Guardian

IOT Based Raspberry Pi Home Security System

Bocereg Alexandra, Marius Mircea May, 2018

1. Repository

The project history, schematics, diagrams and codebase are contained under the following git repository:

https://github.com/Marius2m/Project-Cerberus

2. User Requirements

1) The system must provide information regarding a detected movement.
2) The system should be open for extension (adding more sensors or cameras).
3) The system should be available 24/7.
4) The system must be able to provide live information when an event was detected.
5) The system must provide access to history data for the last 3 weeks.
6) The system should provide a way of turning it on/ off using the assistant from Amazon, Alexa.

3. System overview

Amazon Web Services
Amazon Alexa based subsystem that makes it possible to activate and deactivate the alarm system via voice commands.

Monitoring subsystem
Consists of the camera module and sensors for motion detection that are turning the camera module on.

Email subsystem
Whenever motion is detected an email is sent to the owner informing him that an intruder might have broken into his house.

Cloud storage
Consists of a Firebase Module that registers the events triggered by the sensors (motion detection) in a day.

Website subsystem
Consists of a Flask and JS website that provides a live recording of the camera and the events that have happened.

4. Circuit design

The components used in making the project are the following:

• Raspberry Pi 3B+	x 1
• Arduino Nano	x 1
• Pi Camera	x 1
• PIR Sensor	x 1
• Microphone	x 1
• Speakers	x 1
• NRF24L01 Transceiver	x 2
• Connecting wires	x 20
• Power supply	x 1

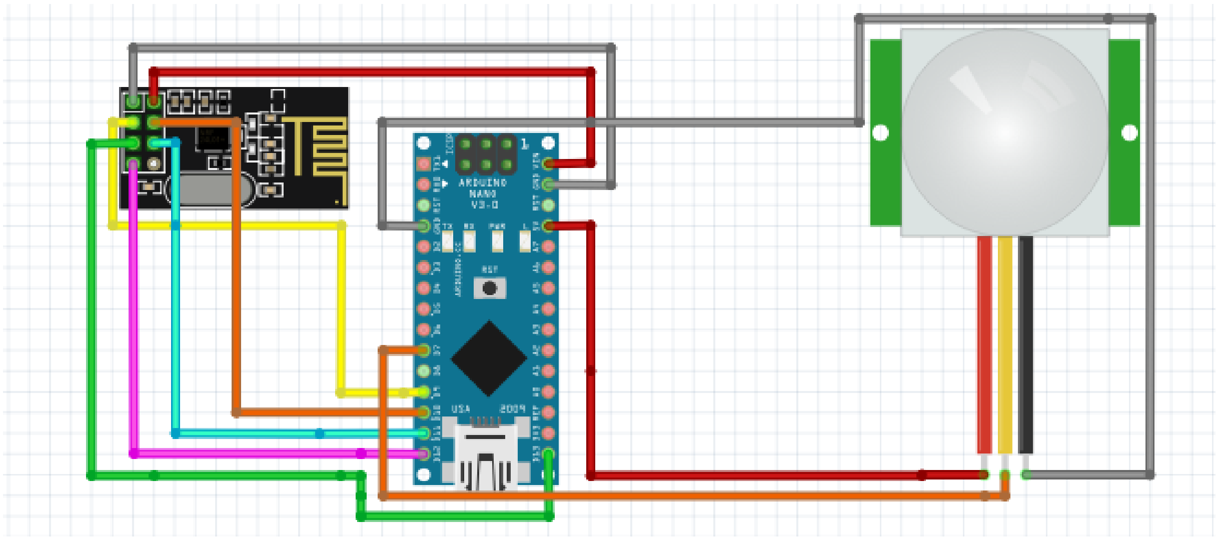
We are using Raspberry Pi 3B+ as it includes a Wi Fi and Bluetooth module, making it easier to get the application up and running and always be connected to the internet. It is a great investment as it is cheap while providing good computational power offered by its Quad-Core CPU. Powerful enough to run multiple and complex programs.

The Arduino Nano is used for reading data continuously (PIR Sensor). Despite its size, it fulfils its purpose perfectly, while keep the price cost low. Whenever the Passive Infrared Sensor detects movement (read data is above a fixed threshold) it sends the data via the nRF24L01 Transceiver Module to the Raspberry. The Raspberry reads the data received via the same module.

