NAIT

Edmonton, Alberta

Home Security System

As a submission to

Mr. Kelly Shepherd, Instructor

English and Communications Department

&

Mr. Marc Anderson, Instructor

Computer Engineering Department

Submitted by

Faiza Yahya, Student

&

Joseph Bossio, Student

CMOE2960

Computer Engineering Technology

March 12, 2020

2508-43 St NW

Edmonton, AB T6L 6C1

March 12, 2020

Mr. Kelly Shepherd, Mr. Marc Anderson
Instructors
NAIT
1762-106 Street NW
Edmonton, AB T6L 6C1

Dear Mr. Shepherd and Mr. Anderson,

As per requirements of CMPE 2960, I am submitting the report titled Home Security for evaluation.

This report has all the details of how the project was initiated, planned, executed and closed. The report also documents the successes and failures throughout the project. In addition, the report includes the programming codes and structure used in the life-cycle of the project. It also includes the electrical diagrams used throughout the project.

I would like to thank all of the instructors at NAIT for their continued support throughout the semesters. I appreciate their energy and enthusiasm in teaching that has helped us guide and be able to complete this capstone project.

Sincerely,
Joseph Bossio & Faiza Yahya
CNT Students

Table of Contents

List of Figures and Tables……..3-6

Abstract…….7

1.0 Introduction……8-9

2.0 Home Security Overview……10-11

3.0 Design

3.1 Electrical Design……12-13

3.1.1 Base Station Design

3.1.1.1 PIR Sensor ……14-15

3.1.1.2 LCD Display……..15-17

3.1.1.3 Night Vision Camera ……..17-18

3.1.1.4 Start/Stop Push Button……..18-19

3.1.2 Bluetooth Station Design……19

3.1.2.1 ESP-32 Microcontroller ……20-21

3.1.2.2 Door Sensor (Reed Sensor)……21-23

3.2 Software Design

3.2.1 Base Station Software……..23-24

3.2.1.1 PIR Sensor ……24-25

3.2.1.2 LCD Display……25

3.2.1.3 Night Vision Camera ……25-28

3.2.1.4 Start/Stop Push Button……28-29

3.2.2 Bluetooth Station Design……29-31

3.2.3 Integration ……31-32

3.3 Project Results……32-33

3.4 Conclusion ……33-34

References……35

List of Figures and Tables

Figures

Figure 1: Base Station ……12

Figure 2: PIR Sensor Top View……..14

Figure 3: PIR Sensor Bottom View ……14

Figure 4: PIR Sensor Connections……15

Figure 5: Simple 16x2 LCD ……16

Figure 6: LCD Circuit Layout……17

Figure 7: Night Vision Camera Raspberry Pi……18

Figure 8: Push Button Circuit……19

Figure 9: ESP-32 Pin Layout ……20

Figure 10: ESP-32 Circuit with Door Sensor……21

Figure 11: ESP-32 Circuit Diagram with Door Sensor ……22

Figure 12: Door Sensor Circuit Diagram ……..23

Figure 13: Code PIR Sensor ……..25

Figure 14: Code Motion Detection Night Vision Camera……26

Figure 15: Code Check Camera ……27

Figure 16: Code Capture Image ……28

Figure 17: Code Push Button Initialization……..28

Figure 18: Code Push Button Callback Function Initialization ……29

Figure 19: Code Push Button Callback Function ……29

Figure 20: Code Bluetooth Class……31

1.0 Abstract

Home Security system is essential part of everyday living. Raspberry Pi was set as a main station to connect all of the sensors and cameras. The Raspberry Pi was programmed using Python IDE and python. The ESP-32 Microcontroller was utilized as a sub-station and is connected to a reed sensor. The communication between the sub-station(ESP-32 Microcontroller) and main station(Raspberry Pi) is done over a Bluetooth. The main program of Raspberry Pi has a constant while loop which detects any motion. If the door or window was opened, or motion was detected, the alarm triggers immediately. The main station also has a camera attached to it. If a person is in front of the door a picture is taken using Raspberry Pi night vision camera and sent via email to the user. The camera uses motion detection that was implemented in software.

The integration of the project was tested using different case scenarios. The user is responsible for starting the Alarm which is done by pressing a push-button. Once activated, the alarm can be triggered by multiple factors such as PIR sensor (motion sensor) or reed sensor (door/window sensor) and the result is displayed on the LCD screen. In order to deactivate the alarm a different set of push-button is pressed to deactivate the alarm. The different tests cases confirmed that alarm was activated by all the trigger elements. The motion detection triggered the camera to send a email via SMTP protocol. This satisfied the goal of developing a successful home security system. This indicated that base station and the main station integration were working perfectly together as a single unit.

2.0 Introduction

Home Security System is very commonly available area to many commercial and residential buildings. There is a demand for a lower cost security system that can be implemented at home by using simple microcontrollers, and SMTP technology to send an email if a person approaches the front or back door. This system allows for one time installation not a monthly fee so overall the security system that is designed in this capstone is very versatile.

The main goal of this report is to outline the work and decisions made during the execution of Home Security System. The report will discuss the general overview of the project. After the general overview, the report will dive into the design and implementation of major components: main station using Raspberry Pi and substation using ESP-32 Microcontroller. Each microcontroller controls different sensors. The conclusion of the project will be based on a integrated project as one unit. The outcomes of the project will also be discussed in the conclusion. The success of the project will based on the ability to finish all the required deliverables listed in the scope of the project.

The main goal of the Home Security System was to develop a home security system that is cost efficient and does not require a monthly maintenance fee. The Raspberry Pi is connected to a PIR sensor which can detect motion, a camera which can take a picture if motion is detected, LCD which displays the data, and On/Off switch to activate the alarm. A reed sensor is connected to ESP-32 Microcontroller which triggers the alarm if the door or a window is opened. The reed sensor was placed with ESP-32 Microcontroller because the need for the system to be able to operate at different remote locations through out the home. There needs

to be a reed sensor at every door/ window through out the home. These two systems need to communicate with each other as a single working unit.

