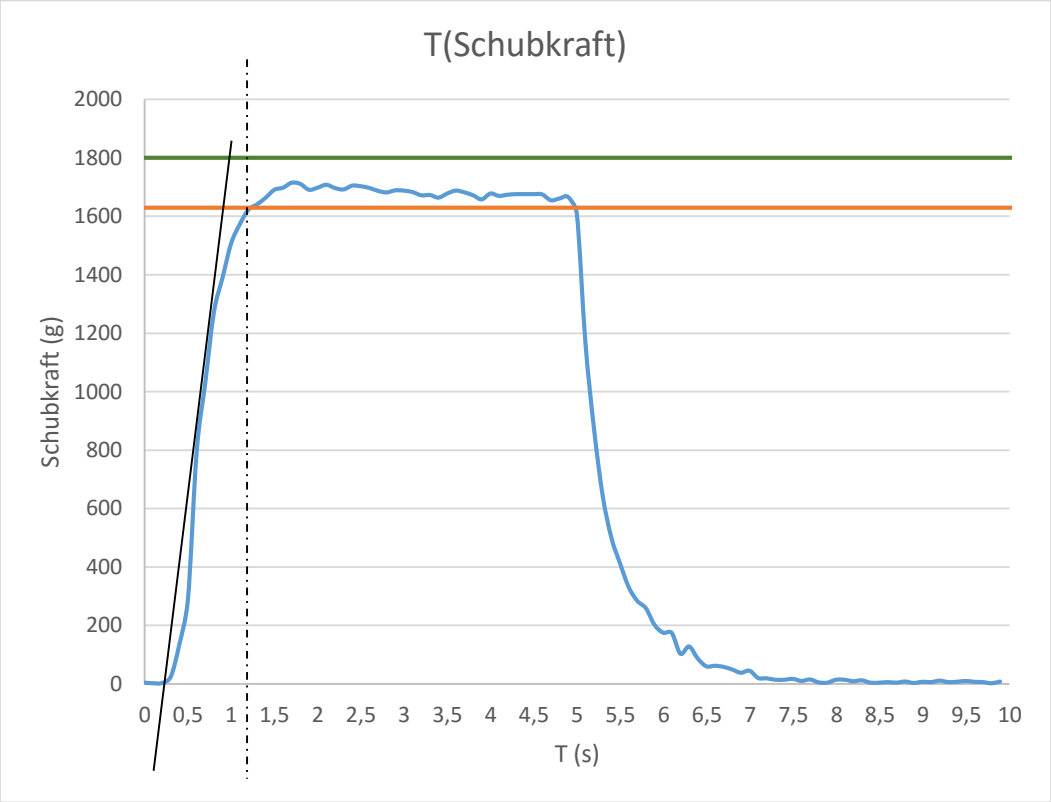


Abbildung 10.1 Sprungantwort bei einer Drehzahl von 1900 U 1/min

Der Toleranzbereich wird bei ±5% der maximalen Schubkraft festgelegt. Durch das Einzeichnen einer Tangente am Wendepunkt der Sprungantwort, kann die Totzeit ermittelt werden. Diese entspricht der Intersektion der Tangente mit der x-Achse. Die Verzögerungszeit hingegen entspricht dem Zeitpunkt, wenn die Kurve die untere Grenze des Toleranzbereiches schneidet.

Die Totzeit stimmt also mit 300 ms überein. Die Verzögerungszeit, bis die Maximale Schubkraft von 1,7 kg erreicht ist, beträgt 1250 ms (also 1550 ms nach Start). Nach fünf Sekunden stoppen die Motoren und die Schubkraft erreicht nach 500 ms wieder ihren Ausgangslevel. Bei Anlauf der Motoren wird der zuvor erwähnte *Stick-Slip*-Effekt beobachtet.

Messen der Sprungantwort bei einer Drehzahl von ca. 4000 U 1/min

Nach Einstellen eines PWM-Signals von 1,5 ms, werden die Motoren erneut gestartet. Folgendes Diagramm Abbildung 10.2 ist das Ergebnis dieser Schubkraftmessung:

