Vigilo Home Security System

Nicholas J. Hernandez, Hirad Pourtahmasbi, and Mohammad A. Sharif

***Abstract*—This project explores the communication between hardware and software. The outcome of this experiment will produce an intelligent door sensor that will communicate with the user via text messaging. This product will be able to act as a security mechanism for doors and containers, notifying the owner if their area or storage has been compromised. Using a custom prototype built with a Raspberry Pi, a Logitech C270 Webcam, and other hardware modifications, a simple, yet a powerful machine is built. This computer will be able to detect signals from a modular sensor, process code, and communicate with networks using a network chip. The key pieces of hardware that will be used for this project is an Adafruit magnetic contact switch, a Raspberry Pi 3 model B+, female to female jumper wires, a soldering kit, and webcam/pi-camera. With the use of the given hardware, this project will also demonstrate the development of an application level software that can utilize the hardware specific features to communicate over a network. To further demonstrate the knowledge gained in this course, this paper will highlight presence detection using the M.A.C. address of the user's phone, and communication via SSH for easier deployment. The source code of this project can be found on GitHub.**

I. Introduction

The Vigilo Home Security System is a smart and simple way to be aware when other individuals are walking in and out of one's house. The system is built on a Raspberry Pi 3 model B+ and makes use of a Logitech C270 webcam and Adafruit magnetic door sensors. Overall the system will alert the user when the door has been opened and they are not home, they will receive a text message stating that an intruder was detected as well as a link to the image of who has entered the property.

II. Monitoring The Door

As stated in the introduction, the key component used with the Raspberry Pi which lets us monitor the door is an Adafruit magnetic door sensor.

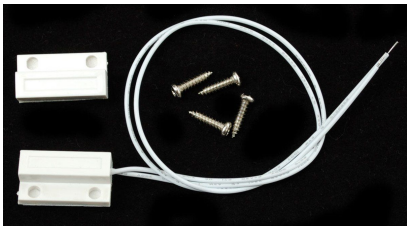

Fig. 1 Adafruit Magnetic Door Sensor

Originally these sensors are meant to be soldered directly to the GPIO pins on the Pi or used with a breadboard; however, in our case we wanted reusability so we soldered the sensor wires to jumper wires with female end caps. This way we could directly connect our sensors to the GPIO pins on the RasPi, with the ability to detach the sensor if we so choose.

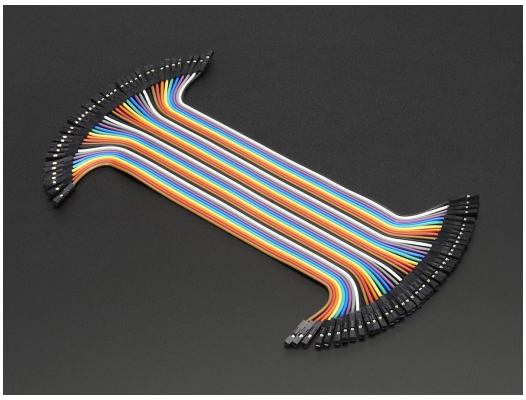

Fig. 2 Female to Female Jumper Wires

The door sensor detects whether the door is opened or closed by running a signal through the one wire in the sensor, the other wire of the sensor is grounded. Inside the sensor is a Reed switch, which closes if the magnetic end of the sensor is near. When this switch closes the circuit is complete, allowing the RasPi to detect a closed door. When the door opens, this magnetic portion of the sensor is moved away, opening the Reed switch in the sensor and allowing the Pi to detect an open door.

```
# Pi and pin setup
GPIO.setmode(GPIO.BCM)
DOOR_SENSOR_PIN = 18
GPIO.setup(DOOR_SENSOR_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

Fig, 3 Door Sensor Setup

The code above shows the import for the needed library to detect activity on the Raspberry Pi's GPIO pins. We set the mode to indicate that our numbering for the GPIO pins is based off the *Broadcom SOC Channel* number given to the pins. We've connected our sensor to pin 18, so that is what we indicate. We then set up a channel for our pin, and indicate it as an input channel. The GPIO.PUD_UP portion is accounting for the fact that when we connect something to our GPIO pin the signal may initially read as undefined, this part of the library allows us to set an initial value and keep the signal as binary values 0 or 1.