3.0 Overview

The home Security System consists of three parts. The first part is the main station, the second is the camera with motion detection implemented using software , and the final part is sub-stations connected to reed sensor. The camera is triggered via movement of an object in front of it and send an email using SMTP protocol. The main station and substation communicate via Bluetooth sending a ‘o’ or ‘c’ string terminated by null which indicates if door or window has opened anywhere in the house.

The camera motion detection was implemented using software which compares a change in a picture with another picture that is taken within a small time frame. After that, the program compares the picture pixel by pixel to see if there is change. If a visible change is observed then, it takes a picture and sends it over SMTP protocol. The program can set the threshold of picture for example how many pixels need to be changed for a motion to trigger. The program also has ability to set the region to analyze the particular region and the motion detected only in that region. This is very useful when there are trees nearby and wind causes a constant motion.

The Raspberry Pi is connected to PIR sensor which detects motion using a levels of infrared radiation. When a warm body like human passes by it generates a positive differential change(Anonymous, 2020). These change pulses are detected and the buzzer is then triggered. The PIR sensor has ability to set the sensitivity range which is the detection area using a potentiometer and the time the sensor stays high after the motion is detected using a potentiometer. The result is displayed on an LCD screen which indicates the alarm is beeping.

There are also two switches connected to the Raspberry Pi that enable and disable the alarm and the result is displayed on the LCD screen.

The ESP-32 Microcontroller is programmed using the C++ application and it comes with built-in Bluetooth module. A circuitry needed to be built to pick up a signal from reed sensor as it had too low current. Utilizing a transistor the circuit was able to generate a 3.3Volts. This helped indicate whether the reed sensor was opened or closed. Only if the reed sensor was open (door/window opened) the ESP-32 Microcontroller sent command over Bluetooth to Raspberry Pi to indicate a motion.

3.1 Design: 3.1.1 Electrical Design

The base station is where all the home security information is gathered and processed. The base station has circuitry for PIR Sensor , night vision camera , start and stop switch and LCD screen to display all the data.

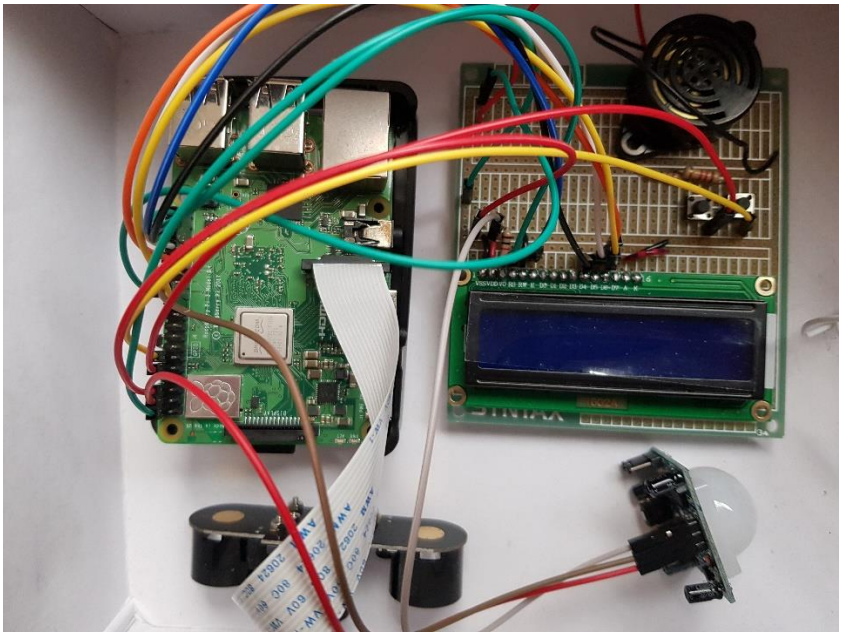

Figure 1 Base Station

As per Figure 1, all the sensors and components are connected to Raspberry Pi. The pins on Raspberry Pi that were utilized were GPIO pins mainly. The board is powered by using 5 volts from the Raspberry Pi. A general overview of the Base Station and its circuitry is described below into four components.

The PIR sensor has a detection range of 170 x 70 degrees and sensitivity range of 7 meters this allows the coverage of an entire room(Anonymous, 2001). These sensors are to be placed in the rooms nearby the front and the back door. For LCDour purposes in this project,

there was only one PIR sensor utilized. If this were to go commercial there will be several PIR sensor used. The circuit for PIR sensor is very straightforward: there are three pins connected which are ground, digital output, and 3-5 VDC input. Whenever the digital output pin goes high, the buzzer starts beeping for one second.

The LCD is used to display the output data for example, every time a motion is detected in front of the door for a camera or PIR sensor causes the alarm to beep. For the LCD circuitry, most of the pins are connected according to the recommendation on the datasheet. The LCD used here is 16 x 2 LCD display that uses Hitachi HD44780 driver. The LCD is operating on full back light brightness and contrast control pin is connected to 1k resistor so that the screen is not to bright.

The night vision camera is has two infrared sensors that are placed on either side of the camera module for Raspberry Pi. The camera first needs to be assembled with screws to create connectivity to the camera. The camera is connected to the Raspberry Pi using the CSA-Camera Port. This allows the connectivity to the Raspberry Pi.

The start and stop button switches are simple 4 pin push-buttons. Since the buttons originally could not send a solid signal to the GPIO pins, pull-up resistors were necessary.

All these circuitry was successfully integrated together on single bread board. The components are all soldered together and connected via jumper cables.

3.1.1.1 Base Station Design: PIR Sensor

The PIR Sensor has two potentiometer: one for adjusting sensitivity meaning the range the motion is to be detected, the other is for adjusting the time the output signal stays high when the motion is detected. In this project, the PIR sensor is configured by leaving motion sensing range to about 1-3 meters and the output stays high as long as person or a moving object is in the sensing range. The time potentiometer was not changed. In Figure two it is visible that PIR has ability to detect a wide range. The Figure three shows the three pins on button and two potentiometers on top.