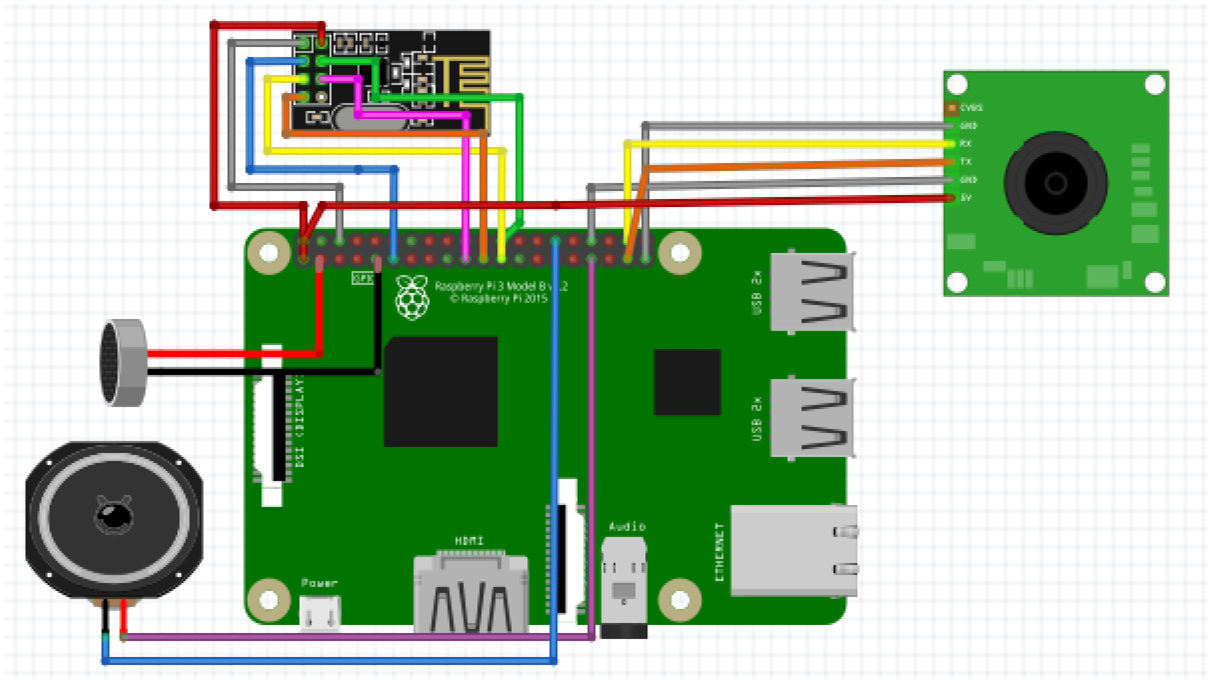
The Pi Camera Module has a dedicated library (picamera) that offers great functionality required for having a live camera on the website.

The PIR sensor is attached to the Arduino Nano and when motion is detected the Raspberry is notified.

The microphone and speakers are used for the Amazon Alexa. A custom skill is enabled on Alexa, making it possible to turn the system on and off.