```
isOpen = 0
oldIsOpen = None

while True:
    oldIsOpen = isOpen
    isOpen = GPIO.input(DOOR_SENSOR_PIN)
    # print(isOpen)

    if (isOpen and (isOpen > oldIsOpen)):
        print("Door opened!")
        if not is_home():
            if take_picture():
                url = upload()
                send_text(url)
            else:
                print("Unable  to take a picture")
        else:
            print(individual + " is home, no text will be sent")


    time.sleep(0.1)
```

Fig. 4 Security System Main Loop

This portion of the code is the main portion of our python script which tracks the door. We keep track of whether the door is open as well as what the previous state of the door was in the last loop. The reason we keep track of these two states is due to the fact that we want to know when the door has been opened, meaning the door was closed and now it appears to be open. The value for an open door is 1 where as the value for a closed door is 0. We loop infinitely checking the state of the door each time, as soon as we have a case where our old state was 0 and our new state is 1 we know that the door has been opened.

III. Presence Detection

A feature we thought would be a great addition to the security system was that of presence detection. What exactly are we referring to? Well, it seems pointless to notify the user that someone's opened the door if that person is home. So we wanted to check whether the user was home or not before we send out the notification.

When taking into consideration how we were going to keep track of the user, the first thing that came to mind was geolocation. If our Pi could track the user's phone at all times it would know whether the user was home or not; however, this would require us creating an app for the phone to connect to our Pi and share the geolocation of the user.

After trying to think of ways we might know a user was home, we started to look at it from a networks perspective. We all have smartphones that automatically connect to the WiFi when we get home, so if we are able to have to Pi scan the network we can know that the user is home or not. The way we keep track of the user's device is via the device's unique MAC address. If we find this address connected to the network, bingo, the user is at home.

Once the idea of how we would go about keeping track of the user was set in place, now came the implementation in code.

There's a useful command line tool known as ARP scan, which gives us the ability to scan the network for current connected devices. With the use of the *subprocess* library in python, we're able to not only run the command line tool but check the output as well.

```
# This function returns true if the mac address registered is scanned
# on the wifi, false otherwise.
def is_home():
    global network
    global last_scan
    # Check if this is the first run or if the last network scan occured
    # more than five minutes ago, if so run a new scan
    if network is None or time.time() - last_scan >= 300:
        network = subprocess.check_output("sudo arp-scan -l", shell=True)
        network = network.decode("utf-8")
        last_scan = time.time()
        print(network)
    # a timer to make sure the arp-scan was finished
    # time.sleep(30)
    if address in network:
        print(individual + " is home")
        return True
    else:
        print("No one is home")
        return False
```

Fig. 5 Presence Detection

We've encapsulated the key functionality of presence detection within a callable function named *is_home()*. This function runs a scan of the network using the ARP scanner

command line tool and checks the output of it. The way python's *subprocess.check_output()* function returns our output as encoded bytes. This is the reason that we decode the data to UTF-8, which is a standard string format.

To know whether the user is home, we simply check whether the user's device MAC address is in the returned output. In the first if statement, we check whether the network has not been scanned yet, or if the last scan was done within the past five minutes. We arbitrarily chose this number because we wanted a time limit that wasn't too short (i.e. seconds) or too long (i.e. 15 mins, 30 mins, etc.). We figured 5 minutes was a good rescanning window so we stuck with it.

```
pi@raspberrypi:~ $ sudo arp-scan -l
Interface: wlan0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9 with 256 hosts (http://www.n
)
10.0.0.1        fc:51:a4:03:67:c1       (Unknown)
10.0.0.155      00:71:cc:70:2f:17       (Unknown)
10.0.0.72       cc:2d:b7:1b:3d:4a       (Unknown)
10.0.0.150      88:71:e5:99:9d:a0       (Unknown)
10.0.0.232      00:e0:4c:11:13:b0       REALTEK SE
10.0.0.254      fc:51:a4:03:67:c2       (Unknown)
10.0.0.175      f8:62:14:3c:fb:52       (Unknown)
10.0.0.215      80:e6:50:09:7a:a0       (Unknown)
10.0.0.21       64:b5:c6:70:ee:b2       (Unknown)