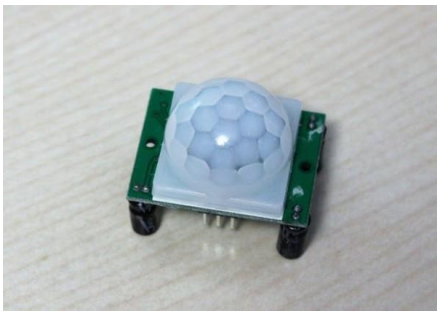

Figure 2 PIR Sensor from top view

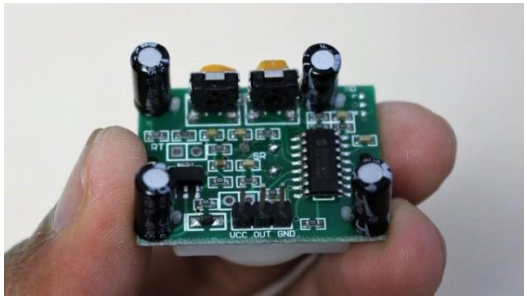

Figure 3 PIR Sensor showing three pin, and two potentiometers on top

The basic connection to the Raspberry Pi is shown below in aa simple layout in Figure four.

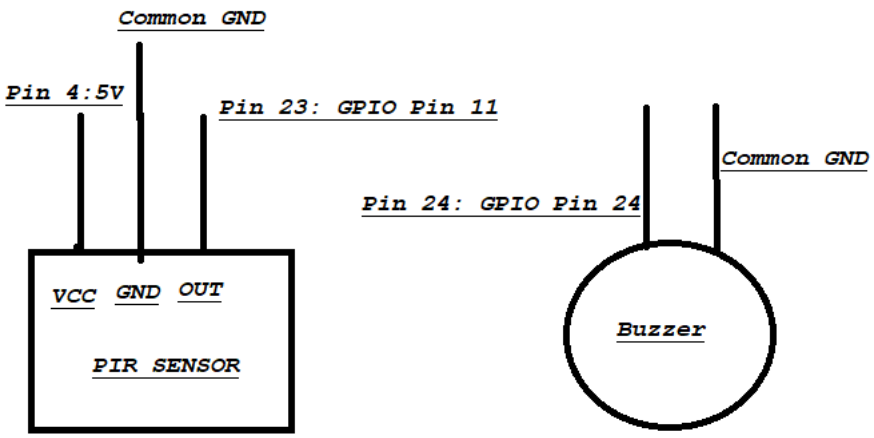

Figure 4: Left Shows the pins of PIR Sensor. On right is the Buzzer and its pin layout

PIR Sensor is simply connected by three pins and the buzzer is connected by two pins. Each of them have shared ground. The Raspberry PI comes with regulated 5 volts and that was used as VCC.

Whenever motion is detected, by detecting change in the IR level of the detection zone it is able to sense motion. Once the motion is detected, the GPIO pin on the buzzer is made high and the buzzer beeps for a set time setting the alarm. The alarm only goes off if the person is out of the detecting range.

3.1.1.2 Base Station: LCD Display

The LCD display is there for output display for change of any event in the entire home security system. It tells the user, alarm is beeping, alarm is armed, alarm is disarmed, and if the door or window was open. The connectivity is based on what is recommended off the data sheet with few adjustments. Instead of using potentiometer to control the backlight brightness and the contrast control, it is hard-wired. The backlight brightness pin is Pin A which is simple

connected to five volts for maximum brightness. In addition, contrast control is connected to 1k resistor to five volts to reduce the brightness to a level where the words on LCD are visible.

Figure 5 Shows a simple 16x2 LCD Screen

The connectivity diagram is shown below. The LCD is used in 4 bit mode with D4, D5, D6 , and D7 used for data (Raj, 2015). The rest of the operation is initialized via code. Figure six shows the connectivity of LCD to Raspberry PI.

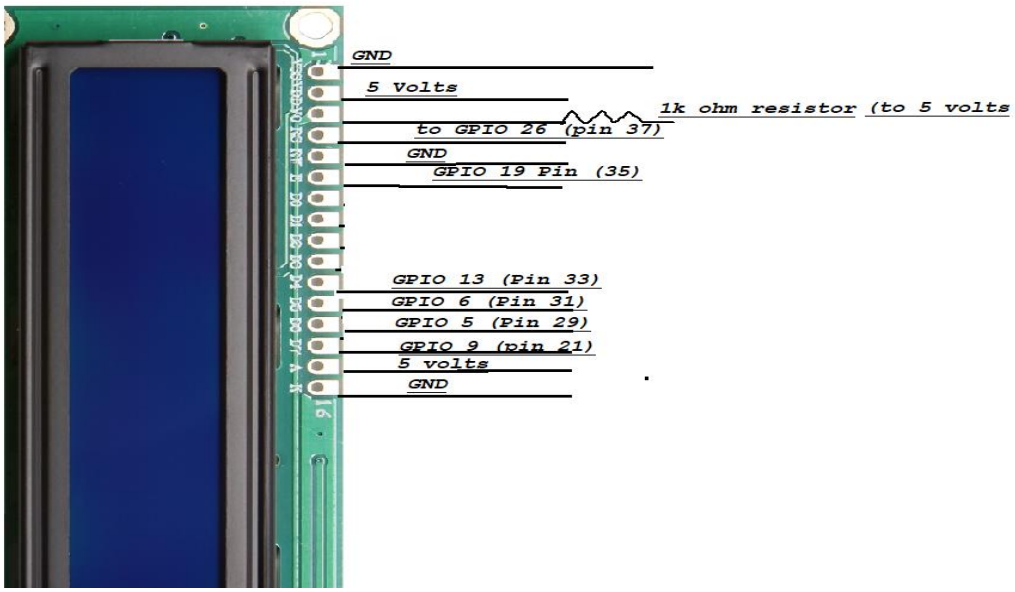

Figure 6 Pin Layout of LCD Screen

3.1.1.3 Base Station: Night Vision Camera

The night vision camera has two infrared sensors. Most of the implementation is implemented using software. The two infrared sensors are bolted to the camera and it is connected using the ribbon cable to the CSA Camera port on Raspberry Pi. The Figure seven shows the basic camera and its configuration to the Pi.

