ECE 471 Final Project

Ultrasonic Security System

Derek Haas, Dustin Knight, and Cameron Sullivan

December 17, 2018

Contents

List of Figures

1 Introduction

The following report describes an ultrasonic security system developed as a final project for ECE 471: Embedded Systems. The final project for ECE 471 is required to be an embedded system that does something interesting while following basic guidelines. The final design of the embedded project must implement the use of any embedded board or microcontroller and may be programmed in any language. The use of libraries may be used to simplify code, however some original code must be written to show understanding of the topics from class. One of the many low-level hardware interfaces must be utilized, while ensuring there is a user input and output. Figure 1 shows a block diagram of the ultrasonic security system.

Figure 1. Block diagram of the ultrasonic security sensor.

This ultrasonic security system utilizes a Raspberry Pi, three ultrasonic sensors, a servo motor, and a Raspberry Pi camera to detect people and take a picture of them walking by the system. The system uses three ultrasonic sensors to allow concurrent motion detection with a full 180 degree angle of coverage. The servo motor is used to rotate the camera to the sensor that detected motion. The Pi camera then takes a photo to capture the motion. After a picture is taken, the photo is posted to a web server run by the Raspberry Pi so that it may be viewed. A practical application of this project would be for a bank vault. This security system may be placed in a vault, and when someone enters the vault, their motion is detected, and a photo is displayed on the web server, which can be viewed by the security guards of the bank.

At the time of writing this paper, the security project works in ideal room conditions. The security system is able to detect a person, rotate the servo motor, take a picture, and post it to a web server within 4.3 seconds. Perfect room conditions include flat walls with nothing protruding from them, and with the assumption that the sensors are elevated at least four feet off the floor. When the system is introduced to a non-ideal room, distance measurements will vary as the sounds waves bounce off of objects that are at different distances away from the sensors.

In this report, Section 2 explains the hardware used in this security system. Section 3 covers the software and languages utilized. Section 4 describes external work that has been done that is related to this security project. Section 5 concludes the report and summarizes findings. Lastly, Section 6 references external works.

2 Hardware

The embedded surveillance hardware includes a Raspberry Pi 3B, three LV-MaxSonar-EZ0 ultrasonic sensors, a MCP3008 serial peripheral interface (SPI) analog-to-digital converter (ADC), a Towerpro SG90 servo motor, and a Raspberry Pi Camera Module V2. The system was constructed in three levels: the base, the detection block, and the camera. The base consisted of the embedded board (Raspberry Pi 3B), a breadboard, the MCP3008 ADC, and the wires that connected the components. The detection block was constructed on top of the base, with the ultrasonic sensors on three faces: left, center, and right. The servo motor is mounted in the top of the detection block with the Pi camera secured on top of the servo shaft. The ultrasonic security system can be seen in Figure 2.

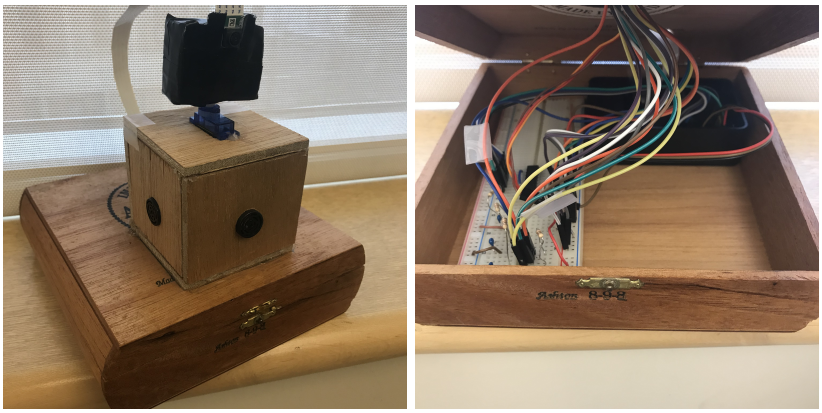

Figure 2. The constructed ultrasonic security system.

2.1 Embedded board

A Raspberry Pi 3B was used as the embedded board in the Ultrasonic Security System. The Raspberry Pi integrated all the components of the security device in the base. The central processing unit (CPU), of the Pi has a 64 bit ARM quad core architecture, with a Broadcom BCM2837 chip running at a speed of 1.2 GHz. The Pi has 1 GB of RAM. The operating system on the Pi is Raspbian Linux running kernel 4.14.79.