0 packets received by filter, 0 packets dropped by
Ending arp-scan 1.9: 256 hosts scanned in 4.041 se
conded
```

Fig. 6 ARP Scan Output

The figure above shows the output that is generated by the

ARP scanner tool. A list of devices are returned along with their corresponding MAC addresses. Converting this from a byte object into a string lets us simply check if our MAC address is in the overall string, if that is the case then we know that the user is home.

The ARP scan only takes a few seconds since home networks tend to not have hundreds of devices connected to them. This would be an issue if someone wanted to use the system on a cabinet at work, since a company may have hundreds to thousands of machines connected to the network. The user will be alerted, it will just take an extended period of time for the scan to complete.

IV. Taking A Picture Of The Intruder

Another feature that we thought would be a good addition to our security system was the functionality to text the user a picture of possible intruders. Since Twilio requires a paid subscription for picture messages we decided to host the images using a popular image sharing website called Imgur to host our images. Once the image is hosted on Imgur we then can text the user the image URL.

In order to implement this process, we must first implement the picture taking capabilities. To do this, we can use another command line tool known as *fswebcam*. After installing this command line tool, the user can check to see if their webcam is connected to the RasPi by using *lsusb* command on the terminal. That being said, we are ready to implement the *take_picture()* function.

```
# This function takes a picture of an intruder, returns true if successful
# and false otherwise
def take_picture():
    picture = subprocess.run("fswebcam " + picture_path, shell=True, check=True)
    return os.path.isfile(picture_path)
```

Fig. 7 Twilio Dashboard

This function uses the pre-defined location to take a picture and it returns true if the picture exists or false otherwise.

We used the Imgur API to upload an image after we created a developer account. The following is a piece of code that will take a file from the user's file system and upload it to Imgur. This requires the user to go to the developer tools of Imgur and create an application. After doing so, we can configure the Imgur client.

```
imgur_id = configurations.get('Imgur', 'CLIENT_ID')
imgur_secret = configurations.get('Imgur', 'CLIENT_SECRET')
imgur_access = configurations.get('Imgur', 'ACCESS_TOKEN')
imgur_refresh = configurations.get('Imgur', 'REFRESH_TOKEN')
imgur = ImgurClient(imgur_id, imgur_secret, imgur_access, imgur_refresh)
imgur_config = {
    'album': None,
    'name':  'Intruder!',
    'title': 'Intruder!',
    'description': 'Possible Intruder'
}
```

Fig. 8 Imgur Setup

After setting up the client, we can write the *upload()* function that uploads an image from the pre-defined location to Imgur and returns the url.

```
def upload():
    image = imgur.upload_from_path(picture_path, config=imgur_config, anon=True)
    os.remove(picture_path)
    return image["link"]
```

Fig. 9 Imgur Upload

The code above is used to post an image at the pre-defined location to imgur using the imgur API. The method returns the url of the uploaded image. If this call fails, the return value of the function will be *None*.

V. Notifying The User

In order to have a effective security system, it is crucial to notify the owner of any detected intrusions. While there are many different ways of implementing notifications, the Vigilo Home Security system utilizes the Twilio API to send text messages to the user.

The Twilio API provides ways of communicating through different channels, including email, text, and phone calls. Due to the nature of this product, the Vigilo team decided that a text message would be the best suited means of communicating. Fortunately, the Twilio API has a trial tier which allows for texting to one phone number. To utilize this API, a Twilio account is needed. After creating the account, keeping track of the account id and token is crucial. This information can be found via the dashboard.

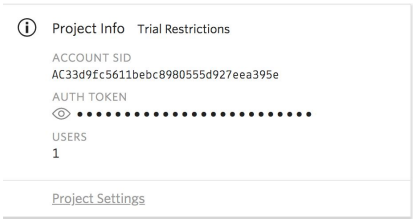

Fig. 10 Twilio Dashboard

It is also important to create a Twilio phone number. Fortunately, the trial version allows for one programmable phone number.

Now that the credentials are stored and the Twilio account has been setup, it is time to configure the Twilio client.

