PIOT - The Smart Door

Jord Hendriks and Torben Westphalen

January 23, 2023

Abstract

Are you annoyed by searching for your door keys when coming home? We developed an IoT project which allows you to open your door by saying your pre-defined command. You can create multiple user accounts with their own commands and connect your account with a Bluetooth device of your choice. The Bluetooth device is an additional security feature, therefore your commands are only enabled if your connected Bluetooth device is nearby. This makes it easy to share an apartment without having multiple keys. You can change your commands at any time or deactivate your account.

1 Introduction

The Smart Door will listen to you, is able to recognize your keywords, and will open or close by itself. Using your own made commands, you can either get inside your house or lock it again when leaving. This application makes use of a Bluetooth connection using your phone for example. All of your own-made commands will be stored in a database. It will know when you are standing in front of your door since you are close by with your Bluetooth device and therefore recognize you and your commands.

2 Technologies

For archiving our goal, we are using a Raspberry Pi4. This controller will measure the Bluetooth strength of the nearby devices and give input to the door, in case a correct command was said. Accordingly, the user would need a Bluetooth device, which could be a mobile phone or a Bluetooth chip, which they can pair with the controller. Each user has their own commands which they can set up, as well as a password for their user account. They will then be linked to their Bluetooth device and the commands can be told.

To recognize that the user is saying something, our Raspberry Pi3 has a microphone that detects the user input. For pairing the Bluetooth devices, adding commands, and testing purposes, we are using a small display that is connected to the controller. We will send the audio input to the Google API and this will translate them back into text.

For storing the commands and passwords from the user, we use a MySQL database. The user can set a command for opening and locking the door. The Bluetooth device id and the commands will be stored, to assure that the commands are working for the correct user. For additional security, we are hashing the user passwords before storing them in the database.

3 Architecture

In our project, we have two main components. One is the microphone and one is the display. In addition, each user account is connected to a Bluetooth device which is our third sensor. The microphone is connected to the Raspberry Pi4 and the display to the Raspberry Pi3. We decided to use the two separate Raspberry Pi's for our project because the microphone is using all the I/O pin connections. Also having a display case for the other Raspberry Pi makes it difficult to add any other components to this microcontroller.

Furthermore, both microcontrollers are using the database, in order to create, update or delete records. The Raspberry Pi4, which is connected to the microphone, also uses the speech recognition

Google API to be able to convert the input audio to a string. We have visualized this architecture in the following diagram.

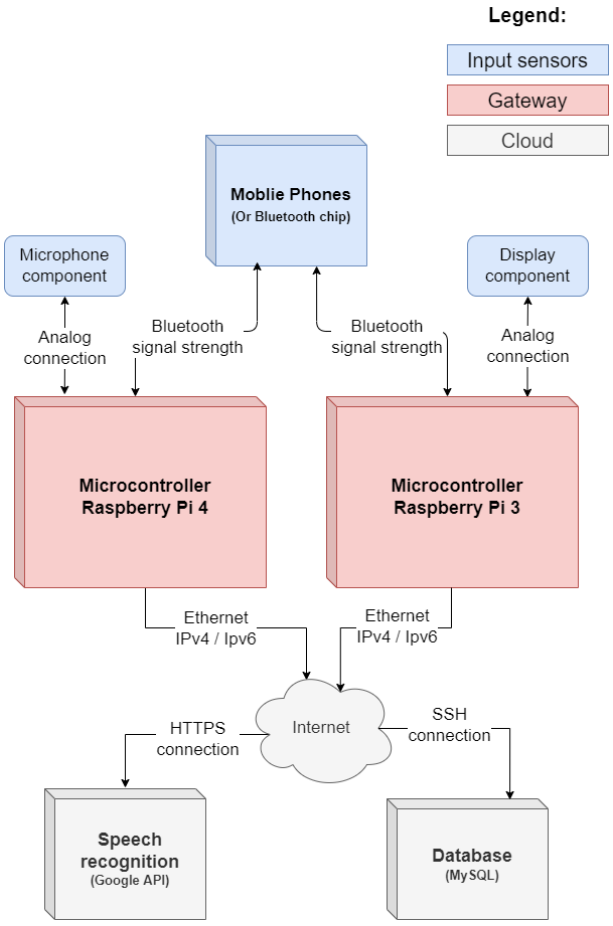

Figure 1: Hardware Architecture

4 Discussion

All main features are working right now. Users are able to register an account, they can change their settings, view all the registered users, and activate/deactivate/delete their account.