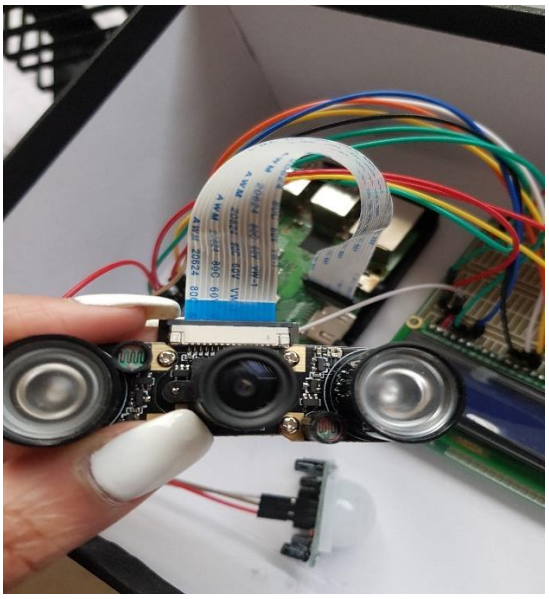

Figure 7 Raspberry Pi Night Vision Camera

3.1.1.4 Base Station: Start/ Stop Push Buttons

The start and stop buttons that are being used are push buttons. When start is enabled the alarm is armed. When stop is pressed the alarm disarms. Working with push buttons involves using pull-up resistors. This is because when the push-button is released, the system does not know what is connected to the input pin, so the pin is floating. That means it is exposed to the noise. The software will not be able to read high and low uniformly. Therefore a 2.2 k pull up resistor was used for both of the switches connected to the 5 volts. A simple diagram of the circuit is shown in Figure 8.

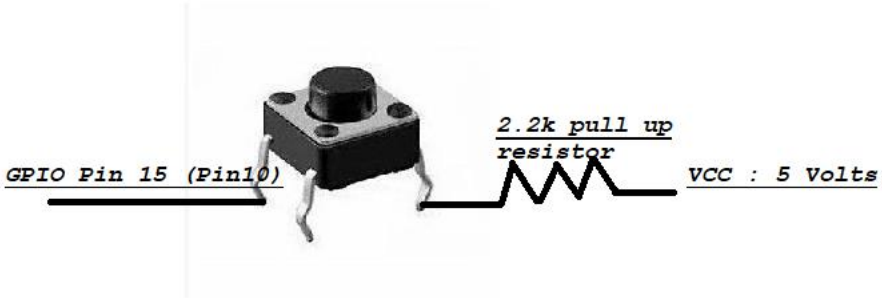

Figure 8 Simple layout of a push button switch

3.1.2 Bluetooth Station Design

The Microcontroller, ESP-32 is connected to the reed sensor which is used for a door/ window sensor. When the reed sensor indicates that the door is open, it indicates a state change. Upon the state change it sends the signal out over the Bluetooth to the main station which is Raspberry Pi. The data sent to the Raspberry Pi is in form of character indicating an ‘o’ for open door.

The magnetic switch sensor or a reed sensor helps check the status of door or a window The circuitry using a PNP transistor was needed to be able to detect a signal that has high signal only when door or window is open. The PNP transistor has voltage higher on the emitter side than the base which helps turn the transistor on, creating a connection between the emitter and collector. This configuration helps in detection of signal if the door or window is opened.

3.1.2.1 Bluetooth Station Design : ESP-32 Microcontroller

The chip has a built in camera and Bluetooth module. This chip is fairly small in size. This helps built a smaller size circuits that can be mounted with a magnetic reed sensor on the doors and windows. These microcontroller are programmed using C++. The pin layout for the ESP-32 microcontroller is shown below.

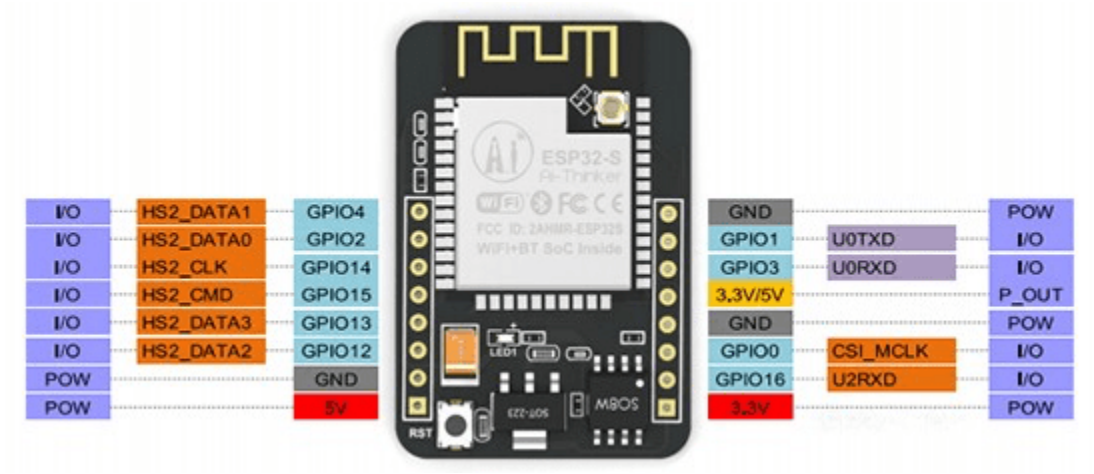

Figure 9 Pin output for ESP-32 Cam Microcontroller

The only pins that are utilized in this Bluetooth station is GND, VCC, 3.3V, and GPIO pin 13. The microcontroller itself is powered by the 9 volts battery that is stepped down by 3.3 volts converter that connects to 3.3V/5V pin on the microcontroller. The Figure 10 shows the circuit for the Bluetooth station.