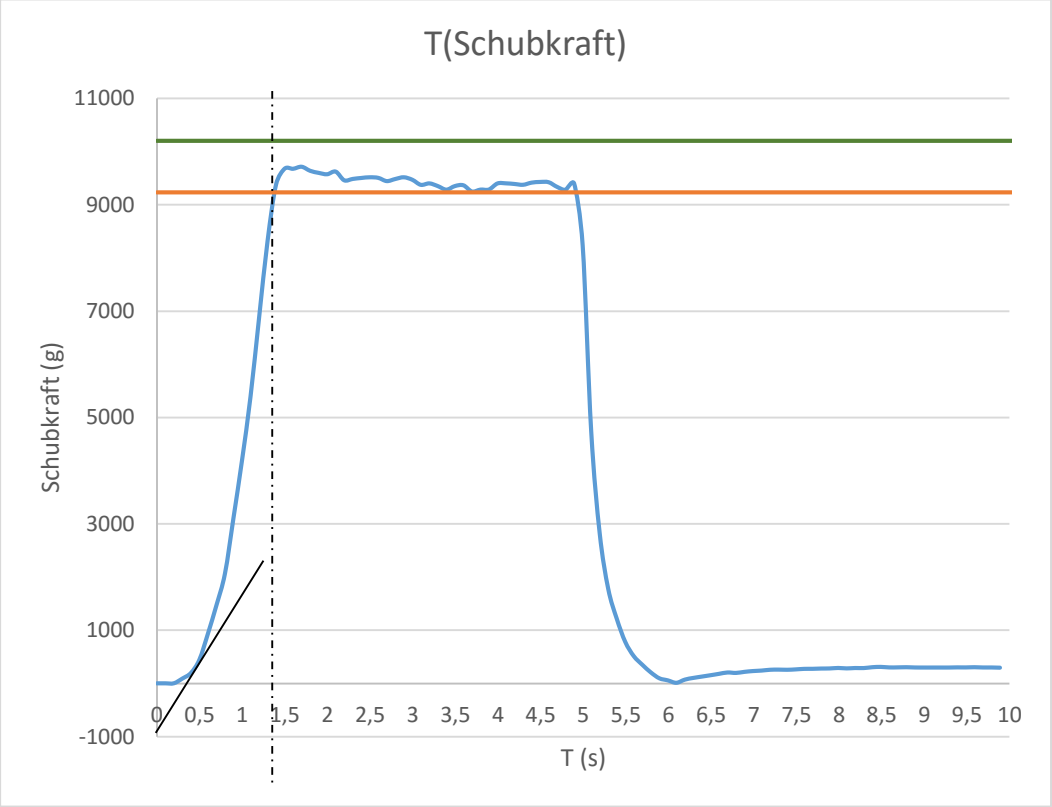


Abbildung 10.2 Sprungantwort bei einer Drehzahl von ca. 4000 U 1/min

Laut Diagramm wird nach ca. 200 ms die Haftreibung überwunden und nach 1200 ms später, die Schubkraft von 9,7 kg erreicht. Nach Abschalten der Motoren und Abnehmen der Drehzahl nimmt die Gleitreibung wieder ab und die Haftreibung zu. Wenn die Haftreibung größer als die Gleitreibung ist, wird die Beschleunigung negativ und die Motoren kommen zum Stopp. Laut Diagramm in Abbildung 10.2 sinkt die Schubkraft nach Abschalten der Motoren nach etwa 400 ms wieder auf null.

Durch Wiederholen der Messung wird auch bestätigt, dass das Ergebnis reproduzierbar ist.

Sprungverhalten bei Drehzahländerung während des Betriebes

Beim Test mit verschiedenen PWM wird ermittelt, wieviel Zeit die Motoren benötigen, um die Drehzahl aus dem laufenden Betrieb heraus zu erhöhen.

Damit wird später für den Schwebeflug ermessen, wie viel Zeit die Motoren brauchen, um für den benötigten Schub zu sorgen. Daneben muss beachtet werden, dass im realen Flugversuch die Motoren die gesamte Last des Multikopters tragen, und diese beschleunigen müssen.

Beim nächsten Test wurde fünf Sekunden lang ein PWM-Signal von 1,2 ms und für die restliche Messzeit eine PWM von 1,3 ms übertragen. Das entspricht einem Drehzahlsprung nach fünf Sekunden von 1900 U 1/min auf 2700 U 1/min.

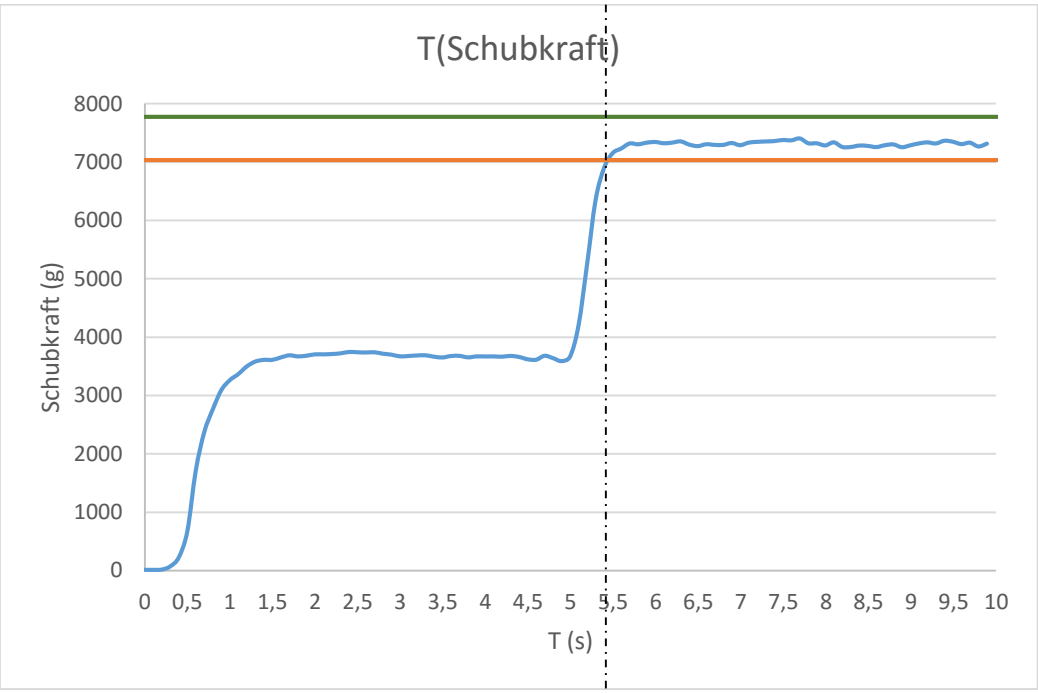


Abbildung 10.3 Sprungantwort bei Steigerung der Drehzahl nach fünf Sekunden

Das Schubgewicht hat sich während der Drehzahländerung von 3,6 kg auf 7,3 kg erhöht. Für die Änderung der Schubkraft haben die Motoren laut Abbildung 10.3 etwa 400 ms benötigt.