4.1 Working features

An account can be registered by pairing a Bluetooth device and assigning a username and password. With this, you also have to define your own commands for opening, closing, locking, and unlocking the door. The settings of the account can later be changed if this is desired. For this, you have to log in to your registered account with the correct username and password. This will unlock the screen for you to change all of your personal settings.

You can also delete your account if you do not want to use your account anymore. An additional confirmation dialog will pop up, to assure that this was not an accident.

If you are going on a vacation, for example, you might want to deactivate your account for a little while. This option is also integrated with the display. It will allow you to deactivate your account, which means that the program will not respond to that account anymore until you activate it again.

If you have forgotten whether your device is activated or not, it is possible to double-check this in the registered user section. All accounts are shown here with their status. It will show you if it is activated or deactivated.

These have been tested to make them completely foolproof. All the main functionalities of the display are working as they should. In addition, the connection to the Bluetooth devices is also working. In the configuration, two values can be adjusted. First, the *discover duration* defines how long the Bluetooth service should search for devices. Second, the *scan interval* describes the interval in which the Bluetooth service will scan for new devices again. Furthermore, the connection to the database is working and configurable.

4.2 What needs to be done?

Right now, the graphical user interface (GUI) has different themes and appearance modes but not all

interfaces are responsible. This means they cannot be resized and might not fit on every screen. This is because we first focused on the technical part and left the design as a less important part for later.

In addition, we wanted to provide the functionality to the users to view their own statistics. For example, when they have been at home on a day or during the week. The storing of this information is already implemented and working but the visualization is not yet done, since it is quite complex to show this in a proper way on our GUI.

5 Individual Jord

During the project, I learned about creating GUIs for a Raspberry Pi. I had to look for different Python libraries to create a GUI and see what would work and look the best. I also learned a bit about the Bluetooth functionalities possible with Python. When some problems occurred, Torben and I looked at them together to see how we could resolve the problems or how we could change the approach. For example, when a new Bluetooth device got registered, it should not show up for the next user that tries to register. This problem occurred and we sat together for a quick solution to remove it from the list of options. I also worked with the functionalities of the microphone on the Raspberry Pi. Testing made sure the spoken words would be heard by the code. I made sure everything works correctly in Python so that it could be easily implemented on the microcontroller itself.

6 Individual Torben

In this project, I mostly focused on service implementations and establishing communication with the database. Another big feature that I implemented, was the multi-threading for having the possibility to run our Bluetooth service at the same time as our GUI. In addition, I created multiple GUIs with their business logic, to implement our use cases.

I learned a lot about Python and its modules and multi-threading architecture. Working with the sensors also improved my knowledge about their speed, range, and limits. One of our main problems was, that the operating system from the Raspberry Pi3 was "Raspbian Stretch", and we needed at least "Raspbian Buster" in order to run our software on this Pi. Therefore, I gained a lot of experience while creating connections to our microcontrollers and deploying our software on different operating systems.

7 Conclusion

Generally, we are very satisfied with our result. We managed to implement the core features of this project and have a working connection to the Bluetooth devices and database.

During the project, we came across a few obstacles which we did not manage to resolve. One of that is the fact that we were not able to have one microcontroller that is connected to the display and microphone. In addition, the Raspberry Pi3 had a very old operating system version which forced us to upgrade the version and install several Python packages manually.

In the future, we could extend this project by integrating an event system. The user would then be able to create events and add a separate command to them. They can define the date, time, and duration of the event. The event itself would be connected to the user account and therefore to the Bluetooth device from the user. This way, the user can, for example, share the event command with their friend(s) and they are able to get into the house/location on their own. Also, if the host is not at home on time, the command will not work, because the event is also connected to the Bluetooth device from the host. This would be a great possibility to host events at home and the guests can join whenever they want and the doorbell will not be missed.