The single input to the Pi was connected to GPIO9. Pin 9 was the master-in slave-out (MISO) for the MCP3008 ADC. The outputs to the Pi were the GPIO pins 8, 10, 11, 18 and 20 as well as 3.3V to the MCP3008 ADC and 5V to the ultrasonic sensors and servo. Pins 8, 10, and 11 are outputs to the SPI MCP3008 ADC where pin 8 (CE0) is the chip select, pin 10 is the master-in slave-out (MISO), and pin 11 is the serial clock (SCLK). Pin 18 is the PWM output signal for the servo motor.

2.2 Input device

The input to the system is the output of the LV-MaxSonar-EZ0 ultrasonic sensor. The LV-EZ0 ultrasonic sensor uses ultrasonic sound to detect the distance an object is from the transducer. The

range of the sensor is between 6 inches to 254 inches away (below 6 inches detects as 6 inches). Depending on the input voltage (3.3V or 5V), the beam characteristics of the sensor will be larger or shorter. The output beam of the LV-EZ0 ultrasonic sensor is provided in the sensor datasheet [1] and shown in Figure 3.

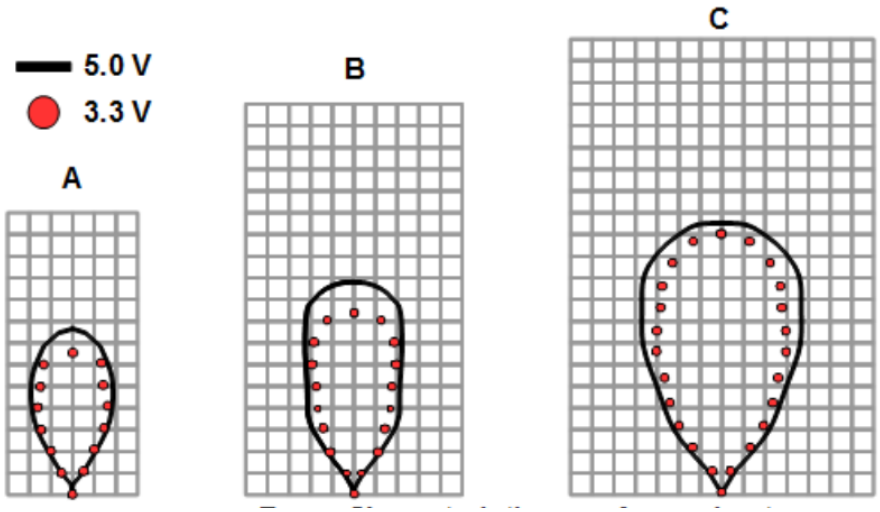

Figure 3. Beam pattern characteristics of LV-MaxSonar-EZ0 [1].

These three beam characteristics demonstrate the detection of dowels of varying size. Beam A, B, and C are measurements of 0.25-inch, 1-inch, and 3.5-inch diameter dowels. The squares of the grid are scaled to 1 foot by 1 foot. It is noted on the data sheet that the detection pattern most accurate for detecting people is the beam characteristics between beams A and B [1]. The sound output of the LV-EZ0 is a cone shape beam, and the input is the time it takes for return of that beam. When sensors are in close proximity to one another, the return signal a sensor recieves may be the signal from a different sensor. This interference is called "cross-talk" in the LV-EZ0 datasheet [1]. The datasheet provides three different solutions for chaining. One of these solutions is called "analog output constantly looping," a diagram of which is shown in Figure 4.

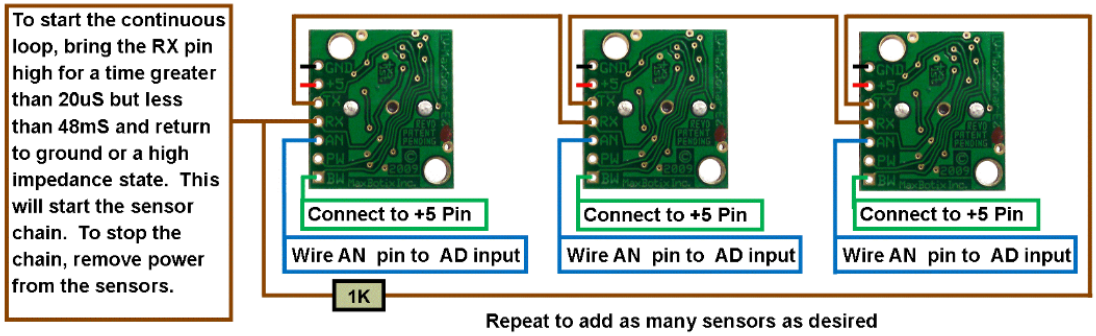

Figure 4. Analog output constantly looping connection diagram [1].