10.3 Vergleich von Simulation und realen Tests

Die Schubkraftmessungen mit dem Prüfstand haben die Simulationsergebnisse verifiziert. Diesen zur Folge, wird bei einer Drehzahl von 1900 U 1/min eine Schubkraft von etwa 1,5 kg erreicht. Bei einer Drehzahl von 4000 U 1/min sind es bereits etwa 8,0 kg. Der Unterschied zu den gemessenen Werten ist auf das Rechenverfahren mit der verwendeten Simulationssoftware zurückzuführen.

In dieser können keine externen Einflüsse simuliert werden. Zudem weicht die in der Simulationssoftware „verwendete Geometrie von der tatsächlichen vorliegenden ab“,[17]. Die Messfähigkeit der Wägezelle wurde ausführlich getestet. Demzufolge kann behauptet werden, dass die erhaltenen Schubkraftwerte zuverlässig sind.

11 FAZIT

Eine Voraussetzung für die Zuverlässigkeit und Präzision des Prüfstandes ist die Messfähigkeit der verwendeten Wägezelle. Diese wurde während einer MSA nachgewiesen.

Darüber hinaus konnten erfolgreich Schubkraftmessungen bei verschiedenen Drehzahlen durchgeführt werden. Somit konnten die Berechnungen und eine Simulation aus einer vorangegangenen Arbeit verifiziert werden. Dabei wurde für den Schwebeflug des manntragenden Multikopters eine Schubkraft von ca. 10 kg und eine Drehzahl von 4500 U 1/min pro Motor vorausgesetzt.

Die ermittelten Totzeiten, vor Anstieg der Schubkraft sowie die Verzögerungszeit bei einer Drehzahländerung, sind wichtige Parameter für die Flugregelung eines Multikopters. Mit dem Prüfstand konnten diese Werte bei einigen Tests haltbar nachgewiesen werden. In Zukunft kann der Prüfstand samt Smartphone Applikation für weitere Untersuchungen eingesetzt werden.

Somit wurde, durch die Weiterentwicklung des Prüfstandes in Konstruktion, Hardware und Software, sowie durch die Implementierung der Smartphone Applikation die Aufgabenstellung in allen Punkten erfüllt.

12 AUSBLICK

Bei der Verwendung des Prüfstandes, zur Durchführung von Schubkraftmessungen eines Multikopter-Antriebs treten mehrere Fragestellungen auf. Einige dieser Punkte könnten für weiterführende Arbeiten interessant sein.

Bezüglich der Hardware, wäre es wichtig einige Änderungen in der Konnektivität vorzunehmen. Zum Beispiel wäre es von Bedeutung die Stromzufuhr der Motoren über Akkus, mit einem Schalter außer Kraft setzen zu können. Ferner wäre es wichtig, den Not-Aus-Schalter tragbar zu machen, sodass der Benutzer des Prüfstandes sich mit diesem in Sicherheit begeben kann, um im Notfall das System außer Kraft setzen zu können.

Die Smartphone Applikation muss hinsichtlich der vollständigen Kontrolle über den Prüfstand verändert werden. Beispielsweise wäre es von Vorteil, wenn mit der App unterschiedliche Prüfszenarien realisiert werden könnten.

Die Kommunikation zwischen MCU und App muss besser vernetzt werden, um die Sicherheit noch besser gewährleisten zu können.

Das wären zum Beispiel:

- Feedback vom MC-Board an die App, sobald ein Start- oder Stopp-Signal von dieser empfangen wird.
- Bessere Fehlerfallbehandlung, zum Beispiel bei einem Fehler bei der Übertragung der Messwerte.
- Erstellung eines interaktiven Diagrammes, zur Anzeige und Auswertung der Daten in einem Diagramm anhand der App.
- Exportieren der Daten auf einen zentralen Datenspeicher im Internet, zum Beispiel Google Drive.

13 INDEX

13.1 Formelverzeichnis

13.2 Literaturverzeichnis

[1] N. Dedic, „Entwicklung und Aufbau eines Prüfstandes zur Auftriebsmessung von Luftschrauben," Hochschule Karlsruhe, 2016.

[2] T. Kuentzle und U. Wahl, „Entwicklung eines Hexakopters," Hochschule Karlsruhe, 2014.

[3] „Wheelsaddicts," 12 04 2016. [Online]. Available: http://wheelsaddicts.com/e-volo-volocopter-vc200/. [Zugriff am 03 09 2016].

[4] Wikipedia, „Wikipedia.org," 2013. [Online]. Available: https://de.wikipedia.org/wiki/Volocopter. [Zugriff am 03 09 2016].