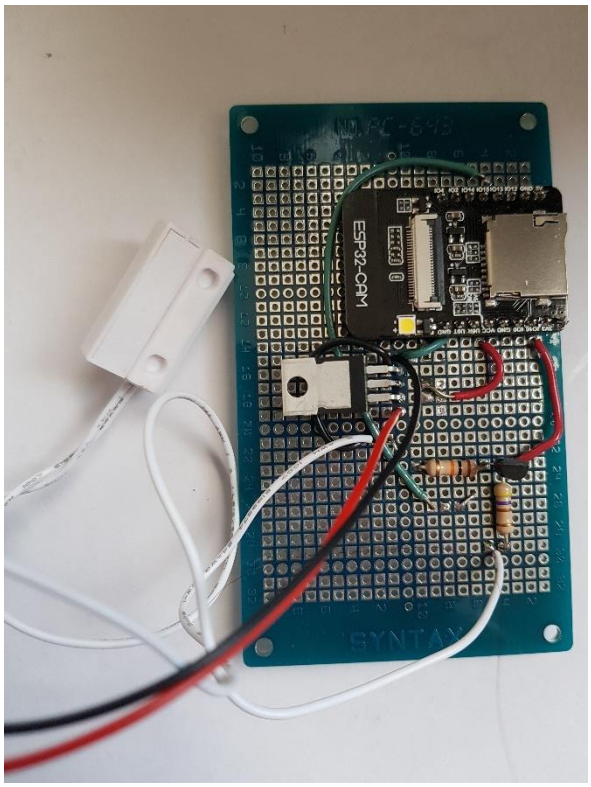

Figure 10 Substation with ESP-32 CAM Microcontroller Circuit

The white switch shown in Figure 10 is the door sensor. The black and red lead goes to a battery. More details on operation is implemented using code so it is discussed in the software side.

3.1.2.2 Bluetooth Station Design: Door Sensor (Reed Sensor)

To be able to read outputs from magnetic reed sensor it is connected to the PNP transistor to indicate a high or low signal. The circuitry for the door sensor is shown below in Figure 11:

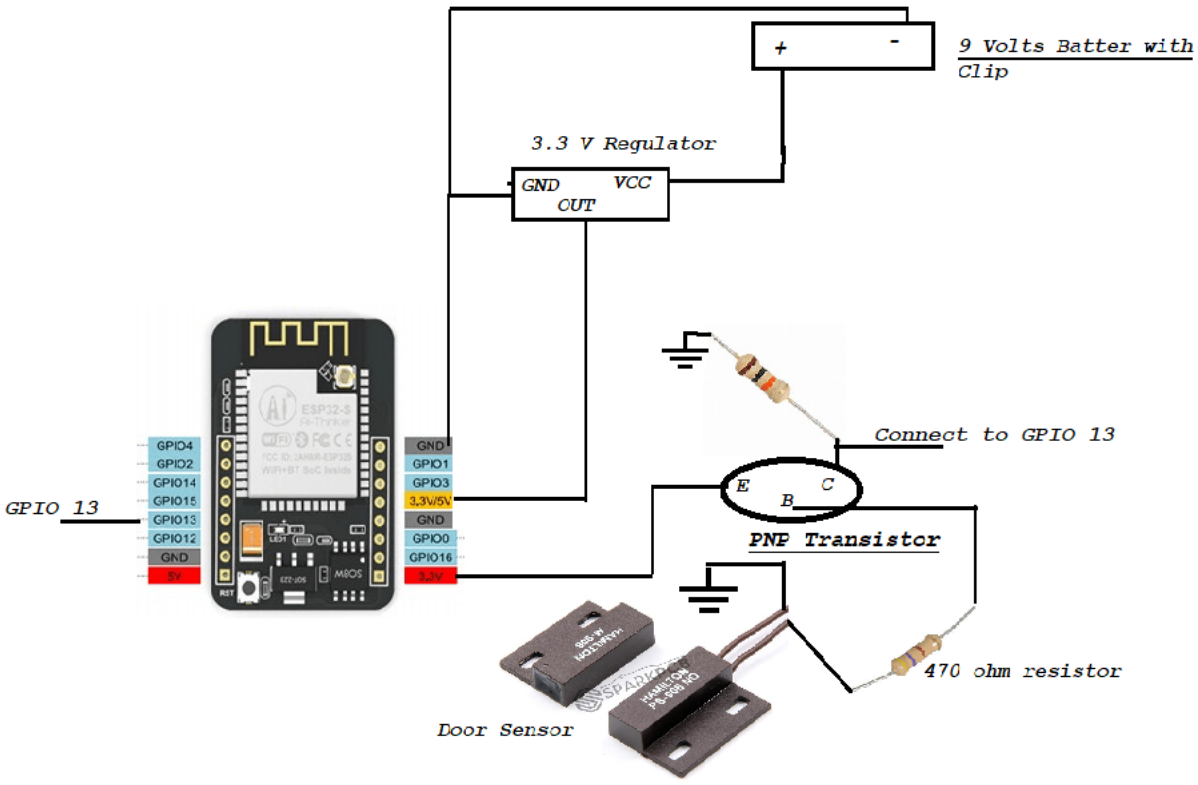

Figure 11 Shows the Substation with the reed sensor

When the magnetic switch is closed as shown in the diagram a signal of low is being read from the GPIO pin. When the magnetic switch is open the circuit is closed and GPIO Pin 13 reads a voltage of 3.3 Volts, indicating the door has opened. When the switch is open the transistor turns on the collector and emitter acts like a short circuit indicating a high at GPIO Pin 13. This mechanism is shown is Figure 12. The data is only sent to the Raspberry Pi when the door is opened in a character format over Bluetooth. It does not matter which of the two cables of magnetic switch is connected to the base of the transistor. One free end can eb attached to the door or window and the one with cable is attached to the door or window frame.