Chaining works by having each sensor pulse and echo before the subsequent sensor ranges, which prevents the sound waves from one sensor from entering another sensor. Additional pins are provided on the LV-EZ0 to enable chaining. The BW pin is used to indicate the mode of the TX output pin, this pin is held high at 5V. The RX pin receives a signal from both the TX pin and the first input signal (only for first in sequence) to initiate ranging for sensor. The TX pin sends a pulse to the RX pin of the next sensor to indicate ranging is complete [1].

The ultrasonic sensor outputs an analog voltage that is read into one of the MCP3008 channels. The MCP3008 is run with SPI that converts the analog voltage into a discrete value. Using

$$vm = \frac{value * 3.3}{1024}, \tag{1}$$

where value is the digital input from the ADC and vm is the measured voltage [13]. From the measured voltage, the distance in inches is calculated with Equation (2) [14].

$$distance = \frac{vm}{vi} \tag{2}$$

The vi is volts per inch, given as 5V / 512 inches [14]. This final distance value is used in the detection algorithm.

The protocol to interface with the ultrasonic sensor is: 1) Ultrasonic sensor sends out a ping of ultrasonic sound, and recieves the signal, 2) Ultrasonic sensor outputs an analog voltage, 3) Analog voltage is input to a channel on the MCP3008 ADC, 4) The MCP3008 provides a digital value over SPI, which is converted to a measured voltage, 5) Using the measured voltage, distance is calculated.

2.3 Output device

The output of the systems are a servo motor and the Raspberry Pi camera. The servo rotates the Pi camera, the camera captures a photo, and the Pi uploads the picture to a web server. The servo is the Towerpro SG90 180 degree motor that is coded in C. To rotate the device, pulse width modulation (PWM) is used. The PWM runs at a period of 20 ms. A ~1 ms pulse rotates the servo to the left, a ~1.5 ms pulse rotates the servo to the center, and a ~2 ms pulse rotates the servo to the right [15]. The camera is the Raspberry Pi Camera V2 [9], coded in python. The Python library `PiCamera` was used to communicate with the Pi camera.

The embedded board, the Raspberry Pi 3B, is powered by a wall outlet. There are no power concerns with this device. Power consumption could be improved by using interrupts to trigger the servo as opposed to having it run constantly.

3 Software

3.1 Programming Languages

Three programming languages were used in the development of the security device: C, Python, and HTML. C was used for most of the hardware interaction. Specifically, the drivers for the ultrasonic sensors and the servo motor were written in C. C was used because it is a low-overhead language, so it has a minimal load on the Raspberry Pi. Low overhead is an important feature of this project, because it needs to have the fastest real-time response possible.

Python code was written for the camera and the web server. The camera code could have been written in C, but the Python `PiCamera` is the standard Raspberry Pi camera library, so it provides the most dependable and effective control of the Pi camera. The `PiCamera` library allowed the team to create a PiCamera object and run functions on that object, such as `start_preview()`, `capture()`, `close()`, and others. Python was also used to run the web server. This is because Python offers the `Flask` module that allows for simple and effective web server management. The use of Flask allowed for a web server to be created in only 14 lines of code. Using the Flask library, the team created a Flask object, and the function `app.run(debug = True, host = '0.0.0.0', threaded=True)` was run to create a webpage that ran on the localhost using the `index.html` markup as its source. Overall, all code in the project files was written by the team, though references were used to learn how to write the code. The only code that was not original to the group was library code, including `PiCamera`, `Flask`, and other minor libraries that made the project simpler.

HTML was used to design the visuals of the webpage. This is because HTML is the standard markup language for webpages, so it allowed the team to easily place the image from the camera on a public website.

If the code for the project was written in assembly, the code would take up less disk space and would run more quickly. The disk space advantage would be good for a system with minimal disk space, but the Raspberry Pi used for this project has a 64 GB SD card, so the code density of assembly is of minimal benefit. The speed advantage of well-written assembly could offer some benefit to the project, if the C code is not compiled to run as efficiently as proper assembly. However, there are significant sources of delays in the project, such as a 1 second camera preview, 500 millisecond servo rotation delays, and 50 millisecond ultrasonic polling delays. These delays are of much greater magnitude than any differences between the execution speed of assembly and C, so it would not be particularly advantageous to convert the code to assembly.

