ELEC 291 SECTION L2D
PROJECT 2
Home Security with Remote Control Capabilities

Lab Section: L2D	**Group #: G18**	**Group's Lab Bench #s: 4C & 5C**
Student Names	**Student Numbers**	**Contribution (%)**
Kenton Ma	32689151	16.66
Linda Munisi	22535158	16.66
Luvian Wang	16939134	16.66
Sirine Trigui	46443157	16.66
Yixin Zhao	49427140	16.66
Xi Chu	17353137	16.66

Contribution summary:

The following is the contribution for each individual member towards the completion of the project:

Kenton Ma: Managed the project as a group leader. He also established the local connection between the Raspberry Pi and the local area network.

Linda Munisi: Design and code the main website with HTML and CSS. Also organised the hardware wiring.

Luvian Wang: Implemented the logic (in Arduino) to detect unwanted guests in the house. Also design and code the login page with javascript.

Sirine Trigui: Established a communication from the Arduino to the Raspberry pi. This was done via serial port using python scripts,contributed in designing and testing the logic of the security system and created the fritzing schematic.

Yixin Zhao: Design and code the logic of the arduino that controls the sensors, and the security in general. Also initiated the setup for the webcam camera used for security.
Xi Chu: Design and code the login page for the website using HTML, CSS and PHP. Also improved the implementation of the main page.

B. Introduction and motivations:

Our project deals with the implementation of a home security system with remote capabilities. In this project, we used the pre-existing home security systems at homes to imitate and get an idea of how the home security should be and how it should function. The main objective was to create a fully functioning home security system that would assist the user in detecting intruders and also having their information in case such a situation occurs.. In addition, the system would be controlled remotely by the user. Meaning the user would be able to control the security settings form whenever they are, example from work through a web page. The components used in this project include:

- 1 Raspberry Pi 3
- Arduino UNO
- 1 Mini webcam -USB 2.0,300k Pixel
- 1 Mini servo
- 2 Ultrasonic sensors
- 1 LM35 temperature sensor
- 1 RGB LED
- 1 green LED
- 1 3x4 keypad
- 1 Piezo buzzer.

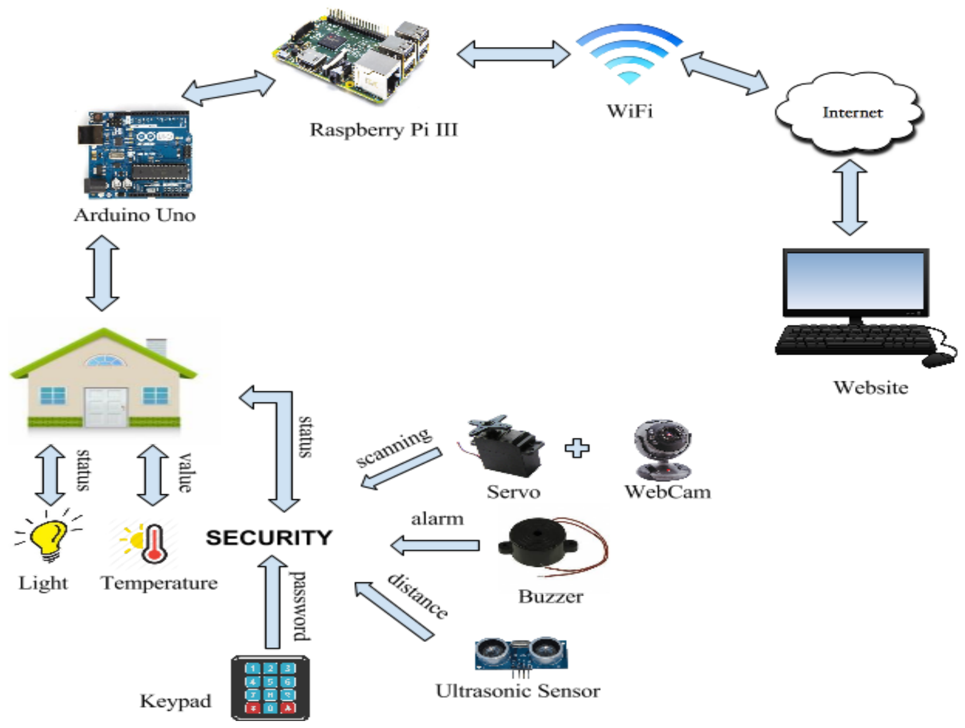
The new components that were purchased for this project were the Raspberry Pi 3, ($69.95) and the mini webcam($14.80) which altogether summed up to $95 including taxes.

