electronics

MDPI

Article

Asterisk as a Tool to Aid in Learning to Program

Pelayo Nuño [1,*], Francisco G. Bulnes [1], Set Pérez-González [2] and Juan C. Granda [1]

[1] Department of Computing, University of Oviedo, Campus de Viesques, 33204 Gijón, Spain
[2] Department of Mathematics, University of Oviedo, Campus de Viesques, 33204 Gijón, Spain
* Correspondence: nunopelayo@uniovi.es

Abstract: Programming is a key subject in many engineering programs. Students often perceive it as a difficult skill to master. There is extensive literature on helping students learn and improve to program, most of which focuses on K-12 education. However, due to the current demand for workers with programming skills, more research must be conducted on techniques for learning programming at the higher education level. In this work, an analysis and evaluation of the usefulness of an Asterisk Private Branch Exchange (PBX) as an educational tool to improve the programming skills of students in higher education is presented. The study worked with undergraduate students in telecommunications engineering, with little work experience in programming, during the completion of their final year project. Results suggest that using Asterisk has a positive impact on the students' perception of their programming knowledge and skills, as well as an increment in the interest and comfort regarding programming.

Keywords: Asterisk; learning programming; VoIP; coding skills; education

Citation: Nuño, P.; Bulnes, F.G.; Pérez-González, S.; Granda, J.C. Asterisk as a Tool to Aid in Learning to Program. *Electronics* **2023**, *12*, 1160. https://doi.org/10.3390/electronics12051160

Academic Editors: Gabriela Gabajová and Martin Krajcovic

Received: 13 January 2023
Revised: 16 February 2023
Accepted: 24 February 2023
Published: 27 February 2023

1. Introduction

Programming is a mandatory subject in computing and other engineering programs. Students usually perceive programming as a daunting task [1]. Regardless of whether the student is using a high- or low-level programming language, an important problem when writing code tends to be related to a lack of knowledge about the syntax and semantics of the language. Thus, programming is prone to syntactic and semantic errors, becoming difficult for students, particularly novices [2]. Learning programming is even more difficult when students are not fluent English speakers, since the vast majority of programming languages use English for reserved words. Variables, objects, control flow, and conditional statements are all expressed with English words. The considerable cognitive effort required to learn programming may lead to decreased motivation and satisfaction and ultimately a negative attitude towards programming [1].

The majority of studies about making the process of learning to program more effective are focused on K-12 education, where there is huge effort to incorporate computing and other related aspects, such as computational thinking, into the curriculum [3]. Due to the demand for workers with programming skills in technological workplaces nowadays, more research is needed not only in K-12 education but also at higher academic levels. Particularly important are students in computer science and other engineering programs since programming is a fundamental skill in these disciplines. Many undergraduate students in these disciplines have difficulty when learning to program in core courses. They struggle to complete introductory programming courses to such an extent that one-third of students in these courses either fail or drop out [4]. Clearly, this is of significant concern to the teaching staff of computer science related programs [5].

Some studies have attempted to improve the learning process in introductory programming courses through new teaching techniques and methodologies, such as the flipped classroom [6]. However, students often lack confidence in their proficiency in programming after their first or second year of studies. Working on students self-efficacy leads to better

performance, as research works show that there is a significant correlation between students' programming self-efficacy and their perceived value of learning to program [4]. In [4], two main approaches to improving self-efficacy are described: embedding interventions to increase students' perceived value of learning programming and carrying out activities related to real-life projects.

There are several works on learning methods and how activities can emulate a real environment using educational tools. Tangible systems, such as robots and other hardware platforms, can provide a physical environment where students can manipulate real objects to solve tasks [7]. Tangible systems increase their motivation, leading to easier knowledge acquisition and retention [1]. In a programming context, tangible systems allow students to work in a realistic environment while programming, receiving immediate feedback on their development. In addition, the use of tangible systems helps reduce the abstract nature of problems, so students perceive programming as an interesting and challenging activity instead of as a daunting task.

The aim of this paper is to analyse whether the use of a real Voice over IP (VoIP) environment based on an Asterisk Private Branch Exchange (PBX) interacting with different physical systems and services improves the programming skills of students in higher education. In addition to programming the Asterisk dial plan using a particular scripting language, students must also develop pieces of code in other languages, such as Bash, C# or Python, to implement more advanced functionalities that are executed from the PBX through user interaction. Within the context of this work, the Asterisk PBX plays the role of a user interface in such a way that it receives voice orders from the users through a VoIP phone to execute specific actions through scripts written in various programming languages. Therefore, all interactions between users and the Asterisk PBX are via multimedia technologies, which are attractive and well-valued by the students [8].

This work serves as a pretest to consider the suitability of extending the use of Asterisk to initial degree courses in order to improve students' programming skills. In order to evaluate this possibility, we have taken senior students, whose level of maturity and knowledge make their opinions and perceptions reliable, as the subject of study. Specifically, the study is focused on 12 undergraduate students in telecommunications engineering. All of them had a basic knowledge about programming, having taken several programming subjects in which they worked superficially with various programming languages. Therefore, the students had not mastered any programming language in depth, nor had they previously faced the development of large or medium-sized projects. The study was carried out while they completed their final year project, for which they must demonstrate the maximum level of autonomy in their studies.

As the final project is the last step for a telecommunications engineering student, and no written test is taken after its completion to assess the level of learning acquired in a quantitative way, the estimation of the benefits of using the Asterisk framework was conducted by means of a qualitative study based on a pretest and a post-test survey. This paper examines three hypotheses related to the use of Asterisk in the final year project. Results suggest that using Asterisk has a positive impact on the students' programming skills.

The remainder of the paper is organized as follows. Work related to using educational tools regarding programming in higher education is presented in Section 2. Section 3 briefly describes the architecture of the Asterisk framework and the configuration of a dial plan. The final year projects involving Asterisk completed by the 12 students during this research are explained in Section 4. The methodology is presented in Section 5, and the results are discussed in Section 6. Finally, Section 7 contains the concluding remarks and outlines future work.