```
twilio_account = configurations.get("Twilio", "SID")
twilio_token = configurations.get("Twilio", "TOKEN")
txt_to = configurations.get("Twilio", "TO")
txt_from = configurations.get("Twilio", "FROM")
twilio = Client(twilio_account, twilio_token)
```

Fig. 11 Twilio Client Setup

The function that is responsible for sending the text message takes in a URL that is supplied from the Imgur upload. This allows the user to be able to quickly view a picture of the intruder.

```
# This function sends an alert to the registered phone number
def send_text(url):
    text = "Intruder detected! Check out the intruder at " + url
    message = twilio.messages.create(to=txt_to, from=txt_from, body=text)
```

Fig. 12 Twilio Send Text

The final step of implementing the Vigilo Home Security system as been complete.

VI. Remote Connection And Deployment

In order to be able to deploy the Vigilo system conveniently, it is crucial to be able to connect to the device remotely. Otherwise, deployment of the machine will require a monitor, a keyboard, and a mouse. To set up SSH connection to the RasPi, switch to root and set the password of the 'pi' account by typing *sudo -i* and *passwd pi* respectively. Enter the password twice and keep note of it.

The next step in the process of setting up remote connection is to traverse to *Preferences - Raspberry Pi Configuration - Interfaces - SSH*. Click on the "Enabled" button and head back to the terminal. Type *ifconfig* to display your devices IP Address.

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:84:17:5c  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 32  bytes 1917 (1.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32  bytes 1917 (1.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.126  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 2601:646:8f00:61b8::9346  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::323f:af32:3a79:d26  prefixlen 64  scopeid 0x20<link>
        inet6 2601:646:8f00:61b8:fda4:3531:3e81:bb  prefixlen 64  scopeid 0x0<global>
        ether b8:27:eb:d1:42:09  txqueuelen 1000  (Ethernet)
        RX packets 19374  bytes 9758884 (9.3 MiB)
        RX errors 0  dropped 4  overruns 0  frame 0
        TX packets 7179  bytes 1051245 (1.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig. 13 ifconfig output

Keep track of the address listed after "inet". After this, Using your personal you can SSH remotely into the Vigilo security system by typing *ssh pi@<IP Address>* and password in the terminal. After doing so, you are able to start the Vigilo Security System by

VII. What We Learned

One of the first things that we learned was soldering wires together can be a bit tedious. One of us had experience with through hole soldering before, but that differs from wire to wire soldering. Another issue that had come up when we were soldering was that we didn't have a stand with clamps to hold the wires in place, so we had one of us hold the wires together while another soldered (which was partly problematic). In the long run we got the soldering job done and out of the way.

Another thing we learned was that although some ideas may sound extremely complicated, the toughest part is turning an idea into an implementation. Once we were able to wrap our heads around how we were going to approach the problem, it was a matter of finding the right tools to get the job done. With the right libraries and APIs, the headache of writing a vast number of lines of code was taken away.

It was nice to look at computer networks from a different perspective with this project. We tend to live each day without really thinking about and appreciating the infrastructure of computer networks and how it makes our daily connectivity work so well. With this project we were able to look at a computer network as a tool, we were able to use it to our advantage by using the knowledge gained from class. Without sufficient knowledge about computer networks, it would have been tough to realize we could implement presence detection without needing geolocation. The project has let us look at the knowledge we've gained throughout the course of this networks class as invaluable, and we may use it as a tool in a future project.

VIII. Moving Forward

Overall this project has proved to be both fun and rewarding. None of us had really written code to run on a Raspberry Pi before and we had definitely not written code that made use of hardware the way we did in this project. Not only did we get to write code that made use of hardware, we got to experience the stress of soldering which was an interesting experience.

Unlike a lot of other projects, the beauty of this one is that it's not only practical but it's extendable. There are a lot of other functionalities that we can add into our security system long after this class is over.

One cool feature that we could add would be to text the Pi asking "Who's home?" and get a response listing out who may be there. A way to do this would be to keep a list of individuals and devices mapped to them to check for when we use the ARP scanning tool.

Another possible addition would be to turn the security system into a full fledged smart home monitoring system. We could have sensors on windows and doors, to always be able to check the state of the house. There are honestly a large number of possibilities when thinking about how we can build on this project, and we look forward to finding out what they are.