The sensors that were used in the project include the temperature and the ultrasonic sensors. The function of the temperature sensor is to display the temperature of the house through the website. This would be useful in situations like if a fire occurs in the house, then the user would realize since the temperature would be abnormally high. The ultrasonic sensors' function were used to detect the motion of the door or the window. This is useful in situations where an intruder enters through the window, then these sensors would detect the object and the alarm would go on. An alternative would be to use a proximity sensor which would detect the temperature and also capture the graphics of the object sensed but purchasing this would go way beyond the budget. Then we have the LEDs, which are used for the lights of the house and also an indicator for the keypad.

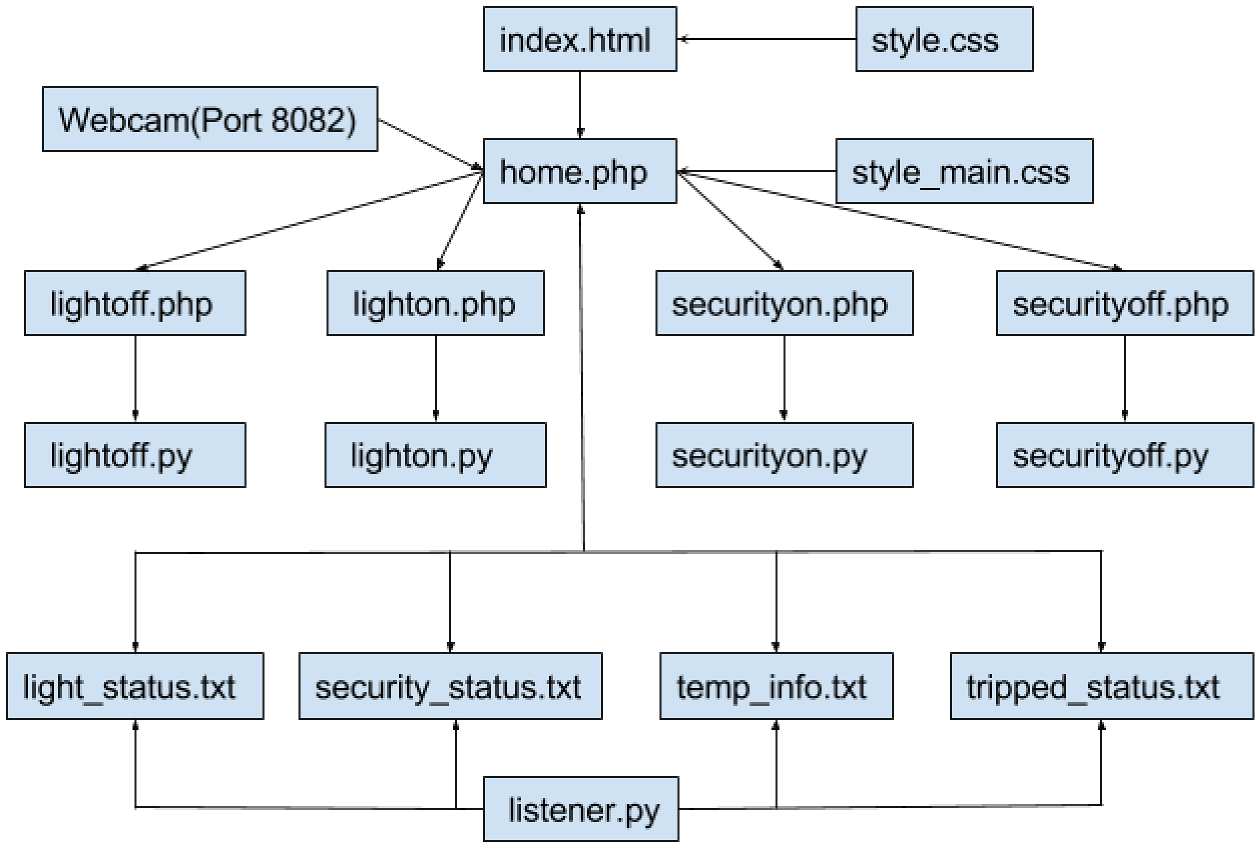
Then altogether, the sensors were used with the webcam and the raspberry pi to control the logic of the security system.