[5] H. M. GmbH, „Bedienungsanleitung für Q80-Motoren," Ergolding, 2010.

[6] H. M. GmbH, „Hacker Motor GmbH," [Online]. Available: http://www.hacker-motor-shop.com/hacker/prodpic/Q80-8M-37410008_b_0.JPG. [Zugriff am 02 09 2016].

[7] JETI Model, „JETI Model," 2016. [Online]. Available: http://www.jetimodel.com/de/katalog/Zubehor/JETIBOX/. [Zugriff am 21 07 2016].

[8] JETI-Modell, „JETI-Box mini Beschreibung," [Online]. Available: http://www.hacker-motor.com/daten/anleitungen/MasterSpin-Anleitung.pdf. [Zugriff am 24 Mai 2016].

[9] Bosche Wägetechnik, „Bosche.eu," [Online]. Available: https://www.bosche.eu/waagenkomponenten/waegezellen/plattform-waegezellen/miniatur-waegezelle-h07a. [Zugriff am 01 09 2016].

[10] „wikipedia," [Online]. Available: https://de.wikipedia.org/wiki/Hookesches_Gesetz. [Zugriff am 24 Mai 2016].

[11] M. Meise, „Google Sites," [Online]. Available: https://sites.google.com/site/martinmeise/. [Zugriff am 24 Mai 2016].

[12] „Sparkfun," [Online]. Available: https://www.sparkfun.com/products/13230. [Zugriff am 24 Mai 2016].

[13] Avia Semiconductor, „24-Bit Analog-to.Digital Converter (ADC) for Weigh Scales," Avia Semiconductor, 2016.

[14] Nico, „Pic-Projekte," März 2012. [Online]. Available: http://pic-projekte.de/wordpress/wp-content/uploads/2014/01/DS18S20.png. [Zugriff am 3 August 2016].

[15] Arduino, „Arduino.cc,“ [Online]. Available: https://www.arduino.cc/en/uploads/Products/Uno_SMD_F.jpg. [Zugriff am 15 08 2016].

[16] Wikipedia, „Wikipedia,“ 25 08 2016. [Online]. Available: https://de.wikipedia.org/wiki/Korrelationskoeffizient. [Zugriff am 01 09 2016].

[17] T. Kuentzle, „Entwicklung eines Manntragenden Multikopters in Hard- und Software,“ Hochschule Karlsruhe, 2015.

[18] S. Laboratories, “Datasheet C8051F58x/F59x,” Austin, TX, 2009.

[19] Wikipedia, „Wikipedia,“ 17 12 2015. [Online]. Available: https://de.wikipedia.org/wiki/Universally_Unique_Identifier. [Zugriff am 03 09 2013].

[20] H. Dietrich, „Codeproject,“ 26 01 2005. [Online]. Available: http://www.codeproject.com/Articles/9424/Single-Server-Multiple-Clients-Win-MFC-classes-f. [Zugriff am 01 09 2016].

[21] J. Richter, „Systemmodelierung eines bürstenlosen Gleichstrommotors mit Drehzahlregelung,“ HTW Berlin, 2009.

[22] Individuapp, „Individuapp,“ [Online]. Available: http://individuapp.com/de/android-kurs/die-klassen-activity-und-intent. [Zugriff am 08 2016].

[23] „Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Dehnungsmessstreifen. [Zugriff am 24 Mai 2016].

[24] W. Müller, „Carl Enger Schule,“ März 2009. [Online]. Available: http://www.carl-engler-schule.de/culm/culm/culm2/th_messtechnik/sensoren/dehnungsmessstreifen.pdf. [Zugriff am 24 Mai 2016].

14 ANHANG

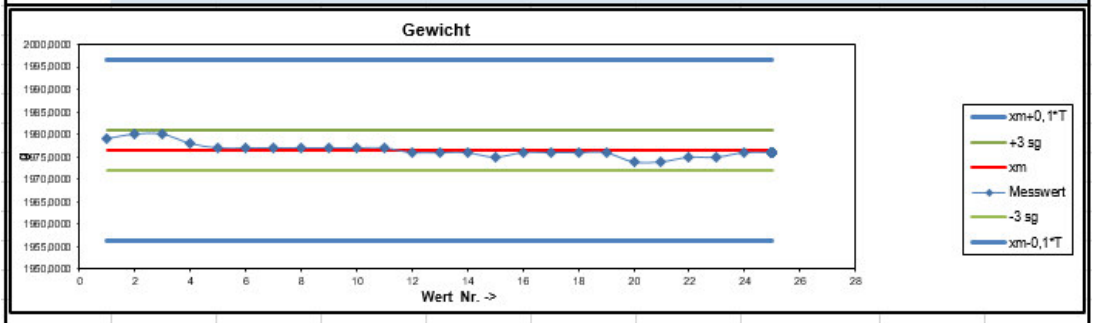
14.1 MSA

Messsystemanalyse
Verfahren 1: Cg / Cgk

Akt. Dat.:	13.05.2016	Bearb.Name:	Christian Hohmann	Abt./Kst.:	Hochschule Karlsruhe	Prüfort:	Li U24