3.2 Real Time Concerns

The ultrasonic security system is a security device, so a fast real time response is essential. Because this project is a complex system, the device has two real time constraints.

The first real time constraint involves the detection and capture of a security breach. The detection and capture of a security breach is a firm real-time constraint. There are three conditions that make this constraint firm. First, if an ultrasonic sensor responds too slowly and misses a person walking through its beam, then the information is useless, because the sensor completely missed the

person. Second, if the servo responds too slowly to detected motion and does not rotate in time to allow the camera to capture a photo, then the photo will not contain the malicious actor, so that image will be useless. Third, if the camera doesn't capture a photograph quickly enough, then the photo will not contain the malicious actor, so it will be useless. These three potential slow responses would prevent the system from properly detecting and capturing a person, so they constitute firm real-time constraints. The team measured this initial system response by starting a timer when a person moved in front of a sensor, then stopping the timer when the servo pivoted to the sensor that the person was in front of. This response time was measured to be 1.22 seconds. The camera takes 1 second to preview, so the complete response time for a photo to be taken is approximately 2.22 seconds.

The second real time constraint involves the transmission of a photograph onto the website. When a breach is detected, and a photo of the breach is captured, that photo is saved to the disk then uploaded onto the Flask web server. These actions take time, so there will be a finite delay between when the photo is captured and when it is visible on the webpage. This delay will not make the information useless, because the breach photograph will still appear on the webpage, but the photo will be less useful because when it isn't seen as soon, so this photo upload delay constitutes a soft real-time constraint. The full real-time response of the system was measured, by having a person stand in front of a sensor. A timer was started when the person stood in front of the sensor, and the timer was stopped when a photo of the person was visible on the webpage. This response time was measured to be 4.3 seconds. The team considers this a suitably-fast response. If this device was used in the real world, a security personnel would be able to see an image of a security breach only 4.3 seconds after the breach occurred. The fast real-time response that the team measured would allow the security team to respond quickly enough to mitigate the threat.

3.3 Security Concerns

The ultrasonic security system has three security concerns: physical security, device security, and web server security.

The physical security of the system is a concern. This is because a malicious actor has the ability to disable the system if they can get close enough to it. Specifically, the device is designed to be powered by a wall outlet, so if a malicious actor could unplug the Pi, the system would be disabled. The second physical security concern is the Pi camera cable, which could be unplugged from the camera if a malicious actor got close enough to the device. If the Pi camera cable was unplugged, malicious actors would still be detected, but images would not be captured. Other than these two physical security concerns, the device is secured within a box that could be locked to prevent access. Also, though these two security concerns are legitimate, the system would be able to detect and capture images of the malicious actor before they reach the device, so it would still accomplish its purpose of alerting security personnel to a breach before it was disabled.

The second security concern is device security. For this project, the Raspberry Pi was configured to allow incoming SSH connections, which allowed the team to work on the Pi remotely. If a malicious actor was able to gain SSH access, they could disable the device and possibly cause physical damage to the device. In order to mitigate this threat, if the device was implemented in the real world, the team would either disable SSH access completely or add SSH key authorization to prevent malicious access.

The final security concern is web server security. For the demonstration of the project, the Pi camera images were posted on a plain web server. This public web server allows anyone to access the site. There is the possibility that a malicious actor could disable the webpage or possibly even access the Pi through the webpage. In order to only allow authorized users to access the site, the team proposes the future addition of a login page to the website.

4 Related Work

Distance sensing is an application of embedded systems that is frequently pursued. The sensors used for distance sensing vary from the classic HC-SR04 ultrasonic sensor [2] to passive infrared (PIR) sensors [3], to the very complicated radar [4] or LiDAR [5] devices. The output devices that interface with humans can vary from a simple LED that turns on when a person is detected to an LCD display, or a camera that captures an image of the motion.

4.1 Ultrasonic Detection Projects

The LV-EZ series ultrasonic range finders are very commonly used in a variety of distance sensing applications. As described in the datasheet [1], the LV-EZ0 sensor is used for "people detection, security, motion detection, autonomous navigation, robotics, and collision avoidance."