2. Related Work

The use of technology integration, such as mobile devices, electronic hardware platforms, and telematic applications, is an innovative approach to learning programming. This approach increases the desire to learn to program among university students [9]. Robotics,

hardware platforms, and serious games are the main educational tools used for learning programming [10] and developing computational thinking [11] in higher education.

In [12], a wide review of the use of educational robotics in post-secondary education was carried out. The study presents several potential benefits of using robotics as an educational tool, especially in computer science and undergraduate engineering courses. The authors state that the use of robotics is effective as a complementary learning tool in higher education. Another study based on the use of robotics together with artificial intelligence (AI) at the university level was presented in [13]. The results show positive attitudes of both lecturers and students about the combination of these techniques as part of the development of teaching practice in higher education. A similar study can be found in [14], where LEGO Robotics was used to foster the learning of programming and AI knowledge in first year undergrads, with excellent results. There are also interesting studies that show that robotics is an effective way to improve programming skills in primary and secondary education [15].

Arduino is a widely used framework in higher education, among other environments, to improve programming skills and develop computational thinking [16]. For example, Ref. [17] studied the use of Arduino during a programming course in which students of the computer education and instructional technologies department took part. Several interesting findings were obtained, including the reinforcement of the topics learned in programming. Another work, developed in [18], involving university science and engineering programs shows that when Arduino is used, more students successfully learn to program and enjoy programming more. The positive effects of Arduino on the computational thinking skills of preservice teachers were studied in [19]. The results show a significant difference in terms of creativity, algorithmic thinking, critical thinking, and overall scoring. Other Arduino-related works focused on enhancing programming skills within the scope of higher education can be found in [20–22]. In addition, Arduino has been used in primary and secondary education to foster the development of programming [23] and robotics skills [24].

Finally, a widespread approach in education that encourages learning is the use of serious games [25,26], which have proven to enhance the learning of programming in higher education. For example, in [27], the use of serious games leads to a positive attitude and increased motivation. Similarly, an activity based on serious games for teaching logic and programming to students in computer engineering at university is presented in [28]. The use of games developed in Scratch to teach programming to first year engineering students is analysed in [29]. Web-based games have also been used to teach robotics and computer programming in [30], increasing both interest and understanding in these topics. In addition, serious games have been used to teach cybersecurity in higher education [31], helping students understand cryptographic algorithms and learn safe and suitable behaviour when connected to the Internet. Other interesting publications describe how LEGO Serious Play is used to teach software engineering to computer science students [32], how a gamification proposal based on escape rooms improves Python skills [33], how serious games affect the programming skills and fluency of university students [34], and the positive impact of serious games combined with flipped classroom teaching on coding skills [35].

The use of serious games is usually a purely software approach, as is the Asterisk framework. To the best of the authors' knowledge, this is the first study on the impact of Asterisk to improve the programming skills of students in higher education.

3. Background: The Asterisk Framework

Asterisk is an open-source framework that enables a regular computer to behave as a Back-to-Back User Agent (B2BUA) in a VoIP call, operating between the participants both in the signaling and data planes [36]. Asterisk configuration relies on multiple configuration files. Each signaling protocol is related to one or several files, depending on the version of Asterisk. Some configuration files of signaling protocols, e.g., *pjsip.conf* for the Session

Initiation Protocol (SIP), define the accounts that can log into the PBX by using such signaling protocols. Authentication parameters, such as usernames, passwords, supported codecs and transport protocols, can also be defined.

Asterisk functionality is programmed by means of a special configuration file (*extensions.conf*) that defines the dial plan. This configuration file is the core of the PBX, where actions and services that are available for an account are specified declaratively by means of a particular scripting language. The dial plan is organised into *contexts*, which are independent sections that correspond to the minimum organisational unit of the dial plan. The purpose of using contexts is to separate and cluster functionalities within the dial plan, for example to provide a different set of services according to each end point. A context can be composed of one or more *extensions* that can be dialed from the end points. An extension is a set of related instructions that are sequentially executed to reach the desired behaviour. Each instruction within an extension must be numbered and is called *priority*. In addition, each executed instruction performs an action in the dial plan. Within the scope of Asterisk, this action is called *application*. A piece of an example dial plan is shown in Figure 1.

```
[MyContext]
exten => 100,1,Answer()
;The call is answered
same => n,Record(en/message.wav)
;An audio from the caller is recorded and saved to disk
same => n,Wait(2)
;A two-second pause
same => n,Playback(message)
;A stored audio is loaded and played to the caller
same => n,Dial(PJSIP/Alice&PJSIP/Bob&PJSIP/Trudy)
;Alice, Bob and Trudy terminals are called simultaneously
same => n,AGI(./scripts/DeleteMessages.sh)
;An external script stored in the system is executed
same => n,Hangup()
;The call is hung up
```

Figure 1. Asterisk dial plan example.

As can be seen, the snippet shows a context, called `MyContext`, composed of a single extension, labelled `100`. When this extension is dialed, seven instructions (priorities) are executed step-by-step, invoking a different application in each of them. For example, the *Record* application saves an audio message in the Asterisk PBX, while *Playback* plays an audio stored in the PBX, such as a previously saved message.

An interesting aspect of the Asterisk framework is that it can be used to develop systems that go beyond the role of a PBX. Within the scope of the final year projects presented in this work, the Asterisk PBX plays the role of a user interface, in such a way that it receives user commands through a VoIP phone (software or hardware) and, in response, executes actions on other systems via scripts written in various programming languages. In this respect, the Asterisk Gateway Interface application (*AGI*) is the key component since it enables developing more complex solutions. In the example dial plan illustrated in Figure 1, the *AGI* application is used to execute a script that deletes all the audio messages recorded by users that were stored in the PBX. Similarly, it could also be used to send commands through the serial port to manipulate a TV or to send HTTP requests to activate/deactivate actuators in an simulated Internet of Things (IoT) environment.

