Marco Gabriel

Emma Jauregui

Michael Rojas

Professor Heckathorn

Computer Science 440

29 November 2022

Mini Security Alarm Analysis

The following documentation is an overall summary of the development process our group partook in when finishing and creating our project.

Our first step was deciding what creation we would construct, so we began researching different types of projects one can make using hardware such as the Raspberry Pi or Arduino. After finding various articles online about numerous hardware projects, we discovered that the projects including cameras were intriguing, as the project’s usage of the camera for security functions was seen as a very useful and productive system in the real world. After delving further into security-oriented projects, we then found the usage of sensors with the Raspberry Pi interesting enough that we would like to explore and learn their usage in hardware development. This was ultimately a big reason for our project's emphasis on tinkering with sensors. We thought making an alarm system would be an interesting and completable enough project, and a fun way to implement sensors. Our final version of our project would detect motion and sound an alarm, using PIR sensors and a Piezo buzzer.

After mutually understanding the functions and purposes of our project, we then began on its construction; This began with us researching which parts we would need to complete our desirable functions, along with ensuring compatibility amongst all the parts. We began with purchasing the most crucial part of the project, the computer board. Initially, we had planned to buy a Raspberry Pi board but soon began searching elsewhere after learning of the high prices of Raspberry Pi boards. We soon found a cheaper alternative with sufficient specs to satisfy our project's needs and functions. The name of this alternative is called AML-S905X-CC (Le Potato), which we were able to purchase on Amazon. To complete the detection portion of our system, we would have to incorporate sensors into our build. After researching various sensors, we ultimately decided to use a passive infrared sensor (PID sensor). PIR sensors measure infrared light radiating from objects in their field of view. To show the output of the PIR sensor, we decided to use a speaker to indicate motion from the PIR sensor. We used a piezoelectric speaker or a piezo speaker, which would provide sufficient sound feedback for our system. Both the PIR sensor and piezo speaker were purchased on Amazon. We also opted into using a breadboard, to allow for work on the pins to be more secure, easier, and efficient. Due to our research regarding PIR sensors, we initially believed that a 100-ohm resistor would be required to limit the power to the PIR sensor. However, we managed to get it functioning properly without limiting the power, so we opted to not include the resistor.

The first step we did was run the wiring on the breadboard. We started by placing a ground pin to a ground rail on the breadboard, then we ran a 10-volt pin to a positive rail on the breadboard. Afterward, we began wiring the piezo buzzer speaker onto the breadboard by placing it on the ground rail and pin 7 respectively. Next was connecting the PIR sensor, which we promptly did to pin 18, a 5-volt pin, and a ground rail on the board. This would be all we needed to do for the wiring aspect of the project, and we would then move on to the programming portion. After noticing most scripts for Pi-based projects were made in Python, we would follow suit and make our script in python. After extensive and quite frankly tedious research, we would import the gpiod and time modules into our script. Our main function started with getting the available GPIO chips, which would be necessary to gather the chips’ resources. The script then gathers the resources from the speaker and sensor with the pins they were connected to and set them each to a “pir\_sensor” and “piezo” variable. Afterward, both variables were initialized so that the piezo would be an output, and the pir\_sensor would be the input. After that, it was essentially a simple loop that showed motion was detected whenever the PIR sensor sent a signal indicating motion. This script goes on indefinitely unless a keyboard interrupt occurs. Lastly, we had to close the chip to properly release the resources used.

Like the development of many projects, ours was not completed without hardships and obstacles faced. The first we faced was selecting the best and most efficient operating system to use the board with. We initially believed that using the Armbian operating system would be the best, as it allowed for an efficient connection to the board. Unfortunately, using this operating system caused most functions used in the script to not be detected for an unknown reason. After multiple efforts to debug this weird issue with Armbian, we decided to switch the environment to Raspian, as we have seen various other projects that use that environment. Unfortunately, we encountered more concerns with this operating system. The operating system was difficult to navigate and traverse through, so we attempted Armbian again. This was because Armbian was similar to a Linux environment, which was more familiar to all of us. We were able to find another library that included functions that were compatible with our board. After settling with the Operating system, the next problem we encountered was managing the electrical connections with the components of our project. Shortly after, we learned that a breadboard can be used to easily manage those connections in a much more convenient manner.

The board we used was the culprit of the many tedious issues we faced when constructing this project. Some of its specs were manageable to work with, however, some of its limitations would just be tedious to work with. For example, the board only supports USB 2.0 Type A, when USB 3.0 would allow us to use the keyboard we had. This led to us spending some time looking for a compatible USB 2.0 keyboard that could be used. Another spec issue was that we could not use 5W, as it would lead to the operating system crashing. The lack of power forced us to increase its wattage to 10W to make it work. The most frustrating and difficult problem that we’d eventually overcome would be learning the libraries and functions that would be required to use our board. This was mostly due to our board having little documentation to help the user from using it. The website from our board’s creators was poorly managed; It included broken links, outdated tutorials/guides, and inactive forums with scarce support, among other things. Our opting to use an alternative board led to us conducting heavy research for functions and libraries used by our board, which took the majority of the time used on this project. We essentially had to view scarce documentation on various libraries and tested most of them until they contained functions that would work with our project/board.

Overall, this group project proved to be a very informative as well as an entertaining project that allowed us to learn about the basics of hardware development. Since all of us lacked any sort of experience with Raspberry Pi development, it was very difficult for us to learn from scratch and complete this project. Fortunately, after learning the hard way about hardware-oriented projects, we all mutually agreed that it would now be easier to develop more projects and even attempt learning more with a bit more complex project. Our initial thoughts were that hardware projects such as this one would not be too difficult to construct. It is probably safe to say we were a bit humbled by this project.