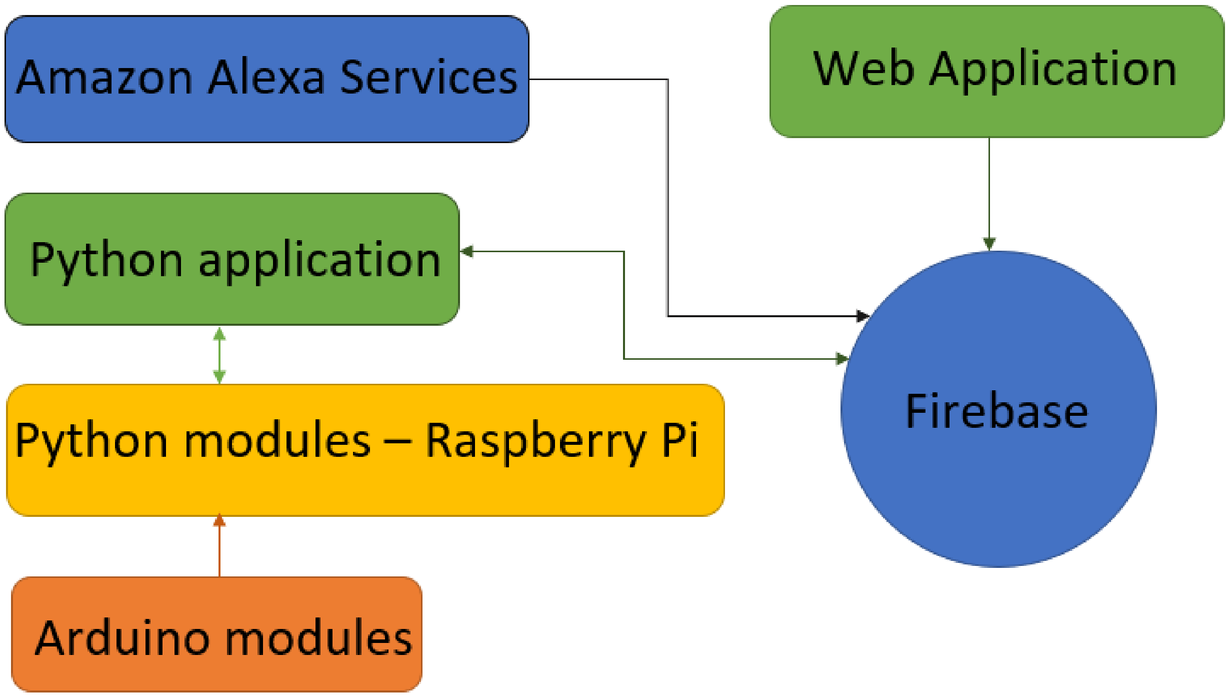
- Arduino Circuit Design -



- Raspberry Pi Circuit Design -

5. Software design

The software design is a simplistic one, based on few software components shown in the following figure.



- Software Entities -

5.1 Web Application

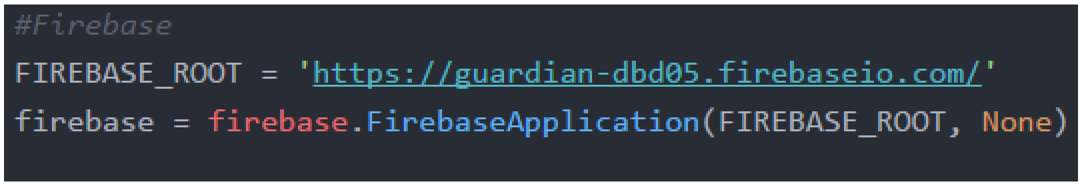
Simple web application that displays entries from firebase when movement is detected. It also shows a live stream of what the camera attached to the Raspberry Pi is capturing.

5.2 Python application & Python Modules - Raspberry Pi

Contains the modules that connect to firebase and connect to the server. Reads data sent by Arduino when the system is activated and the PIR Sensor detects motion.

Supported firebase operations:

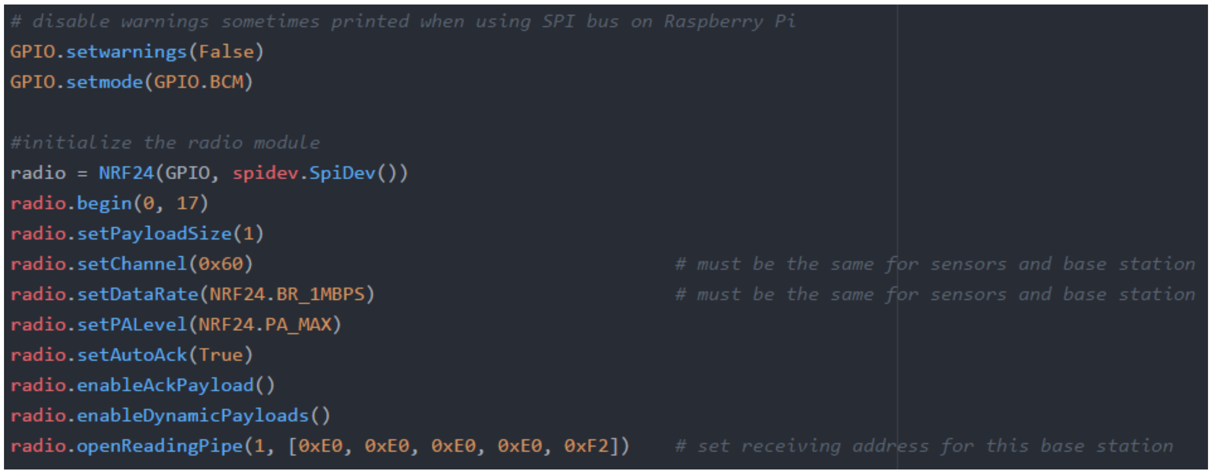
- get
- patch



```
#Firebase
FIREBASE_ROOT = 'https://guardian-dbd05.firebaseio.com/'
firebase = firebase.FirebaseApplication(FIREBASE_ROOT, None)
```

- Firebase Initialization -

To receive data from Arduino, the nrf24l01 library is used.