4. Final Year Projects Based on Asterisk

During this five-year study, several final year projects based on Asterisk were developed. In this section, these projects are presented and classified according to their main

goal. It should be noted that the 12 students chose the topic of their final year project freely among those available.

4.1. Remote Control of a Television and Implementation of Smart TV Functionalities

Two final year projects were carried out using an Asterisk PBX to control a television remotely. The architecture of these projects is illustrated in Figure 2. The user interacts with the Asterisk PBX through a set of predefined voice commands with an Interactive Voice Response (IVR). These voice commands are sent from the PBX to the speech recognition service offered by Google to transcribe voice commands into a text response. The Asterisk PBX parses the response and according to the text command received executes a local script. This sends an order to an LG television (32LG5000 model) through a serial RS-232 port.

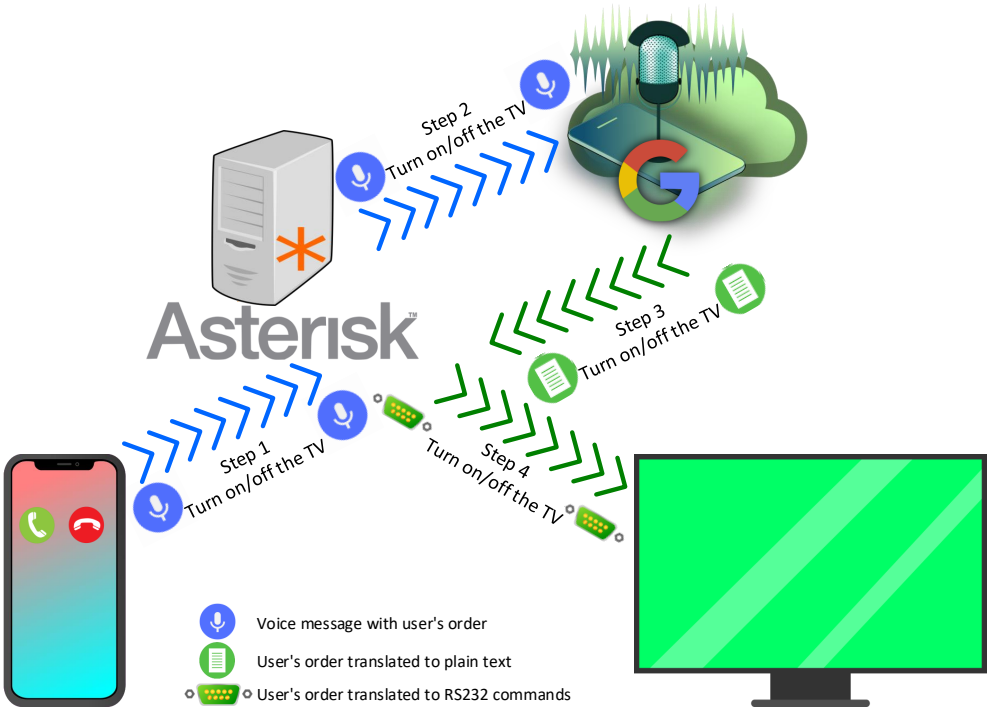

Figure 2. Architecture of the projects to manage a television.

The first project was focused exclusively on performing the basic actions for controlling a television remotely, such as turning it on and off, tuning in a channel, moving forwards and backwards through the list of available channels, and adjusting or presetting the volume or brightness. For example, to turn off the television, the user dials the extension which executes the IVR and says *Turn off the TV*. Asterisk then sends that message to the Google speech-to-text service and, after parsing the string received as a response, runs a script written in Bash to send the "`k a`" command to the television through the serial port. This command turns the TV off.

The second project implemented Smart TV functionalities in the LG TV, which lacks these features. Specifically, parental control measures were implemented. Thus, by interacting with Asterisk, a user may block certain channels or disable the operation of the remote control, either immediately or by scheduling a future date or time. In addition, the PBX periodically assesses the status of the TV. Then, Asterisk stores information about the active channel in a relational database. It also allows a user to access channel viewing statistics for different time ranges. In this second project, all the local scripts executed from the AGI application were written in Python.

4.2. Control of a Smart IoT Home

Two final year projects were proposed to use an Asterisk PBX to interact with a Smart IoT Home. The architecture of these projects is shown in Figure 3. The architecture is similar to the previous projects but when the response is received by the PBX and subsequently parsed, Asterisk executes a local script written in Bash that sends a Hypertext Transfer Protocol (HTTP) request to the Smart IoT Home controller in order to activate an actuator. For these projects, an OSOYOO Smart Home learning kit with Arduino MEGA2560 is used as a Smart IoT Home model.

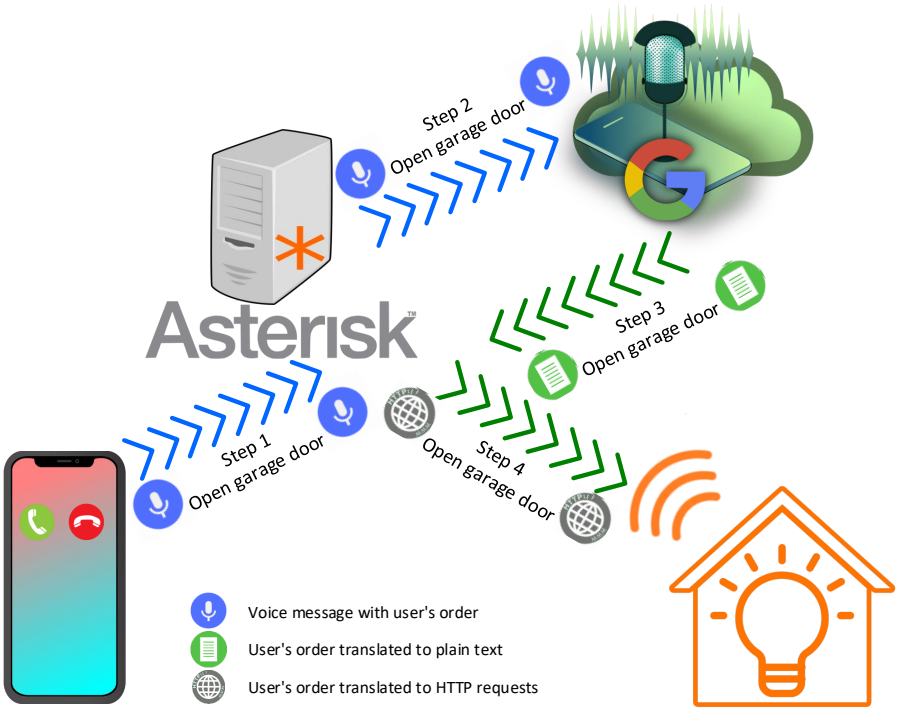

Figure 3. Architecture of the projects to manipulate an IoT Smart Home.