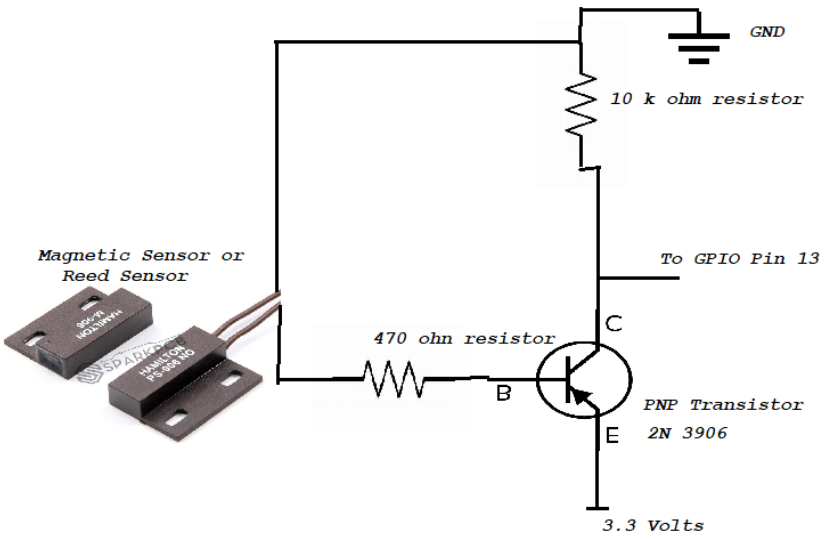

Figure 12 PNP Circuit Diagram

3.2 Software Design: 3.2.1 Base Station Software

In the software side of the base station is written in Python using the Python IDE compiler. The Raspberry Pi night vision camera code is written as a separate file and is then imported into the main program. The SMTP protocol to save a picture is also written in a separate file and is imported to the main program. The main program also has libraries imported for Bluetooth and threading. The program starts by using pressing a push button switch. If the push button switch is not pressed the alarm is not enabled. There is an callback function that sets the Boolean variable to true when the enable push-button is pressed. The main program then has a while loop that calls “check motion”, “check camera” or “check door sensor” functions repeatedly

When the motion is triggered , camera detects the motion or door is opened the program enters the required function accordingly and enables the alarm. The alarm will keep beeping depending on the what triggered it and will only disable when user presses the push button again to disable the alarm. Each function checks for the Boolean variable “IsAlarmed” and if the motion is detected, the program updates the LCD screen accordingly. The length of the buzzer is dependent on the type of motion that triggered it. For example, if picture is taken the buzzer gives 0.1 second beeps three times.

The program for motion sensor for camera simple compares two instances of images pixel by pixel and returns true or false indicating the change in the pixel colors. The sensitivity or the amount of pixel changed in order for the motion to be detected is set by sensitivity level constant in the beginning of the program. The program also lets the user exclude the regions where there is trees constantly moving that could set the alarm.

The program for SMTP simply connects to the user’s email that is hardcoded in the program and sends an email to the address when the camera detects that there is a person at the front or the back door. The path of the picture is passed to the program to indicate where to retrieve the saved file. This way it can be extracted then emailed to the user.

3.2.1.1 Base Station: PIR Sensor Software

PIR Sensor code initializes in the main program by setting the output pin on PIR sensor as a GPIO input because we are reading data from the output of sensor pin. Then there is a call to the function “check motion movement” in the main while loop. If the output pin on the sensor indicated a high that means there is motion detected. See the Figure 13 below that shows the method written in python. The

buzzer is set as a output GPIO pin and the value is written to it. This way a value of high or low can turn the buzzer on and off easily. Before writing to the LCD, a threading functionality lock (displaylock.acquire) is used so that LCD screen is not overwritten and a garbage value is not displayed.

```
def check_motion_movement():
    global isAlarmed
    if GPIO.input(MOTION_S): #If there is a movement, PIR sensor gives input to GPIO23
        GPIO.output(BUZZER, True) #Output given to Buzzer through GPIO24
        displayLock.acquire()
        lcd.clear()
        if isAlarmed:
            lcd.write_string(u'BEEP .. BEEP ..')
        else:
            lcd.write_string(u'Alarm Disabled')
        time.sleep(1) #Buzzer turns on for 1 second
        GPIO.output(BUZZER, False)
```

Figure 13: PIR Sensor Method triggers alarm upon motion detection

The LCD is cleared and then written to. The PIR Sensor causes the buzzer to trigger for 1 second on every iteration in the while loop till the push button disables the alarm or user leaves the sensitivity area.

3.2.1.2 Base Station : LCD Display

The LCD Module is initialized by importing a RPLCD library. Then the pins are initialized as data pins, enable pin, reset pin, and the numbering mode is set. The numbering mode can be either board mode or pin mode for Raspberry Pi pin configuration. Throughout the main program if the alarm is triggered by opening a door, motion sensor, or by motion detected by camera, the LCD screen is updated accordingly. Every time the LCD screen is updated, the lock is used from the threading library to make sure that the LCD screen does not overlap the data. Once the screen is written to the lock is released. If a new data is to be written to the LCD, it is cleared first.