Prüfmittel		Normal		Merkmal			
Bezeichnung:	Bosche Wägezelle	Bezeichnung:	Gewicht	Bezeichnung:	Gewicht		
Nummer:	S40S-G3-0020	Nummer:		Nummer	1		
Auflösung:	10	Istwert:	1976,5200	Nennmaß:	1976,5200	OSG:	2076,5200
Prüfgrnd.:	Abnahme	Einheit:	g	Einheit:	g	USG:	1876,5200
		U Kal					

Bemerkung:



Einzelwerte

1 - 5	6 - 10	11 - 15	16 - 20	21 - 25	26 - 30	31 - 35	36 - 40	41 - 45	46 - 50
1979,0000	1977,0000	1977,0000	1976,0000	1974,0000					
1980,0000	1977,0000	1976,0000	1976,0000	1975,0000					
1980,0000	1977,0000	1976,0000	1976,0000	1975,0000					
1978,0000	1977,0000	1976,0000	1976,0000	1976,0000					
1977,0000	1977,0000	1975,0000	1974,0000	1976,0000					

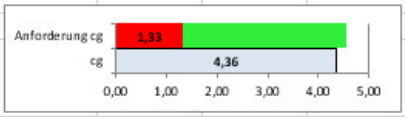
Spezifikationswerte		Gemessene Werte		Statistische Werte		Syst. Messabweichung		
x_m	1976,5200			$\bar{x}_g$	1976,5200	BIAS	0,0000	
x_m - 0,1*T	1956,5200	x_{min}	1974,0000	$\bar{x}_g - 3 * s_g$	1971,9276	t	0,0000	
x_m + 0,1*T	1996,5200	x_{max}	1980,0000	$\bar{x}_g + 3 * s_g$	1981,1124			
0,2 * T	40,0000	R	6,0000	$6 * s_g$	9,1848			
T	200,0000	n_{ges}	25	s_g	1,5308			

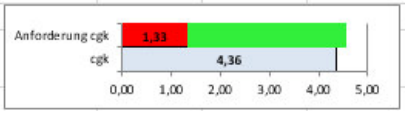
Mindestforderung an die Prüfmittelkennzahl		
	Cg ≥	1,33
	Cgk ≥	1,33

$$C_g = \frac{0,2 * T}{6 * s_g} = 4,36$$

$$C_{gk} = \frac{0,1*T - |\bar{x}_g - x_m|}{3 * s_g} = 4,36$$

Auflösung in % von T = 5,00%





Messsystem fähig für T bis

$T_{min/Cg}$ =	61,0787	g
$T_{min/Cgk}$ =	61,0787	g
$T_{max\ Aufl.}$ =	200,0000	g

Abbildung 14.1 MSA

14.2 Software

14.2.1 Temperaturmessung

Matlab Programm: Starten der Simulation

```
for k=0:1:47
    % Start Simulation
    filename='Thesis_SerDatTransm_Waegezelle.slx';
    simOut = sim(filename);

    % Save Data in different sheets
    my_cell = sprintf('A%d',k);
    xlswrite('Gewicht.xlsx',Gewicht,my_cell);
end
```

Matlab Programm: Excel Export

```
% Select file
filename='file.xlsx';
dataSheet=1;
timeSheet=2;
range='A1:H968850';

% Select y Axis
yAxis = xlsread(filename,dataSheet,range);
ylabel('Messwerte')

% Select x Axis
xAxis=xlsread(filename,timeSheet,range);
xlabel('Uhrzeit')

% Plot
h = plot(xAxis,yAxis);
datetick('x','HH:MM')
```

Arduino-Programm

```
#include <Onewire.h>

// OneWire DS18S20, DS18B20, DS1822 Temperature Example
//
// http://www.pjrc.com/teensy/td_libs_OneWire.html
//
// The DallasTemperature library can do all this work for you!
// http://milesburton.com/Dallas_Temperature_Control_Library

OneWire  ds(10);  // on pin 10 (a 4.7K resistor is necessary)
int x =0;
int row = 0;
float celsius;
void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
```

```
byte addr[8];

if ( !ds.search(addr)) {
  ds.reset_search();
  delay(250);
  return;
}

for( i = 0; i < 8; i++) {
}

if (OneWire::crc8(addr, 7) != addr[7]) {
    return;
}

// the first ROM byte indicates which chip
switch (addr[0]) {
  case 0x10:
    type_s = 1;
    break;
  case 0x28:
    type_s = 0;
    break;
  case 0x22:
    type_s = 0;
    break;
  default:
    return;
}

ds.reset();
ds.select(addr);
ds.write(0x44, 1);// start conversion, with parasite power on at the end

delay(500);     // maybe 750ms is enough, maybe not
// we might do a ds.depower() here, but the reset will take care of it.

present = ds.reset();
ds.select(addr);
ds.write(0xBE);         // Read Scratchpad
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
  data[i] = ds.read();
}

// Convert the data to actual temperature
// because the result is a 16 bit signed integer, it should
// be stored to an "int16_t" type, which is always 16 bits
// even when compiled on a 32 bit processor.
int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
  raw = raw << 3; // 9 bit resolution default
  if (data[7] == 0x10) {
    // "count remain" gives full 12 bit resolution
    raw = (raw & 0xFFF0) + 12 - data[6];
  }
} else {
  byte cfg = (data[4] & 0x60);
  // at lower res, the low bits are undefined, so let's zero them
  if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
  else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
  else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
  //// default is 12 bit resolution, 750 ms conversion time
}
celsius = (float)raw / 16.0;
```

```
  Serial.println(celsius);
}
```