The first project was focused exclusively on activating and deactivating the actuators installed in the Smart IoT Home. The user could turn the lights, sprinklers, or heaters on and off, open and close the garage door, and write messages on a digital whiteboard. The second project also incorporates the management of the sensors installed in the Smart IoT Home. If it detects smoke inside the house, movement in a room, or temperature above or below predetermined thresholds, the Smart IoT Home calls a specific extension of the Asterisk PBX. After processing the call, the PBX calls the user to inform them of the situation in the IoT environment.

4.3. Increasing the Functionalities of an External Application

This final year project differs from previous ones in that rather than communicating with an IoT Smart Home prototype using the HTTP protocol, the interactions are with an external service using Hypertext Transfer Protocol Secure (HTTPS). In this project, the target application was Strava, a social network for tracking human exercise, frequently used by cyclists, swimmers, and runners using Global Positioning System (GPS) data. The purpose of the project was to develop new features, making it possible to extend the catalogue of functionalities that Strava offers by default.

Users can send voice commands to the IVR to request total, average, and maximum values of performance parameters of recorded activities such as distance covered, elevation gain, and calories consumed. The main improvement with respect to the functionalities available in Strava, even in the subscription version of the application, is that the time

range to perform the analysis was completely customizable and could cover several days, weeks, or months.

The Asterisk PBX collects user activities from the profile within the specified date range and parses the desired performances. As a response, the PBX returns a voice message indicating the total, average, or maximum value. To do so, the Asterisk PBX runs a local script written in Python that calls the Strava API [37], which provides developers with a simple interface to access the registered activities in the social network.

4.4. Deployment of a Presence Control System

Two final year projects used Asterisk to deploy a presence control system within a corporate network. The architecture of these projects is shown in Figure 4. In these projects, speech-to-text translation technology is not used since the user performs a desired action by simply dialing a specific extension in the dial plan. The system allows an employee to register working hours and breaks by dialing different extensions. The Asterisk PBX records such interactions in a relational database.

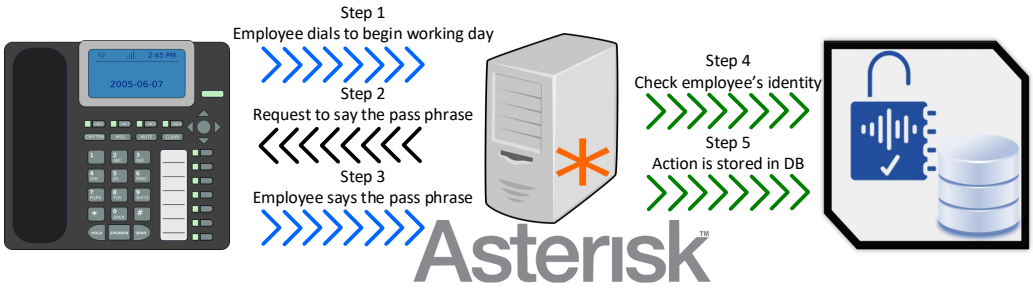

Figure 4. Architecture of the projects to deploy a presence control system.

In the initial version of the project, employees could register the beginning and end of their working day and breaks. They could also request a leave of absence for a specific date. The interactions between employees and Asterisk are carried out through different IVRs and well-known extensions for each purpose (one for the beginning of the working day, another for the end, etc.).

In another version of the project, a low-cost solution to improve the reliability of the previously described system was presented. The idea behind this project was to avoid identity fraud when employees interact with the Asterisk PBX. As this is impossible to achieve by using only login and password, this project introduced a new entity in the architecture, specifically a biometric module [38], as shown in Figure 4.

In order to avoid fraudulent registrations, the PBX asks employees to say a pass phrase after dialing a specific extension. In this example, the extension 1000 could register the beginning of the working day, so that the system can identify them by voice. To do so, the Asterisk PBX communicates with the biometric module that processes the caller's voice to verify the identity and thus avoids fraud. The voice samples of all the employees of the corporate network are stored in a relational database used by the biometric module to perform queries. If the employee identification is successful, Asterisk stores the time of the call in a second relational database. Thus, it registers the beginning and end of working days and breaks. In these projects, Asterisk interacts with the database and the biometric module through scripts written in Bash.

4.5. Development of a Notification Management System

This final year project was focused on the development of a notification management system for employees within a corporate network through the use of their VoIP telephones. The architecture of this project is shown in Figure 5. The proposed system allows employees to schedule future reminders and notices interactively, as well as to manage them once they have been registered in the system.

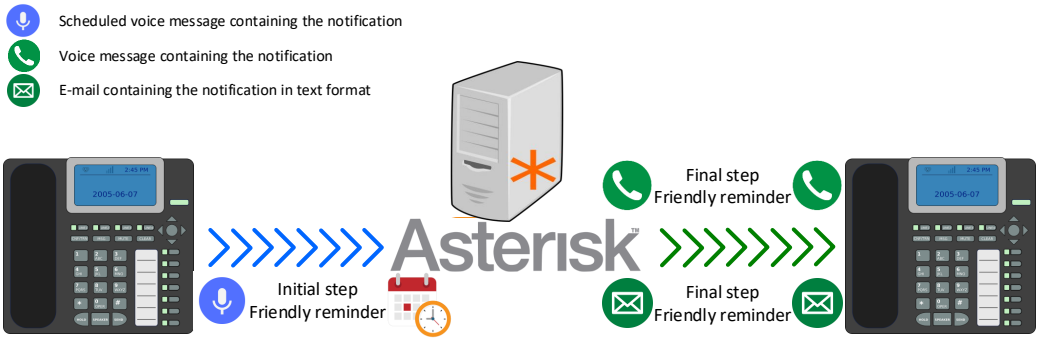

Figure 5. Architecture of the project to develop a notification management system.