The LV-EZ series sensors have been frequently used for range detecting projects. Source [6] used a combination of four HC-SR04 sensors and one LV-EZ0 sensor to detect the actions of a human standing, sitting, and falling. Source [7] used a single LV-EZ0 in a sonar system to detect obstacles. Source [8] used a single LV-EZ0 sensor that rotates on a stepper motor to make a radar system.

4.2 Raspberry Pi Camera Projects

The Raspberry Pi Camera Module V2 [9] is the official camera for the Raspberry Pi, so it is very commonly used in all different kinds of Raspberry Pi projects. When the Pi camera is used as a security camera, projects typically use image processing to detect motion. Source [10] uses the Pi camera and Python libraries to implement an image-processing security system. Source [11] uses the Pi camera and a custom OS to provide advanced image-based security. Source [12] was the only online source found that combined the use of distance sensors and the Pi camera to detect and capture motion. This source used the combination of a PIR sensor and an ultrasonic sensor to detect motion, and used the Pi camera to take photos.

The ultrasonic security project is similar in some way to all of the referenced projects. The device uses ultrasonic range finders, which are commonly used for a wide variety of human detection projects. The device also uses the Pi camera, which is commonly used for a wide variety of security camera projects. The unique part about the team's device is that it combined ultrasonic human detection with the photography of the Raspberry Pi camera to provide an effective security system. Only source [12] developed a similar prototype, and their device was very different because it used different types of sensors and actuators, and all code was written using high-level libraries. Source

[12] also did not use a servo motor to rotate the camera. The team's project is a new innovation in the security industry that offers potential to become a standard device for a variety of applications.

5 Conclusion

The hardware, software, and related work of an ultrasonic security system has been described. Team members for this project are Derek Haas, Dustin Knight, and Cameron Sullivan. Derek owns a Raspberry Pi camera, so his task in the project was to figure out how to take a picture and upload that image to a web server using python. Dustin took advantage of the ADC given in class to receive analog outputs from the three ultrasonic sensors. Dustin also developed the distance sensing and calculating code. Cameron used his SG90 servo motor from the ECE 271 labs and wrote PWM code in C to control it. When these three key components of the project were finished, the three team members worked together to put the components together. The main task that the three team members collaborated on was the development of an algorithm to detect motion based on the distance measurements from the ultrasonic sensors.

5.1 Challenges

Many challenges arose during the development of the project. The main challenge was cross-talk between the ultrasonic sensors. Cross-talk complicated the project because the ultrasonic pings from one sensors would be received by the sensor next to it, giving incorrect measurement readings. The cross-talk between sensors caused distance measurements to decrease, as a sensor received sound earlier than it expected. Terminal output of the measurements associated with cross-talk is shown in Figure 5. To solve this issue, the transmission and receive pins of the ultrasonic sensors were used. When one sensor would get its reading, it would trigger the next sensor to get its reading, and so on. This solution corrected the measurements and gave more stable readings.

```
Distance from right sensor: 0.000000
Distance from center sensor: 38.280000

Distance from right sensor: 0.000000
Distance from center sensor: 24.420000

Distance from right sensor: 0.000000
Distance from center sensor: 25.080000

Distance from right sensor: 0.000000
Distance from center sensor: 38.610000

Distance from right sensor: 3.300000
Distance from center sensor: 38.940000

Distance from right sensor: 5.280000
Distance from center sensor: 38.610000

Distance from right sensor: 5.610000
Distance from center sensor: 37.290000
```

Figure 5. Terminal output showing the decreasing distance measurements that occur during cross-talk. The correct distance measurement is about 38 inches. The 24-25 inch measurements are from cross-talk.

Another challenge included posting an image to a web server. The team was not very familiar with using web servers, so they had difficulties figuring out how to dynamically upload photos to a website. Basic HTML code was written, along with simple Python web server code. When this code was written, it was unclear as to how to update the picture in real time. While using the `Flask` module in python, it was noticed that the server would have to be restarted for each update. It was also noted that the viewer of the webpage must use Firefox to view the dynamic updates. If they used any other browser, they had clear their cookies and cache before reloading the page.

With a security device, it is important to have all aspects of the system act quickly in the case of an intruder. This posed to be a challenge to the group, as each task required a brief moment of time to initialize and execute. Delays were shortened in each respective code file, however the response time of the system is not at the optimal instantaneous speed of detection. Further delay tuning will be done to improve the real-time response rate of the security system.

