**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+ 128GB. A smaller microSD would have caused less issues, however this was what was available so it’s what I stuck with.

As a finishing touch, I used speakers connected to the Raspberry Pi to play a warning message when movement was detected, to further increase security.

When the implementation of everything was complete, I made a carboard box to put everything in for ease of use.

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

***For the final product:***

Raspberry Pi Model 3B+

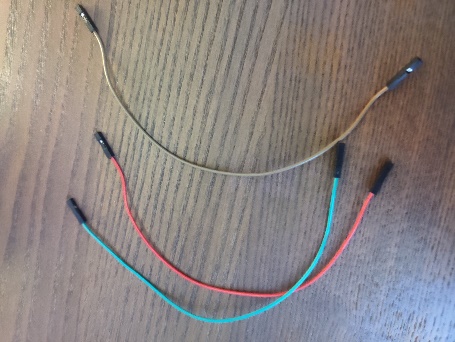
MicroSD – Samsung EVO+ 128GB

Passive Infrared (PIR) Motion Sensor SR501

Nedis-CSPR10020BU Speakers

3x female to female jumper cables

Power cable for Raspberry Pi



***For the setup:***

Mouse

Keyboard

Ethernet Cable/Wifi Connection

HDMI Cable

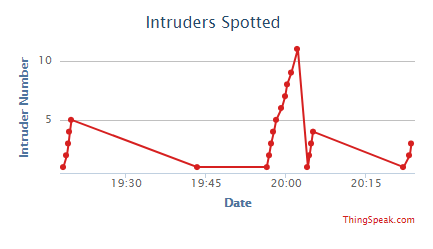
**Software decisions:**

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 already 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 tested both Pushbullet and Pushover and decided to stick with Pushbullet.

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 Pictures above show the Thingspeak graph generated by the Python Script. The script starts counting from 0 every time it restarts. This means that the Intruder ID/Number will reset to 1 every time. However this is not an issue if you keep the script running like it will in a realistic situation. You can then also clear the channel via Thingspeak to reset the graph to then fill it out with new data. The 2nd picture shows me testing the script multiple times and triggering it multiple times which is why it has many dots.

**The issues:**

1. 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, however it is not possible to use it near the door. So sadly I have to stick to WIFI. Currently I’ve had no issues with it while testing.
2. 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.

1. 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.
2. 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
3. For installing the Raspbian OS I opted to use the Noobs installer directly via the Raspberry Pi. I followed the instructions I found online yet my Raspberry Pi did not boot into the installation menu, the lights would turn on but the screen would get no input from it.
4. Pushover is a paid service with a free trial.

**Solutions**

1. As mentioned before, use of ethernet is not possible near the door where this device is supposed to stay. Next to that, my WIFI connection is hidden which is more difficult to add as a connection. But by editing the “wpa\_supplicant.conf” file in my Pi to include the line “scan\_ssid=1” the Pi is forced to scan for my WIFI until it is found. I also toggled the option “Wait for network” on so the Pi does not fully boot until it has a internet connection. This way I’m guaranteeing that when the program starts to run, there is an internet connection available.
2. Thingspeak’s free servers allow for 1 data input every 15 seconds. At first this seemed like an issue, however in a realistic scenario this is perfect for this system. An intruder should not trigger the system multiple times within a very short time anyway. So to prevent issues with spam and lost data on its way to Thingspeak I included timeouts during the program after an initial intruder has been spotted. The program will not send anything to Thingspeak or my phone for 15 seconds. This turned out to be optimal to reduce spam and accuracy.
3. To autostart the python script, I had multiple options. The issues are the fact that internet is required and that the voice message will not be played when using certain methods to run the program. The network issue was solved on solution 1. The voice message turned out to play as it would when using a crontab to auto run the python script. I added the line “@reboot sudo python3 /home/pi/PIR-IOT-PROJECT.py &” to auto run the script. If python3 is not specified, the script won’t run. And to prevent a few issues which happen with scripts like this one which are infinite loops it is important to write “&” at the end of the line. Now the program works automatically without input as long as you connect the Pi to a power source.
4. The regulator on the sensor was missing which made it incapable of functioning for this project. I noticed this very late because I am a beginner. Due to this I went through many iterations of my script, and now I know that they must have all been fine. After I bought a new PIR Sensor (The same SR501 model), my script started working as intended. I also experimented with the sensitivity issues to get them as I wanted.
5. This turned out to be a rather simple issue to solve. The problem was my MicroSD card. Because it is a large capacity card (128GB), it used the ExFat file system which is incompatible with the Raspberry Pi. Windows does not allow me to format this card to FAT32 for the Pi without the use of 3rd party programs. I used GuiFormat to format the card to FAT32 and then everything worked fine from there on out.
6. I initially used Pushover because of the more alarm-like notification sound and easier implementation into my Python script. However the trial ran out and to keep this project working I had to implement another API like that of Pushbullet or pay a monthly fee for Pushover to keep working. Pushbullet has a free and paid service but it provides everything I need with the free version, so I ended up using that for the project. The Python script I have does still contain the code for using Pushover in the future if I so desire. I would just have to uncomment it and use it.

**Final Build:**

For the final build I put everything in a homemade carboard box which ended up looking like a robot. The pi is at the bottom and is connected with USB and 3.5MM audio jack to the speakers. The PIR sensor is connected with the GPIO pins.



**Demonstratie video:**

<https://www.youtube.com/watch?v=Obue1x5DiwU&feature=youtu.be>

**Conclusion**

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.