Notices are delivered from the Asterisk PBX to the recipient by means of a conventional call so employees can listen to the message that was planned for them. Notices can also be sent by email with the written transcript of the content of the notification. The functionalities of this project are implemented in scripts written in Bash. Although the transcript of the notification from audio to text is performed using the Google speech recognition service, this entity has not been included in the figure for the sake of clarity.

Within the scope of this project, a special feature available in Asterisk called *call files* is used. They are files with a special structure that can be used to program the Asterisk PBX to manage the sending of notifications on a specific date and time.

4.6. Development of a COVID-19 Tracker

Two final year projects were proposed to develop a system based on Asterisk enabling a user to register interactions with their contacts and to act in case of a positive COVID-19 test. The architecture of these projects is illustrated in Figure 6. Again, although the user and the Asterisk PBX communicate by voice, which is transcribed by the Google speech recognition service, this entity is omitted from the figure for the sake of clarity.

The user dials an extension in the Asterisk dial plan in order to update a relational database which stores the names and phone numbers of their contacts. By dialing another extension, the user can register all interactions with those contacts in the database. Thus, a log of interactions is created, which can be tracked in the case of a positive contact. Similarly, contacts can be notified in the case of a positive from the user themself. These functionalities are programmed using scripts written in Python.

If the user tests positive for COVID-19, a dedicated extension can be dialed. Then, the Asterisk PBX queries the contact database and retrieves all the user's interactions in the last two weeks. Each affected contact receives a message notifying about the positive test of the user so appropriate measures can be taken. This message is sent from Asterisk to a contact by the Telegram messaging app. Finally, if a contact tests positive, the user can dial another extension in the dial plan to check for interactions within the last two weeks.

Figure 6. Architecture of the project to develop a COVID-19 tracker.

4.7. Securing and Hardening an Asterisk PBX

Another proposed final year project was related to the development of a platform to perform several on-demand assessments and self-reconfigurations in the Asterisk PBX to improve overall security. The platform performs reactive actions to reconfigure and harden the Asterisk PBX. Some of the securing actions include: the platform reacts to account scans and malicious login attempts by updating access control rules; the platform checks the correctness of the programmed dial plan and automatically fixes source code lines that compromise the security of the PBX; and the platform automatically replaces deprecated and compromised methods and functions. The platform can be easily configured to include new vulnerabilities to evaluate in the assessment process in order to avoid premature obsolescence. More details regarding this work, implemented in C#, can be found in [39].

5. Methodology

The research proposed in this work investigates the impact of using Asterisk as an educational tool to improve the programming skills of students in higher education during their final year project. All the projects used the Asterisk framework, along with other hardware devices and services, to simulate realistic scenarios. This section describes the hypotheses, methods, questions, and assertions that were posed to students.

5.1. Hypotheses of the Study

Three main hypotheses are made in this study:

Hypothesis 1 (H1). *Students with a higher level of Asterisk prefer its own language to other ones.*

Hypothesis 2 (H2). *Programming in Asterisk positively impacts programming knowledge and skills.*

Hypothesis 3 (H3). *The previous level of coding skills influences comfort level when programming after completing the final year project.*

5.2. Method

This research covered five years between 2018 and 2022. Figure 7 shows the number of final degree projects presented in each academic course of the time span of this study. Due to the size of the sample and the limited availability of students per academic course, a control group was not feasible, so this research used a single pretest and post-test group.

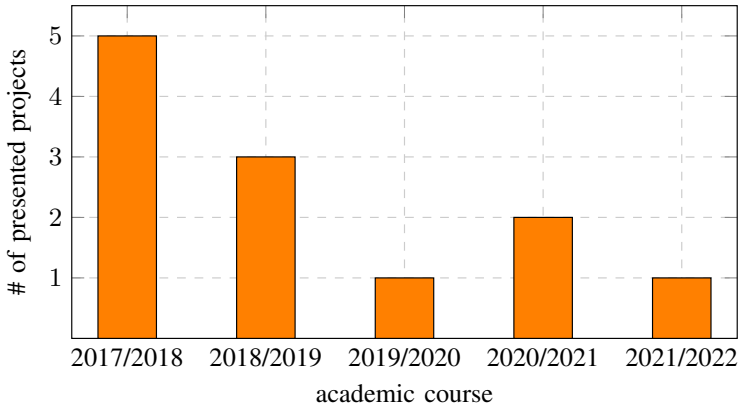

Figure 7. Number of projects presented in each academic course.

The procedure followed for each student consisted of an initial pretest just before the beginning of the final year project, the development of the project, which is designed in the curriculum in such a way that it can be completed by the student in 300 h, and a post-test after completing the project. For both the pretest and the post-test, the students completed surveys in which several questions and assertions were posed to evaluate various aspects. Both tests were anonymous and implemented using Microsoft Forms. They were answered on a Likert scale, and neither included open-ended questions.

In the pretest, students were asked about demographic information such as sex and age, as well as the specialisation of the telecommunications engineering program they were studying. Several questions were asked regarding their knowledge of Asterisk, their self-perceived level of programming skills, and programming languages previously used. Part of the survey used for the pretest is shown in Table 1. It should be noted that Q3 allowed for the selection of multiple languages.

Table 1. Questions in the pretest survey.

Question	Description
Q1	What do you consider to be your level of knowledge about Asterisk?
Q2	How would you evaluate your programming skills?
Q3	How many programming languages do you know?