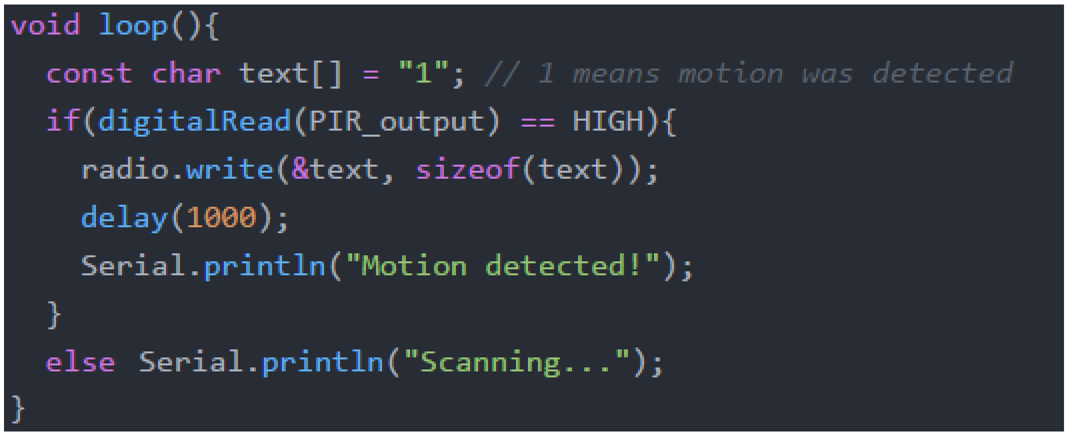
```
# disable warnings sometimes printed when using SPI bus on Raspberry Pi
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

#initialize the radio module
radio = NRF24(GPIO, spidev.SpiDev())
radio.begin(0, 17)
radio.setPayloadSize(1)
radio.setChannel(0x60)                                      # must be the same for sensors and base station
radio.setDataRate(NRF24.BR_1MBPS)                           # must be the same for sensors and base station
radio.setPALevel(NRF24.PA_MAX)
radio.setAutoAck(True)
radio.enableAckPayload()
radio.enableDynamicPayloads()
radio.openReadingPipe(1, [0xE0, 0xE0, 0xE0, 0xE0, 0xF2])    # set receiving address for this base station
```

- Initialization as receiver -

5.3 Arduino Modules

Data send from Arduino to Raspberry Pi is done via nrf24l01 again. The writing pipe from Arduino must match the reading pipe from Pi, so that the data transfer works. Using this method, multiple Arduinos can be connected to the Raspberry Pi.