14.2.2 C-Programm

Main.c

```
/*-----------------------------------------------------------------------------

    Programmiert von: Christian Hohmann
    Matrikelnummer:     50799
    Studiengang:        Mechatronik und Mikromechatronik System / Master
                        / EU4M. Hochschule Karlsruhe und EPI Gijon

    Datum:                      16.06.2016

    Hauptapplikation: Auftriebsmessung von Luftschrauben fr die Masterarbeit
                      mit dem Thema: "Steuerung von BLDC Motoren ber mobile
                      Applikation"
        - In der Main Routine wird zunaechst die On-Chip-Peripherie der F580-
          MCU Initialisiert
        - Anschlie nd wird die Wägezelle tariert
        - Sobald über UART ein "a" empfangen wurde, wird die Endlosschleife
          ausgeloest

-----------------------------------------------------------------------------*/

/*-----------------------------------------------------------------------------
    Includes
-----------------------------------------------------------------------------*/
#include "compiler_defs.h"
#include "C8051F580_defs.h"                 // System header file - SFR
(variable,macro-)definitions for the C8051F58x family
#include "stdio.h"

/*-----------------------------------------------------------------------------
    Funktionsdeklarationen
-----------------------------------------------------------------------------*/
extern Init_Device(void);                 // Initialisierung der Peripherie

extern unsigned long ReadAD(void);       // Funktion fuer das Auslesen des LCA
                                         // und der Wägezellenmessdaten

void Timer0_1_Init(void);                   // Timer 0 und 1 Initialisierung

void PCA1_1_Init(void);                     // PCA1 Initialisierung

void ItoA( int z, char* Buffer );         // Konvertierung Integer to ASCII

char getKey(void);    // Funktion fr das Abrufen eines Characters aus dem UART

/*-----------------------------------------------------------------------------
    Variablendeklaration
-----------------------------------------------------------------------------*/
// Messwerte Gewicht
unsigned long Messwert=0;
unsigned long Tarawert=0;
float k = 0.00453;                // k-(Multiplikations-)Gewichtsfaktor
int  gewichtG;

// Counter
int CounterM = 0;                 // Zaehler fuer die Inkrementierung
                                  // der ISR fuer Timer 0
int counterP = 0;                 // Zaehler fuer die Inkrementierung
                                  // der ISR fuer Timer 1
int counterR_motor1 = 0;
// int counterR_motor2 = 0;
```

```
// App Steuern
char SerialTrigger;                 // Variable fr die Überwachung des UART
                                    // Inputs (Eingabe ber Terminal oder App)

// Inkrementierungsvariablen
unsigned char i;
int d;

// Buffer
char buffer [5];    // Buffer zum Übertragen der Wägezellen Messdaten als Char

// Berechnung Drehzahl
//Motor 1
int RPM1;
static unsigned int first_capture_motor1, second_capture_motor1;
int capture_periode_motor1;

//Motor 2
int RPM2;
static unsigned int first_capture_motor2, second_capture_motor2;
int capture_periode_motor2;

/*-----------------------------------------------------------------------------
    START OF FILE
-----------------------------------------------------------------------------*/

/*-----------------------------------------------------------------------------
    Funktion fuer die Generierung des PWM Signals fuer Stopp
-----------------------------------------------------------------------------*/
void PWM_Signal_Stopp(void){
    PCA0CPL2 = 0x44;            // Uebertragung von 1,0 ms PWM
    PCA0CPL1 = 0x44;

    PCA0CPH2 = 0xF4;            // Uebertragung von 1,0 ms PWM
    PCA0CPH1 = 0xF4;

    counterP = 0;               // Reset counterP
    CounterM = 0;               // Reset CounterM

    TR0 = 0;                    // Timer Off
    ET0 = 0;                    // Timer Interrupt Off

    TR1 = 0;                    // Timer Off
    ET1 = 0;                    // Timer Interrupt Off

    TR4 = 0;
    CCF6 = 0;
}

/*-----------------------------------------------------------------------------
    Main Program

    - Abfrage ob Durch die App GESTARTET und ein 'a' bertragen wurde. Wenn ja, dann
      Messung starten und PWM bertragen
    - Abfrage ob durch die App GESTOPPT wurde. Wenn ja, dann Timer OFF und ISR verlassen
    - Nach jedem Ueberlauf (alle 11,1ms) wird die ISR des Timers 0 aufgerufen fuer die
      Messwertaufnahme und Timer 1 fuer die Generierung der PWM.
-----------------------------------------------------------------------------*/
void main (void){
    Init_Device();                  // Initialiserung der Peripherie
    EA = 1;                          // Enable fr alle Interrupts

    // Tarieren der Waegezelle
    for(i=0;i<10;i++){
```

```
        Messwert = Messwert + ReadAD();      // Mittelwert aus 10 Messungen
        Tarawert = (Messwert/10);            // Tarawert fuer Offset-Kompensation
    }

    // Endlosschleife
    while(1){
        SerialTrigger = getKey();       // Abfrage des UART Inputs ob das benötigte
                                        // Character empfangen wurde

        if(SerialTrigger == 'a'){
            Timer0_1_Init();             // Initialisierung des Timers 0 und 1 und Start
            PCA1 Init();                // Drehzahlmessung starten
        }

        if(SerialTrigger != 'a'){          // Wenn NICHT 'a' empfangen wird, dann STOPP