C. System diagrams:

Hardware System Diagram



Software System Diagram

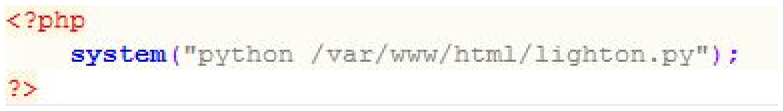


D. Project Description:

Website to Raspberry Pi (RBPi) to Arduino Communication:

For this project, we used Raspbian as our operating system. To start off making the web server, we have downloaded the Apache and PHP packages. First, we made a simple PHP file that contained 2 buttons called *home.php*. We use php because it allows us to run scripts on the web server that is being hosted by the RBPi. These buttons were created using the *button* html tag. One button is to turn ON an LED and the other button is to turn it OFF. This was the preliminary step we took in order to proceed in adding additional features such as controlling the security features of our home. After creating the buttons, they should appear in a browser at *http://localhost/home.php* (make sure the Apache package is installed). When pressing the ON button on , it will invoke another PHP file named *lighton.php*. This PHP file will execute a python script named *lighton.py*. This python script will open the serial and send serial input to the

Arduino via a USB connection to turn ON the LED. The same process is done for the OFF button. The file containing the 2 buttons was implemented using a click function. The click functions sends requests for the *lighton.php* or *lightoff.php*. The *lighton.php* and *lightoff.php* were implemented by using a simple system call to run the python script (see Figure 1.0).



```
<?php
    system("python /var/www/html/lighton.py");
?>
```

Figure 1.0 Invoking Python Scripts via PHP

We had troubles learning how to write the python script and getting it to work on our local website i.e. *home.php*. We found some helpful resources that gave us a basis to our python implementation. The basic outline of the python script is to open the serial port, then send the requested data through the serial port[1]. We write to the serial special values e.g. '3' and Arduino will read from serial and enter into a case statement containing this special value and do the corresponding action e.g. digitalWrite(LEDpin, LOW). When we tried to run the scripts through a web browser it did not work, but running it through the terminal did. To resolve this, we needed to disable the Serial option under Menu->Preferences->Raspberry Pi Configuration->Interfaces[2]. Upon rebooting the RBPi, we were able to turn the LED connected to the Arduino ON and OFF via the buttons. The SECURITY ON/OFF buttons were implemented in a similar fashion with the only difference being that the python code is sending a different special value through the serial. A better approach would have been to invoke 1 PHP file upon clicking a button. At this point, we have the LIGHT ON/OFF and SECURITY ON/OFF buttons working. The direction of communication that has been implemented so far is Website->RBPi->Arduino.

Arduino to RBPi to Website Communication:

- **Data communication between arduino and Raspberry Pi:**

Upon learning on how to use Python script and how to communicate with the Raspberry Pi, we came across an example that sends a string from arduino to raspberry every 2 seconds:

```
void setup(){
Serial.begin(9600);      }
void loop(){
Serial.println("Hello Pi");
delay(2000);
}
```

After we run python 2 on raspberry, we typed the following:

```
import serial
ser= serial.Serial('/dev/ttyACM0',9600)
```

The first argument – /dev/ttyACM0 is the name for the USB interface used. To find out the port name, we need to run this command in the pi terminal without Arduino plugged in:

```
ls /dev/tty*
```

Then plug in the arduino and run the same command again and an additional name will be added that did not exist before. The additional one is the name that will be used for the port in the code, which in our case was ACM0.