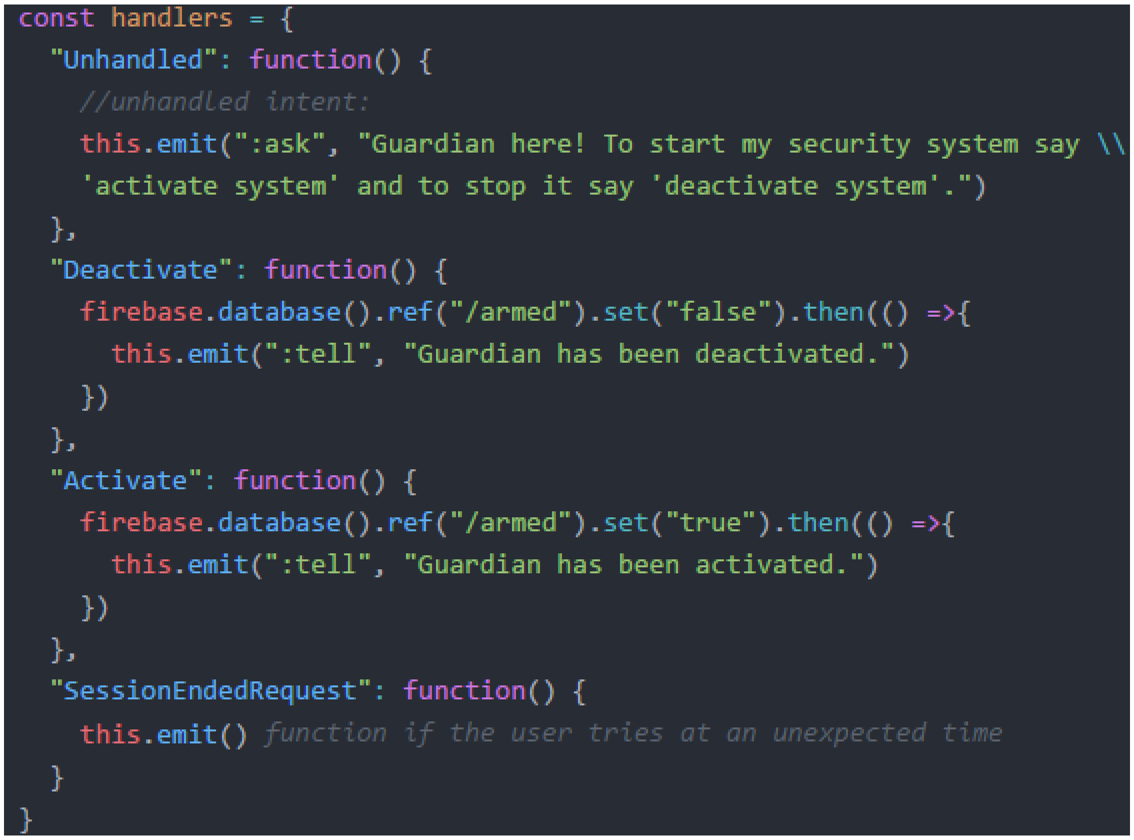
```
void loop(){
  const char text[] = "1"; // 1 means motion was detected
  if(digitalRead(PIR_output) == HIGH){
    radio.write(&text, sizeof(text));
    delay(1000);
    Serial.println("Motion detected!");
  }
  else Serial.println("Scanning...");
}
```

- Sending data from Arduino (transmitter) to Raspberry Pi (receiver) -

5.4 Amazon Alexa Services

Amazon Alexa is used to start the system. When this happens, the armed entry in the firebase is changed to true, meaning the system is alarmed (listening for events from the Arduinos). Telling Alexa to turn Guardian off, sets the armed entry to false, stopping incoming connections to Raspberry Pi from the Arduinos.

The custom skills are implemented on the Amazon Alexa Skill Console and are deployed using Amazon Web Services Lambda Functions. The intents are: activate [system, turn system on, activate security system] and deactivate [system, deactivate security system, turn system off]. The custom skill implementation is the following:



```
const handlers = {
  "Unhandled": function() {
    //unhandled intent:
    this.emit(":ask", "Guardian here! To start my security system say \\
    'activate system' and to stop it say 'deactivate system'.")
  },
  "Deactivate": function() {
    firebase.database().ref("/armed").set("false").then(() =>{
      this.emit(":tell", "Guardian has been deactivated.")
    })
  },
  "Activate": function() {
    firebase.database().ref("/armed").set("true").then(() =>{
      this.emit(":tell", "Guardian has been activated.")
    })
  },
  "SessionEndedRequest": function() {
    this.emit() function if the user tries at an unexpected time
  }
}
```

6. Results and further work

We obtained an IoT project which combines the functionalities of different systems and services.

Even though the system is in an initial stage it can be easily extended and combined with different sensors that collect data or with machine learning to recognize the home owners so that they will not trigger the system.

Our project can also be extended to include a mobile application which will send notifications to the user when motion has been detected in the house. Also, the activation of the system can be done remotely using Amazon Alexa through the mobile application.

7. References

1) Firebase Database [last seen: May 2018], *https://firebase.google.com/docs/web/setup*

2) Amazon Alexa & Amazon Web Services [last seen: May 2018], *https://developer.amazon.com/docs/ask-overviews/build-skills-with-the-alexa-skills-kit.htm*l

3) Amazon Alexa Setup [last seen: May 2018], *https://github.com/alexa/alexa-avs-sample-app/wiki/Raspberry-Pi*

4) Fritzing [last seen: May 2018], *http://fritzing.org/home/*

5) Passive Infrared Sensor [last seen: May 2018], *https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview*

6) Raspberry Pi Camera Module v1.3 [last seen: May 2018], *https://picamera.readthedocs.io/en/release-1.3/*

7) NRF24L01 Python Library [last seen: May 2018],

8) NRF24L01 Arduino Library [last seen: May 2018], *https://github.com/BLavery/lib_nrf24,*

9) http://flask.pocoo.org/docs/0.12/