3.2.1.3 Base Station Software: Night Vision Camera Software

The motion detection implemented using Raspberry Pi camera involves capturing two images and comparing each one of them to see if there is any change in sensitivity. In this program, the sensitivity is set to 20 meaning if 20 pixels are changed that means there is motion detected. It compares the pictures pixel by pixel to detect a change. This motion detection is in an entire different file that has a function called motion returning a Boolean expression indicating if the pixels are different from one image to another. In the main program this is imported as a library and the function motion is called to check if motion is detected. The Figure 14 shows a snippet of the code entering a for loop that compares a picture pixel by pixel.

```
while (True):

    # Get comparison image
    image2, buffer2 = captureTestImage(cameraSettings, testWidth, testHeight)

    # Count changed pixels
    changedPixels = 0
    takePicture = False

    if (debugMode): # in debug mode, save a bitmap-file with marked changed pixels and with visible testarea-borders
        debugimage = Image.new("RGB",(testWidth, testHeight))
        debugim = debugimage.load()

    for z in range(0, testAreaCount): # = xrange(0,1) with default-values = z will only have the value of 0 = only one scan-
        for x in range(testBorders[z][0][0]-1, testBorders[z][0][1]): # = xrange(0,100) with default-values
            for y in range(testBorders[z][1][0]-1, testBorders[z][1][1]):   # = xrange(0,75) with default-values; testBorder
                if (debugMode):
                    debugim[x,y] = buffer2[x,y]
                    if ((x == testBorders[z][0][0]-1) or (x == testBorders[z][0][1]-1) or (y == testBorders[z][1][0]-1) or (
                        # print "Border %s %s" % (x,y)
                        debugim[x,y] = (0, 0, 255) # in debug mode, mark all border pixel to blue
                # Just check green channel as it's the highest quality channel
                pixdiff = abs(buffer1[x,y][1] - buffer2[x,y][1])
                if pixdiff > threshold:
                    changedPixels += 1
                    if (debugMode):
                        debugim[x,y] = (0, 255, 0) # in debug mode, mark all changed pixel to green
                # Save an image if pixels changed
                if (changedPixels > sensitivity):
                    takePicture = True # will shoot the photo later
                if ((debugMode == False) and (changedPixels > sensitivity)):
                    break  # break the y loop
            if ((debugMode == False) and (changedPixels > sensitivity)):
                break  # break the x loop
        if ((debugMode == False) and (changedPixels > sensitivity)):
            break  # break the z loop

    if (debugMode):
        debugimage.save(filepath + "/debug.bmp") # save debug image as bmp
        print ("debug.bmp saved, %s changed pixel" % changedPixels)
    # else:
    #     print "%s changed pixel" % changedPixels

    # Check force capture
    if forceCapture:
        if time.time() - lastCapture > forceCaptureTime:
            takePicture = True
```

Figure 14 Raspberry Pi motion detection loop comparing image pixel by pixel

In the main program the “check camera” function repeatedly checks if there is a change in picture by calling the motion function from the imported code. The Figure 15 shows the “check camera” function implemented in the main program. If there is motion detected, a current timestamp is obtained and the picture is saved at a current directory. The picture is saved with the timestamp so the user can identify which picture was taken at particular time. The capture image function does this procedure and stores the image in the given path along with the timestamp.

```
def check_camera():
    global isAlarmed
    motionState = P3picam.motion()
    print (motionState)
    if motionState:
        currentTime = getTime()
        captureImage(currentTime, picPath)
        alert_with_speaker(0.1, 3)
        displayLock.acquire()
        lcd.clear()
        if isAlarmed:
            lcd.write_string(u'PHOTO TAKEN')
        else:
            lcd.write_string(u'Alarm Disabled')
```

Figure 15 Check Camera function the main program constantly calls upon this to detect motion

The capture image is called withing the check camera function. In addition, in the capture image function as shown in Figure 16, the email code is called and the image is emailed using SMTP protocol. When capturing the image, the buzzer beeps three times very quickly to indicate the picture has been captured and emailed to the user.

```
def captureImage(currentTime, picPath):
    #Generate the picture's name
    picName = currentTime.strftime("%Y.%m.%d-%H%M%S") + '.jpg'
    with picamera.PiCamera() as camera:
        camera.resolution= (1280, 720)
        camera.capture(picPath + picName)
    email4.pictureSave(picPath+ picName)
    print("We have taken a picture.")
```

Figure 16 Capture Image function saves the picture to the specified directory and calls the email code to send a email with a picture if motion is detected

3.2.1.4 Base Station Software: Push Button Software

The initialization of the push button involves setting the push buttons with software implemented pull-up resistor. The Figure 17 shows the software implemented pull up resistors. Both of the push buttons are set as GPIO inputs as the data is read from them.

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.IN, pull_up_down= GPIO.PUD_DOWN)
GPIO.setup(10, GPIO.IN, pull_up_down= GPIO.PUD_DOWN)
GPIO.setup(MOTION_S, GPIO.IN)
```

Figure 17: Initialization of Push Button inputs as GPIO

A callback function was necessary that was detected on the rising edge. If the call back function was not implemented then, there would be switch debouncing leading to the function being called many times within small framework. To avoid that callback functions were necessary so that whenever there is a rising edge on either buttons a function will be called that will enable or disable the alarm. Figure 18 shows the initialization of callback functions on the rising edge meaning when the button is pressed.

```
        elif val == "o":
            lcd.clear()
            alert_with_speaker(0.1, 2)
            print("Door opened")
            lcd.write_string(u'Door opened')
        displayLock.release()

GPIO.add_event_detect(10,GPIO.RISING,callback=button_callback_disabled)
GPIO.add_event_detect(12,GPIO.RISING,callback=button_callback_enabled)

##doorSensor = Bluetooth(doorSensorName, 1)
```

Figure 18: Callback Functions on Rising Edge of the Clock for push buttons

Once the call back functions are triggered, the LCD screen is updated accordingly indicating the alarm is enabled/disabled and the Boolean flag “IsAlarmed” is set. If the alarm is disabled, then the buzzer is also written a low to turn it off. The LCD as per usual is locked to make sure there is no data overridden. Figure 19 shows both the call back functions as described above for push buttons.

```
def button_callback_disabled(channel):
    global isAlarmed
    isAlarmed = False
    print("Stopped Alarm")
    displayLock.acquire()
    lcd.clear()
    lcd.write_string(u' Alarm Disabled ')
    displayLock.release()
    GPIO.output(24, False)

def button_callback_enabled(channel):
    global isAlarmed
    isAlarmed = True
    print("Start Alarm Mode")
    displayLock.acquire()
    lcd.clear()
    lcd.write_string(u' Alarm Enabled ')
    displayLock.release()
```

Figure 19: Alarm Enable/Disable with Push Button

3.2.2 Bluetooth Station Design Software

The ESP32 GPIO pin checks for high or low voltage depending on door senor being open or closed. The C++ code inside the ESP32 microcontroller checks for door state change every 300

milliseconds. Upon a state change (door sensor going from open to close or close to open), the Bluetooth serial protocol sends a character 'c' or 'o' over the Bluetooth to any client listening to it.

Once Raspberry Pi main program runs, it makes a serial connection to the ESP32 microcontroller Bluetooth chip. The main program uses another thread to receive 'c' and 'o' messages from ESP32 Bluetooth module and puts them into a queue. The queue is checked in the main program to see if a new message is received from the door sensor. Depending on the message that the main program removes from the queue, the main program displays the message on the LCD screen and beeps 3 times if the door is opened and 2 times if the door is closed. The need for thread to receive messages from ESP32 Bluetooth was needed as python Bluetooth library (pybluez) receive function is a blocking call (does not allow the new statement in the program to execute). If it is in our main program, it will block other operations like motion detection and writing to the LCD screen which will make the system behavior incorrect. The Figure 20 shows the Bluetooth class in the main program. This initializes the main connection with the Bluetooth protocol.