            PWM_Signal_Stopp();
        }
    }
}
/*-------------------------------------------------------------------------------------
    END OF MAIN
-------------------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------------------
    ISR fuer die Generierung des PWM Signals fuer Start
-------------------------------------------------------------------------------------*/
INTERRUPT (TIMER1_ISR, INTERRUPT_TIMER1){
    TH1 = 0xD4;
    TL1 = 0x98;

    counterP++;

    if(counterP <= 450){
        // Uebertragung von 1,2 ms PWM
        // fuer 450 Interrupts (11,1ms * 450 = 4,995 Sekunden)
        PCA0CPL2 = 0xEB;
        PCA0CPL1 = 0xEB;

        PCA0CPH2 = 0xF1;
        PCA0CPH1 = 0xF1;

    }

    if((counterP >= 451) && (counterP <= 901)){
        PCA0CPL2 = 0x44;              // Uebertragung von 1,0 ms PWM
        PCA0CPL1 = 0x44;

        PCA0CPH2 = 0xF4;         // fuer 5 Sekunden
        PCA0CPH1 = 0xF4;

        // Testen von verschiedenen PWM
        // Hier PWM 1,3
//      PCA0CPL2 = 0xBF;
//      PCA0CPL1 = 0xBF;

//      PCA0CPH2 = 0xF0;
//      PCA0CPH1 = 0xF0;

    }
```

```
// Reset der ISR
    if(counterP == 902){
        PCA0CPL2 = 0x44;                // Uebertragung von 1,0 ms PWM Low-Byte
        PCA0CPL1 = 0x44;

        PCA0CPH2 = 0xF4;                // Uebertragung von 1,0 ms PWM High-Byte
        PCA0CPH1 = 0xF4;

        counterP = 0;                                   // Reset counterP
        CounterM = 0;

        TR1 = 0;                                        // Timer Off
        ET1 = 0;                                        // Timer Interrupt Off

        TR0 = 0;                                        // Timer Off
        ET0 = 0;                                        // Timer Interrupt Off
    }
}

/*-------------------------------------------------------------------------------------
    Funktion fuer die Messwertaufnahme und Uebertragung der Messdaten
-------------------------------------------------------------------------------------*/
void MesswertAufnahme (void){
    //Aufnahme des Rohwertes aus dem AD-Wandler und Berechnung des Gewichtes
    Messwert = ReadAD();

    if(Messwert < Tarawert){
        gewichtG = ((Tarawert-Messwert)*k);
    }
    else{
        gewichtG = (Messwert-Tarawert)*k;
    }

    // UART-Uebertragung fuer die Schubkraft
    ItoA(gewichtG, buffer);      // Umrechnung Integer => ASCII Zeichen
                                 // fr die Anzeige in der App

    SBUF0 = '#';                 // Anfangs-Zeichen des Strings zur Erkennung in der App
    while (TI0==0);
    TI0 = 0;
    while (THRE0==0);

    for(d=0; d<5; d++){           // For-Schleife fr die ertragung der
                                  // im Buffer enthaltenen Zeichen (4 Zeichen)
    SBUF0 = buffer[d];
    while (TI0==0);
    TI0 = 0;
    while (THRE0==0);
    }

    SBUF0 = '~';            // Schluss-Zeichen des Strings zur Erkennung in der App
    while (TI0==0);
    TI0 = 0;
    while (THRE0==0);
}

/*-------------------------------------------------------------------------------------
    Interrupt Service Routine des Timers 0 fuer die Schubkraftmessung
-------------------------------------------------------------------------------------*/
INTERRUPT (TIMER0_ISR, INTERRUPT_TIMER0){
    TH0 = 0xD4;                   // Zuruecksetzen des Timers
    TL0 = 0x98;                   // nach jedem Timerueberlauf
                                  // Inkrementierung des Zaehlers

    if(CounterM % 9 == 0){        // Ausfuehrung der Routine Messwertaufnahme alle
                                  // 200ms fuer die Uebertragung an die App

        MesswertAufnahme();              // fuer 45 Interrupts (11,1ms * 9 = 99,9 ms)
    }

    if(CounterM == 912){
        TR0 = 0;                                // Timer Off
```

```
        ET0 = 0;                                    // Timer Interrupt Off
    }
}


/*------------------------------------------------------------------------------------
    ISR fuer das Capture von Motor 1 am Pin mit dem PCA1
------------------------------------------------------------------------------------*/
INTERRUPT(PCA1_ISR, INTERRUPT_PCA1){
    if(CCF6){           // Wenn Modul 6 ein Interrupt ausloest
        CCF6 = 0;      // Interruptflag muss softwareseitig auf 0 (Reset) gesetzt werden

        TH0 = 0xD3;          // Zuruecksetzen des Timers
        TL0 = 0xD0;         // nach jedem Timerueberlauf

        counterR ++;

        // Speichere den letzten Capture-Wert im Capture Register der Module 6 und 7
        Akt_Capture_Wert1 = PCA1CP6;
        Akt_Capture_Wert2 = PCA1CP7;

        // Berechne den Capture-Wert aus zwei vorherigen Werten
        Capture_Periode1 = Akt_Capture_Wert1 - Vorh_Capture_Wert1;
        Capture_Periode2 = Akt_Capture_Wert2 - Vorh_Capture_Wert2;

        // Aktualisiere den vorherigen Wert mit dem aktuellsten Wert
        Vorh_Capture_Wert1 = Akt_Capture_Wert1;
        Vorh_Capture_Wert2 = Akt_Capture_Wert2;

        if (counterR % 4 == 0){
            //RPM1-Wert berechnen
            RPM1 = 60000000/(4 * Capture_Periode1);     // RPM Motor1

            //RPM2-Wert berechnen
            RPM2 = 60000000/(4 * Capture_Periode2);     // RPM Motor2
        }
    }
}
/*------------------------------------------------------------------------------------
    END OF FILE
------------------------------------------------------------------------------------*/
```