5.2 Future Work

The team has many ideas of how they can improve the security system. The first addition the team plans to make is code to run the system at boot. When the Raspberry Pi first boots, the code is not automatically executed, so the team has to plug in a monitor and keyboard in order to open a terminal and manually run the code. A real security system must work once it is plugged in. Also, the team did not implement a safe option to power off the pi, so a script will be written to perform a safe shutdown when a certain GPIO connection is opened.

Second, there is work that the team plans to do with the ultrasonic sensors. First, even after hours of measurements and tuning, the team is still having issues with inconsistent ultrasonic sensor readings. The reason for this is not known for sure, but is suspected to be inherent to the sensors themselves. The sensors project a wide cone of sound (up to 4 feet in diameter), which is good for detecting people within a wide radius, but is bad because it means that unwanted objects will be detected. These unwanted objects include the table the device rests on, anything sticking up from the ground, and anything projecting outwards from the wall. It is difficult to account for inconsistent physical objects in code, so the team may need to replace the LV-EZ0 sensors with a different model that projects a narrower cone of sound. Second, the ultrasonic sensors are placed in a configuration that causes a dead zone at the corners of the detection block. These dead zones occur as the cones of sound from the three sensors are not perfectly aligned, so there is a gap between the cones where no sound travels, so objects cannot be detected. To fix this issue, the sensors could be placed in a more optimal configuration to eliminate the dead zone space.

Overall, the team is pleased with the functional prototype that they designed and constructed for the ECE 471 final project. Though there are many aspects that could be improved for better, more consistent device performance, the security system did function as intended when tested. The video of this successful test was recorded and sent to professor Weaver.

6 References

[1] MaxBotix Incorporated "LV-MaxSonar-EZ Series High Performance Sonar Range Finder", 2018. [Online]. Available: https://www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf [Accessed: 16- Dec- 2018].

[2] Elec Freaks "Ultrasonic Ranging Module HC-SR04", 2018. [Online]. Available: https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf [Accessed: 16- Dec- 2018].

[3] Adafruit "PIR (motion) sensor", 2018. [Online]. Available: https://www.adafruit.com/product/189 [Accessed: 16- Dec- 2018].

[4] Sparkfun Electronics "A111 Pulsed Radar Breakout", 2018. [Online]. Available: https://www.sparkfun.com/products/14811 [Accessed: 16- Dec- 2018].

[5] Sparkfun Electronics "TFMini - Micro LiDAR Module", 2018. [Online]. Available: https://www.sparkfun.com/products/14588 [Accessed: 16- Dec- 2018].

[6] Ghosh Et al. "On Automatizing Recognition of Multiple Human Activities using Ultrasonic Sensor Grid", 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7945440 [Accessed: 16- Dec- 2018].

[7] Eady, Fred. "Building a Sonar System", 2008. [Online]. Available: https://www.servomagazine.com/uploads/issue_downloads/898/Building_a_Sonar_System.pdf [Accessed: 16- Dec- 2018].

[8] Ramya Et al. "Embedded Controller for Radar based Robotic Security Monitoring and Alerting System", 2018. [Online]. Available: https://pdfs.semanticscholar.org/b946/c80a688afaa0ab143f86d62b079702cc7652.pdf [Accessed: 16- Dec- 2018].

[9] Raspberry Pi Foundation. "Camera Module V2", 2018. [Online]. Available: https://www.raspberrypi.org/products/camera-module-v2/ [Accessed: 16- Dec- 2018].

[10] Golge, Eren. "RaspberryPi Home Surveillance with only 150 lines of Python Code", 2017. [Online]. Available: https://hackernoon.com/raspberrypi-home-surveillance-with-only-150-lines-of-python-code-2701bd0373c9 [Accessed: 16- Dec- 2018].

[11] Drake, Nate. "Build a Raspberry Pi CCTV camera network", 2016. [Online]. Available: https://www.techradar.com/how-to/build-a-raspberry-pi-cctv-camera-network

[12] Rao, Maneesh. "How to Build a Surveillance System With Raspberry Pi 3", 2018. [Online]. Available: https://dzone.com/articles/how-to-build-a-surveillance-system-with-raspberry

[13] Microchip. "2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface", 2018.

[Online]. Available: http://web.eece.maine.edu/ vweaver/classes/ece471_2014f/datasheets/MCP3008.pdf

[14] MaxBotix. “Finding Distance Using Analog”, 2018. [Online]. Available:
https://www.maxbotix.com/tutorials6/032-using-analog-voltage-pin-3.htm

[15] Towerpro. “Servo Motor SG90 Datasheet”, 2018. [Online]. Available:
http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf