**IoT Project: PIR Motion Sensor Security**

After a lot of thinking I made the decision to focus on a security system using a motion sensor for this IoT project. My idea when starting was to have a motion detector set up at my door which would alarm my phone when someone entered its sight and also save its data on Thingspeak to be viewed later on.

**The components I used:**

For this project I went for a Raspberry PI 3B+ model and a PIR Motion Sensor SR501. I chose this model of Raspberry PI because I felt like it was a good balance of money to performance. I felt like upgrading to a 4 would not make sense for me since the performance increase wouldn’t help, not to mention the use of Micro HDMI would be less convenient for me. Downgrading wouldn’t really be a good option either because I would be sacrificing performance to barely save any money. 4 USB ports, on board Bluetooth and Wifi made the 3B+ an attractive model to buy for only €30. I went for a PIR Motion Sensor because it is a simple and effective way to detect intruders. My goal was not to make images of them, so I just needed to know whether there was movement or not by a person. I did not opt to get a breadboard because of the fact that there was one sensor that had to be connected to the PI which would be very simple and easy to do directly with 3 F to F jumper cables. As for the sensor, I went for the SR501 model. Smaller PIR Motion detectors do exist but they do not offer the same level of customizability. The SR501 model has 2 screws which can be turned left or right to adjust both delay timer of reaction and sensitivity of the distance. Both settings that would be useful to manipulate when setting up a security system. For storage I chose a Samsung EVO+ 32GB. Smaller MicroSD cards were available for me but those did not have the high read/write speeds which would benefit boot times nor did they have the heat resistance of this card which is re-ensuring when exposed to the heat of the Pi CPU.

**So the list of necessary hardware included:**

Raspberry Pi Model 3B+

MicroSD – Samsung EVO+ 32GB

Passive Infrared (PIR) Motion Sensor SR501

3x female to female jumper cables

Mouse

Keyboard

Ethernet Cable (My WIFI connection can sometimes experience issues and not work, WHILE ethernet is working just fine. So for guaranteed transmission of message I used ethernet.)

Power cable for Raspberry Pi

HDMI Cable

**Software decisions:**

For software I used Thonny IDE which was pre-installed on the Raspbian OS I downloaded. Python makes the use of GPIO pins very simple and because I had experience with Python it made even more sense to use it.

As for the Android side of the project, I soon came to realize that making an application that would always be able to communicate with my Pi was way too advanced for me to be doing right now. I then also discovered that applications to do my tasks do exist. A few examples are Pushover, Pushbullet and Pushetta. These applications use easy to implement API to send notifications from other devices such as Raspberry Pi’s to your phone when you have their application installed. I chose Pushbullet for this project to send notifications to my phone in case of intruders.

To save the data captured by the sensor I used Thingspeak. Thingspeak also uses a simple to use API to send data to the servers. Because Thingspeak also shows the date and time of data entry which is what I would want to save when talking about when an intruder came into my house, it was very convenient to use. Below is an example of how the graph looks. Every time an intruder comes through the door, he gets a number which is 1 higher than the last one. This number is uploaded to Thingspeak where the time and date of entry is saved.



**The issues:**

This project is a very cloud-based architecture. The Pi itself sends the data it receives and processes from the sensor to the router through ethernet. The router is our gateway to Pushbullet and Thingspeak. If the router would experience issues the whole system would be inoperable. As mentioned before, ethernet connection is more stable for me so I opted to use that to increase reliability and reduce system shutdown.

Another issue is the Thingspeak free servers. They allow data to be sent every 15 seconds. This combined with the fact that the PIR Sensor sends multiple readings per motion detected cause the intruder IDs to not always be 1 higher than the last one.

The next issue is that the program has to auto start. It would be inconvenient to have to connect the Pi to a screen near my door to start the program every time it boots up.

But perhaps the obstacle that hindered me the most is the output of the PIR sensor. The PIR sensor outputs a 3.3V signal to the PI when it detects movement, referred to as a high signal. The other times when it doesn’t detect movement it would send a low signal which is 0V. That is what is supposed to happen at least. However, I came to discover late in the project that the sensor I purchased was a defect product that did not regulate its voltage. I tried adjusting my code and using different cables or GPIO pins, but the programs would always act like there was constant movement being spotted. The output of the sensor was always high. I bought a new sensor and checked it for issues, the new one luckily works normally.

An additional thing to note is that according to experience and research, the PIR motion sensor seems to function better and reduce the chances of false positives when it has been turned on for 60 seconds before use to “Warm up”. I also adjusted the Sensors sensitivity to detect better in a range of 1.5 – 2m and reduced its delay to a minimum for fastest possible notifications

**Solutions and Conclusion:**

To ensure maximum stability and reduced chances of system failure I use a ethernet cable to connect my Pi to the internet. And even though the Thingspeak servers may not record 2 consecutive data uploads because motion is spotted twice within 15 seconds, the user will still be alarmed twice on his phone. The Pi is also setup to auto start the python program when booting up. This means that I only need to connect power and ethernet and it will work. The program also starts of by waiting 60 seconds before doing anything to allow the PIR sensor to get ready. Other than that, I took some measurements to stop spam notifications and spam uploads as much as I could. The Pushbullet app is also set up as a priority application on my phone, meaning the notification sound will always go off on max volume even in silent mode. The data in Thingspeak includes the time and date of upload, which is only a second or two after the motion was spotted. The code is documented properly and an explanation for each function is added.

With the issues and obstacles overcome, the program works as intended and just like I imagined it would work from the start. I learned a lot of new things during this project and got a newfound appreciation and fondness for the small device with infinite potential known as Raspberry Pi. The making of this cloud based IoT Project was not just educating, it was fun.