In the post-test, several assertions were presented to the students in order to determine the cognitive and attitudinal impact of using Asterisk on their programming skills. The post-test included assertions to evaluate the students' subjective perception of their improvement in programming after the final year project and especially how much Asterisk had contributed to that. The survey used for the post-test is shown in Table 2.

Table 2. Assertions in the post-test survey.

Assert	Description
A1	I felt more comfortable programming in Asterisk than using other languages I already knew
A2	Programming in Asterisk was easier than using programming languages I already knew
A3	I think that programming in Asterisk has improved my programming skills
A4	I am more interested in programming
A5	I feel more comfortable when programming

5.3. Participants

This study involved telecommunications engineering undergraduate students from the University of Oviedo (Spain). A total of 12 students agreed to take part in this study while working on their final year projects. Students did not receive any incentive for participating in the research. The average age of the students was 23.67 ($sigma = 2.27$), 8 (66.67%) were female and 4 (33.33%) were male. None of the students participating in the study had been involved in the development of large or medium-sized programming projects in the past. The pretest revealed that the majority of the students (91%) considered they had enough knowledge about programming due to previously completed courses. Most students had prior experience with programming in C# (64%), Python (58%), and C++ (45%) programming languages.

5.4. Limitations

Due to the duration of the study and the availability of students completing their final project per year, this work did not have a control group to analyse the impact of the Asterisk framework on the programming skills of the students. The validation of the hypotheses could have been conducted more exhaustively with a higher number of students taking part in the study, so this research can be considered as a preliminary study.

The final year project is the last step for a student in the telecommunications engineering program. This may affect the students' perceptions of Asterisk, since the implicit achievement may entail a much more optimistic assessment than if the student was in previous stages of the program. Even so, due to their academic situation near the end of their studies, these students have reached a high level of maturity, so their opinions and perceptions can be considered reasonably reliable.

6. Results and Discussion

This section shows the analysis of the data provided by the students in the sample. To verify each of the hypotheses, one or several pretest questions and post-test assertions were used. The independence of the replies was analysed as well as the relationship between the answers provided in the questions and those provided in the assertions. Since all the questions in the questionnaires are categorical in nature, independence tests are used. All calculations were performed using JASP 0.16.1 [40], the latest version at the time of writing.

6.1. First Hypothesis

The first hypothesis was that those students with a higher level of Asterisk prior to their final year projects, prefer the Asterisk programming language to others. This hypothesis was tested with the data collected. For that purpose, pretest question Q1, shown in Table 1, was used:

- Q1: What do you consider to be your level of knowledge about Asterisk?

The responses to this question have been related to those provided by the A1 and A2 post-test assertions, shown in Table 2:

- A1: I felt more comfortable programming in Asterisk than using other languages I already knew.
- A2: Programming in Asterisk was easier than using programming languages I already knew.

The assumption made by the authors was that the higher the students' knowledge of Asterisk, the more comfortable they feel using this technology once their respective final year projects are completed. For this reason, those students who indicate that they had a higher previous level of Asterisk are more likely to respond that they feel more comfortable and/or find it easier to program in Asterisk. As stated at the beginning of this section, a test of independence was used to verify whether these variables are independent or whether there is a relationship between them.

The results of the first test performed show no evidence of dependence between Q1 and A1, with a p-value greater than 0.05; $\chi^2 = 3.429$, $p = 0.18$. Nor does the second test, between Q1 and A2, confirm the initial assumption; $\chi^2 = 3.429$, $p = 0.18$.

Based on the results obtained, we have no evidence to affirm that the previous level of Asterisk influences the students' perception of the comfort (A1) or ease of programming (A2) in Asterisk after the completion of their project. Therefore, the tests conducted do not validate the hypothesis that students with a higher level of Asterisk would prefer programming in the Asterisk scripting language rather than other programming languages. In any case, since the work presented in this article is a preliminary study based on a small sample, such a low p-value is not sufficient to rule out the existence of the proposed relationship. Thus, the study of this hypothesis should continue using a large sample.

6.2. Second Hypothesis

The second hypothesis proposed by the authors states that programming in Asterisk has a positive impact on the knowledge and programming skills of the students. In this case, the results of pretest questions Q1 and Q3 were analysed:

- Q1: What do you consider to be your level of knowledge about Asterisk?
- Q3: How many programming languages do you know?

The possible relationship between the answers provided for the A3 post-test assertion was studied:

- A3: I think that programming in Asterisk has improved my programming skills.

The authors' assumption prior to data collection was that those students with a greater knowledge of Asterisk and those who know more programming languages will benefit more from the work conducted with Asterisk during their final year projects, so the answers to the post-test questions should be more positive. For this purpose, their independence was analysed.

The tests carried out to determine the independence between Q1 and A3; $\chi^2 = 3.704$, $p = 0.295$; show no evidence of such dependence, obtaining very high p-values. However, the tests of independence between Q3 and A3; $\chi^2 = 14.22$, $p = 0.027$; do show evidence of dependence.

Based on the results obtained in these tests, it can be concluded that for those students who know more programming languages, programming in Asterisk has helped them to improve their programming skills, regardless of their previous knowledge in Asterisk technology. These results seem to give partial support to the hypothesis, suggesting that programming in Asterisk improves the programming skills of students based on their prior knowledge of more than one programming language.

6.3. Third Hypothesis

In the third hypothesis, it was stated that the higher the previous level of programming, the more the students' comfort in programming increases after completing the final year project. In this case, the results of pretest questions Q2 and Q3 were analysed:

- Q2: How would you evaluate your programming skills?
- Q3: How many programming languages do you know?

The possible relationship between the answers to these pretest questions and those provided for A5 and A6 post-test assertions was studied:

- A4: I am more interested in programming.
- A5: I feel more comfortable when programming.

In this case, it was expected that those students with stronger programming skills and who know a wider variety of languages will feel more comfortable and more interested in programming once their project are completed.