```
# Bluetooth connection class
class Bluetooth:
    def __init__(self, name, port):
        self.deviceName = name
        self.address = ""
        self.port = port
        self.socket = None
        self.isDiscovered = False
        self.isConnected = False
        self.previousVal = ""

    def discover_device(self):
        devices = bluetooth.discover_devices(lookup_names=True)
        print("Found {} devices.".format(len(devices)))

        for addr, name in devices:
            if name == self.deviceName:
                print("Found {} - {}".format(addr, name))
                self.address = addr
                self.isDiscovered = True
                #create socket
                self.socket = bluetooth.BluetoothSocket( bluetooth.RFCOMM )
                return True
        return False

    def connect(self):
        if self.isDiscovered:
            self.socket.connect((self.address, self.port))
            self.isConnected = True
            return True
        else:
            return False

    def receive_message(self):
        if self.isConnected:
            data = self.socket.recv(100)
            val = data.decode('ascii')
            return val[-1]

    def __del__(self):
        print ("Closing connection ...")
        self.socket.close()

def recv_door_sensor_vals(sensorBT, dataQueue):
    global isAlarmed
    while(True):
```

Figure 20 Bluetooth Class in main Program

3.2.3 Integration

Integration of PIR Sensor, LCD, Push Buttons, and Camera Module was successfully achieved. The main program constantly calls upon checking the output of the PIR Sensor, queue of data received over the Bluetooth and the change in pictures using camera code. Each sensor has its own function. Inside each function, if the motion is detected, LCD is locked and the data is updated accordingly. In addition each sensor has a unique was of setting a buzzer. The alarm continuously keeps ringing till the

user decides to turn it off. In terms of integration, the hardest element to integrate was Bluetooth module. This was the biggest obstacle that was overcome using threading.

The biggest issue was adding the Bluetooth module to the main program that initially caused a blocking call. If the Bluetooth receive function is inside the main while loop it waits till the door is opened. Thus, the rest of the functions are not called. This was one of the greatest hindrance in the project. This required some research and a conclusion was made to use a separate thread to see the messages from ESP-32 microcontroller. The thread will continuously check for messages being received from the ESP-32 Bluetooth module and store them in the queue. The main program will only check the queue to see if there are any received messages. If the door or window is opened, there will be an "open message" inside the queue and thus that will trigger the alarm to beep three times indicating that the door is opened.

Once this was achieved the project was able to run cohesively together in the main program.

3.3 Project Results:

The first goal of the project was to make a motion detection sensor using just a simple camera. This was the most time consuming task as it was very complex software to write. Using the idea of comparing two images taken few seconds after each other and comparing them pixel by pixel the code was successfully written that can detect motion. The picture is successfully taken and the motion detection works perfectly.

The second aspect of home security was to detect any motion inside a room with few sensors. After some research the PIR sensor was the best choice as it had angular range as well allowing it to cover the entire room easily. The motion detection using PIR sensor was placed at

different areas of the house to best understand how to set it. After testing the PIR sensor, it was decided to set to continuously beep till the moving object leaves the sensitivity zone. The wall of the room itself stops the IR sensor to be active beyond the room if the room is not as wide as seven meters.

The third major aspect was to have a sensor that can indicate if the door or window was opened and that needs to be wireless. Since there was no option in using wires around the house to connect to the base station. Initially a lot of time went in utilizing the RF sensors, however they were very unreliable source of communication. The last choice was to use Bluetooth as it allowed security in communicating information. The chip ESP-32 had already a built in microcontroller that allowed a simpler implementation that is more compact. The goal was not to use a larger circuitry with the door sensor since it is a smaller substation. The successful integration of all the components of home security system makes it a successful project.

The items that could be add on are sending information to the user via an app, and that can turn on and off the alarm from anywhere.

3.4 Conclusion

The development of Home Security System using cheaper reliable components helped built an affordable product. The multiple features of this home security system can make a person feel safe at their home. The additional features that can be added could also be a keypad to set enable and disable key. Moreover, a functionality that allows user to set a

particular sensor alarm. Lets say the user is at home and since they will be walking around it would not make sense to use a motion sensor. Thus, the door sensors just need to be activated.

The report discussed the experiences of the team making the Home Security project. Initially the report outlined the general overview of how the Home Security System is to operate. After the overview, the details of both electrical design and software design were discussed at a detail level. Finally, analyzing the results of the Home Security Project helped reveal that project has been successfully achieved.

Reviewing the code more thoroughly and making enclosures for the product could lead to commercialization of this Home Security System. This can be tested out among the neighbors first. Adding more advanced features like an app to control the security system is what the future technology should look like. This is one of the implementation of Smart Homes to be able to control the security remotely.

References:

NA. (2020, March 18). How PIRs work. Retrieved from https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work

Raj, Aswinth (2015, Oct 24). 16x2 LCD Display Module. Retrieved from https://circuitdigest.com/article/16x2-lcd-display-module-pinout-datasheet/

Emmet (2019, Dec 15). Bluetooth on Raspberry Pi. Retrieved from https://pimylifeup.com/raspberry-pi-bluetooth/

SafeWise Team (2020, February 6). What is a Security System and How Does it Work ? Retrieved from https://www.safewise.com/home-security-faq/how-do-security-systems-work/

Kaysen, Ronda (2017, December 22). Do Home Security Systems Make Your Home Safer ? Retrieved from https://www.nytimes.com/2017/12/22/realestate/do-security-systems-make-your-home-safer.html