14.2.3 Android-App

Quellcode

Da der Quellcode zu Umfangreich ist, kann dieser der beigelegten CD-ROM entnommen werden.

Flussdiagramm

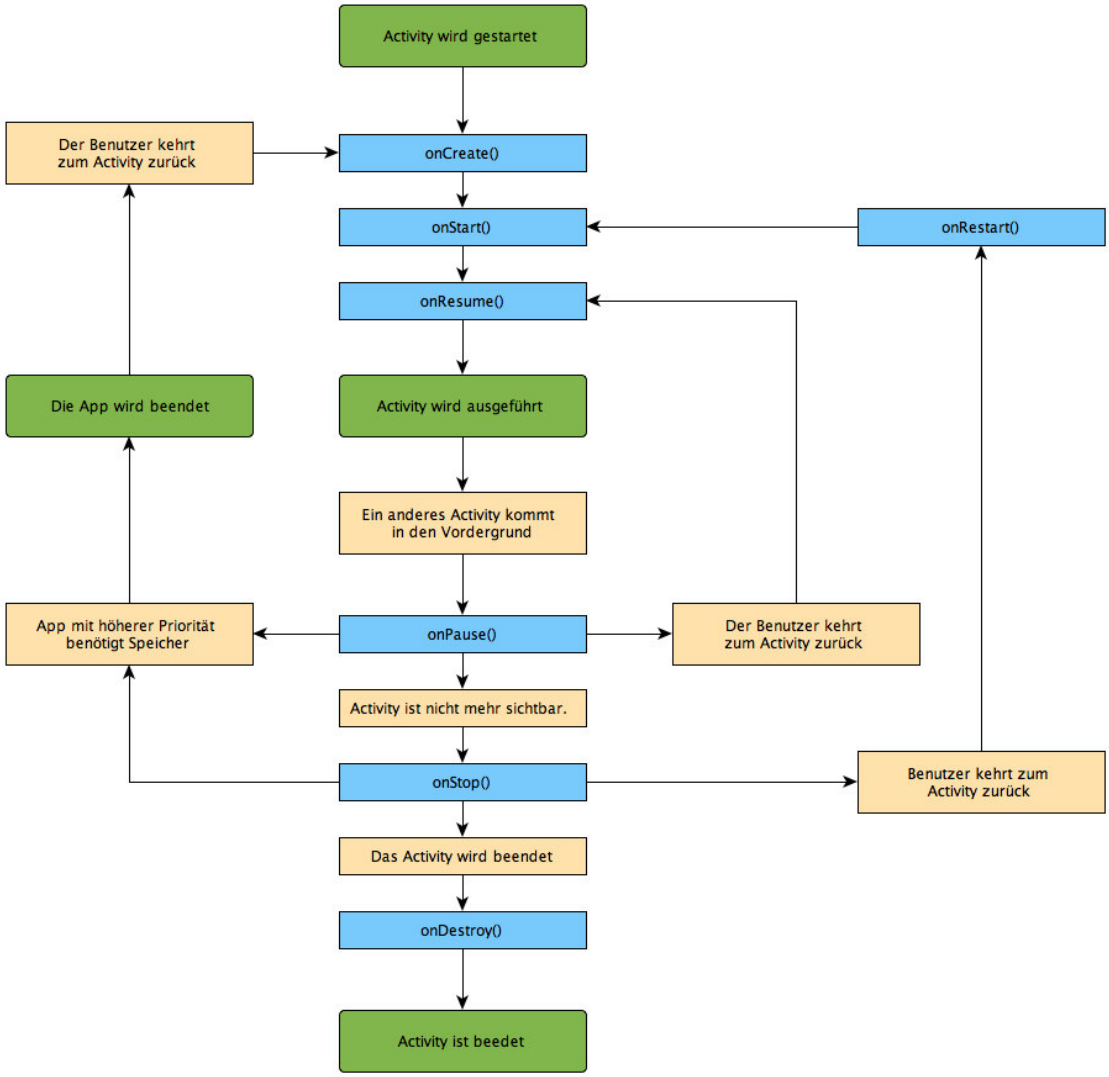


Abbildung 14.2 Lebenszyklus einer Activity [22]