The first test performed does show evidence of dependence between Q2 and A4; $\chi^2 = 6.667$, $p = 0.036$. In addition, the second test, relating Q2 and A5; $\chi^2 = 9.333$, $p = 0.009$; does show evidence of dependence for these variables as well. However, the last two tests performed show no evidence of the dependence of Q3 and the A4 assertions; $\chi^2 = 6.9$, $p = 0.141$; and A5; $\chi^2 = 8.22$, $p = 0.084$.

Based on the results obtained in these tests, it can be concluded that for those students with a previous higher self-perceived level of programming skills, completing their final year project using Asterisk has increased their interest in programming. Furthermore, from these results it is also possible to assert that the use of Asterisk has increased their comfort when programming. Therefore, these results seem to give support to the third hypothesis.

7. Conclusions

An analysis and evaluation of the Asterisk PBX as an educational tool to improve the programming skills of higher education students has been presented. The study involved 12 undergraduate students of telecommunications engineering who had not mastered any programming language in depth, nor had they previously faced the development of large or medium-sized projects. The scope of this research was focused on the final year project. Several real VoIP environments based on an Asterisk PBX, interacting with physical systems and other services, were developed by the students.

While not all of the authors' hypotheses can be confirmed, it was observed that the completion of the final year project using Asterisk leads to beneficial consequences. In a significant percentage of the cases, it helped to improve the students' perception of their programming knowledge and coding skills. In addition, results also suggest that both the interest and comfort regarding programming were increased in the students.

Although this study is a preliminary test and it is based on a small sample of students, the results have been very satisfactory. Therefore, they give reasons to believe that Asterisk can indeed have a positive influence on the programming skills of higher education students. Based on these results, the use of the Asterisk framework could be considered as an interesting alternative to be used as an educational tool to improve programming skills, although more research is needed.

Future work will be focused on carrying out a more in-depth study, using a larger sample to confirm the results of this preliminary test. To do so, we are open to collaborating with lecturers from other universities in order to gather a sufficiently large sample in a reasonable amount of time. In addition, it could be interesting to study the use of Asterisk as an educational tool at a stage other than the final year project, for example in initial degree courses in engineering programs.

Author Contributions: Conceptualization, P.N. and F.G.B.; design, P.N., F.G.B., S.P.-G. and J.C.G.; research, P.N.; methodology, P.N., F.G.B., S.P.-G. and J.C.G.; resources, P.N., F.G.B., S.P.-G. and J.C.G.; data curation, F.G.B. and S.P.-G.; writing, P.N., F.G.B., S.P.-G. and J.C.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the project PID2021-124383OB-I00 of the Spanish National Plan for Research, Development and Innovation. Financial support given by the Asturian University Institute of Industrial Technology (IUTA) is also acknowledged.

Data Availability Statement: The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tsalmpouris, G.; Tsinarakis, G.; Gertsakis, N.; Chatzichristofis, S.A.; Doitsidis, L. Hydra: Introducing a low-cost framework for STEM education using open tools. *Electronics* **2021**, *10*, 3056. [CrossRef]
2. Jamil, M.G.; Isiaq, S.O. Teaching technology with technology: Approaches to bridging learning and teaching gaps in simulation-based programming education. *Int. J. Educ. Technol. High. Educ.* **2019**, *16*, 25. [CrossRef]
3. Lyon, J.A.; Magana, A.J. Computational thinking in higher education: A review of the literature. *Comput. Appl. Eng. Educ.* **2020**, *28*, 1174–1189. [CrossRef]
4. Abdunabi, R.; Hbaci, I.; Ku, H.Y. Towards enhancing programming self-efficacy perceptions among undergraduate information systems students. *J. Inf. Technol. Educ. Res.* **2019**, *18*, 185–206. [CrossRef] [PubMed]
5. Malik, S.I.; Coldwell-Neilson, J. A model for teaching an introductory programming course using ADRI. *Educ. Inf. Technol.* **2017**, *22*, 1089–1120. [CrossRef]
6. Nikolic, S.; Ros, M.; Hastie, D.B. Teaching programming in common first year engineering: Discipline insights applying a flipped learning problem-solving approach. *Australas. J. Eng. Educ.* **2018**, *23*, 3–14. [CrossRef]
7. Alessandrini, A. A study of students engaged in electronic circuit wiring in an undergraduate course. *J. Sci. Educ. Technol.* **2022**, *32*, 78–95. [CrossRef]
8. Abdulrahaman, M.D.; Faruk, N.; Oloyede, A.A.; Surajudeen-Bakinde, N.T.; Olawoyin, L.A.; Mejabi, O.V.; Imam-Fulani, Y.O.; Fahm, A.O.; Azeez, A.L. Multimedia tools in the teaching and learning processes: A systematic review. *Heliyon* **2020**, *6*, e05312. [CrossRef] [PubMed]
9. Husin, N.F.; Judi, H.M.; Hanawi, S.A.; Amin, H.M. Technology integration to promote desire to learn programming in higher education. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2020**, *10*, 253–259. [CrossRef]
10. Sobral, S.R. Teaching and learning to program: Umbrella review of introductory programming in higher education. *Mathematics* **2021**, *9*, 1737. [CrossRef]
11. Tikva, C.; Tambouris, E. A systematic mapping study on teaching and learning computational thinking through programming in higher education. *Think. Ski. Creat.* **2021**, *41*, 100849. [CrossRef]
12. Spolaôr, N.; Benitti, F.B.V. Robotics applications grounded in learning theories on tertiary education: A systematic review. *Comput. Educ.* **2017**, *112*, 97–107. [CrossRef]
13. Leoste, J.; Jõgi, L.; Õun, T.; Pastor, L.; San Martín López, J.; Grauberg, I. Perceptions about the future of integrating emerging technologies into higher education–the case of robotics with artificial intelligence. *Computers* **2021**, *10*, 110. [CrossRef]
14. Weng, C.; Matere, I.M.; Hsia, C.H.; Wang, M.Y.; Weng, A. Effects of LEGO robotic on freshmen students' computational thinking and programming learning attitudes in Taiwan. *Libr. Hi. Tech.* **2021**, *40*, 947–962. [CrossRef]
15. Yilmaz Ince, E.; Koc, M. The consequences of robotics programming education on computational thinking skills: An intervention of the Young Engineer's Workshop (YEW). *Comput. Appl. Eng. Educ.* **2021**, *29*, 191–208. [CrossRef]
16. Fidai, A.; Capraro, M.M.; Capraro, R.M. "Scratch"-ing computational thinking with Arduino: A meta-analysis. *Think. Ski. Creat.* **2020**, *38*, 100726. [CrossRef]
17. Arslan, K.; Tanel, Z. Analyzing the effects of Arduino applications on students' opinions, attitude and self-efficacy in programming class. *Educ. Inf. Technol.* **2021**, *26*, 1143–1163. [CrossRef]
18. Escudero, M.A.R.; Hierro, C.M.; de Madrid y Pablo, Á.P. Using Arduino to enhance computer programming courses in science and engineering. In Proceedings of the EDULEARN13 Conference, Barcelona, Spain, 1–3 July 2013; pp. 5127–5133.
19. Pala, F.K.; Mıhcı Türker, P. The effects of different programming trainings on the computational thinking skills. *Interact. Learn. Environ.* **2019**, *29*, 1090–1100. [CrossRef]
20. Park, J.H.; Kim, S. Case study on utilizing arduino in programming education of engineering. *J. IKEEE* **2015**, *19*, 276–281. [CrossRef]
21. Bicer, A.; Lee, Y.; Capraro, R.M.; Capraro, M.M.; Barroso, L.R.; Bevan, D.; Vela, K. Cracking the code: The effects of using microcontrollers to code on student' interest in computer and electrical engineering. In Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE), San Jose, CA, USA, 3–6 October 2018; pp. 1–7. [CrossRef]
22. Pratiwi, U.; Nanto, D. Students' strategic thinking ability enhancement in applying Scratch for Arduino of block programming in computational physics lecture. *J. Penelit. Pengemb. Pendidik. Fis.* **2019**, *5*, 193–202. [CrossRef]

23. Chun, H. A study on the SW coding education method using Arduino in the age of Internet of Things. *J. Phys. Conf. Ser.* **2021**, *1875*, 12–16. [CrossRef]
24. Plaza, P.; Sancristobal, E.; Carro, G.; Blazquez, M.; García-Loro, F.; Martin, S.; Perez, C.; Castro, M. Arduino as an educational tool to introduce robotics. In Proceedings of the 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), Wollongong, Australia, 4–7 December 2018; pp. 1–8. [CrossRef]
25. Schez-Sobrino, S.; Vallejo, D.; Glez-Morcillo, C.; Redondo, M.Á.; Castro-Schez, J.J. RoboTIC: A serious game based on augmented reality for learning programming. *Multimed. Tools Appl.* **2020**, *79*, 34079–34099. [CrossRef]
26. Checa, D.; Bustillo, A. A review of immersive virtual reality serious games to enhance learning and training. *Multimed. Tools Appl.* **2020**, *79*, 5501–5527. [CrossRef]
27. Ibrahim, R.; Yusoff, R.C.M.; Mohamed-Omar, H.; Jaafar, A. Students perceptions of using educational games to learn introductory programming. *Comput. Inf. Sci.* **2011**, *4*, 205–216. [CrossRef]
28. Gallego-Durán, F.J.; Villagrá-Arnedo, C.J.; Largo, F.; Molina-Carmona, R. PLMan: A game-based learning activity for teaching logic thinking and programming. *Int. J. Eng. Educ.* **2017**, *33*, 807–815.
29. Topalli, D.; Cagiltay, N.E. Improving programming skills in engineering education through problem-based game projects with Scratch. *Comput. Educ.* **2018**, *120*, 64–74. [CrossRef]
30. Chichekian, T.; Trudeau, J.; Jawhar, T. Disrupted lessons in engineering robotics: Pivoting knowledge transfer from physical to virtual learning environments. *J. Sci. Educ. Technol.* **2022**, *31*, 555–569. [CrossRef]
31. Alghamdi, M.; Younis, Y.A. The use of computer games for teaching and learning cybersecurity in higher education institutions. *J. Eng. Res.* **2021**, *9*, 143–152. [CrossRef]
32. Lopez-Fernandez, D.; Gordillo, A.; Ortega, F.; Yagüe, A.; Tovar, E. LEGO® serious play in software engineering education. *IEEE Access* **2021**, *9*, 103120–103131. [CrossRef]
33. Llerena-Izquierdo, J.; Sherry, L.L. Combining escape rooms and Google forms to reinforce Python programming learning. In *Communication, Smart Technologies and Innovation for Society*; Springer: Singapore, 2022; pp. 107–116. [CrossRef]
34. Garneli, V.; Chorianopoulos, K. The effects of video game making within science content on student computational thinking skills and performance. *Interact. Technol. Smart Educ.* **2019**, *16*, 301–318. [CrossRef]
35. Özer, H.H.; Kanbul, S.; Ozdamli, F. Effects of the gamification supported flipped classroom model on the attitudes and opinions regarding game-coding education. *Int. J. Emerg. Technol. Learn.* **2018**, *13*, 109–123. [CrossRef]
36. Asterisk. Asterisk Open Source Framework. 1999. Available online: https://www.asterisk.org/ (accessed on 12 December 2022).
37. Strava. Strava API v3: API and SDK Reference. 2021. Available online: https://developers.strava.com/docs/reference/ (accessed on 12 December 2022).
38. Verispeak. Verispeak SDK Reference. 2019. Available online: https://www.neurotechnology.com/verispeak.html (accessed on 12 December 2022).
39. Nuño, P.; Suárez, C.; Suárez, E.; Bulnes, F.G.; delaCalle, F.J.; Granda, J.C. A diagnosis and hardening platform for an Asterisk VoIP PBX. *Secur. Commun. Netw.* **2020**, *2020*, 8853625. [CrossRef]
40. JASP. JASP, a Fresh Way to Do Statistics. 2022. Available online: https://jasp-stats.org/ (accessed on 12 December 2022).