The second argument – 9600 is the baud rate and should match with what you set in the Arduino program.

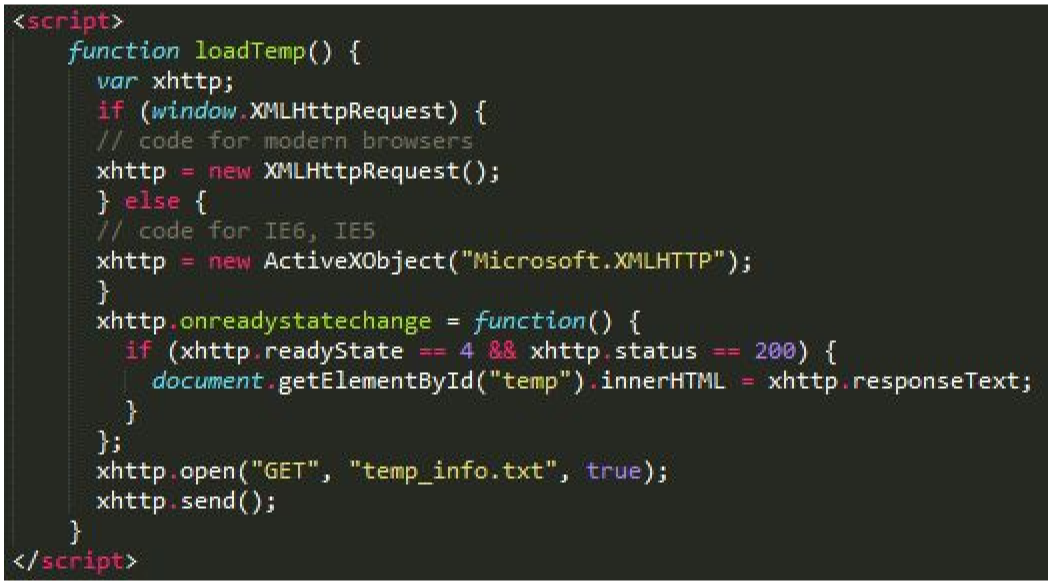
The while loop is to listen for messages from the Arduino.

```
while 1:
ser.readline()
```

After trying the sample code, we actually tested our sensors(temperature and ultrasonic) and we sent their values from Arduino to raspberry pi using the process described above.

- **From RBpi to website:**

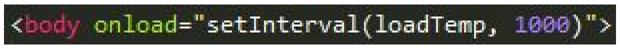
Next, we implemented the communication from RBPi->Website. For the RBPi code we have a listener called *listener.py*, which will be constantly reading from the serial port. Again, we use special values to determine what action to take. In this case, we will be writing to text files. The reason we will be writing to text files is because we will be reading from the text files and displaying the information in the text files on the website. For example, if we recently turned the light on from the Arduino, the Arduino will send a special value e.g. '-100' and then write to *security_status.txt* the word "ON". We tested this part by sending serial data from the Arduino to the RBPi and check to see if data has been written to the text files. A problem we had with writing to the files is that we opened all of the text files and writing to the corresponding text files after going through the *if* statements. The problem with this is that it wrote to one text file and left the other files blank. To resolve this issue, we only opened one file at a time depending on the conditions of the *if* statements. Now we are able to write Arduino's sent information to text files. Next, we will take the information in the text files and display it on the website. To do this, we will be using asynchronous JavaScript and XML (AJAX). The reason for implementing this part with AJAX is because it avoids having the user refresh the page every time they want the most recent information e.g. status of light, status of security system etc. We first started with trying to display the temperature data. A script will be used to read from our text file (see Figure 2.0)



```
<script>
    function loadTemp() {
      var xhttp;
      if (window.XMLHttpRequest) {
      // code for modern browsers
      xhttp = new XMLHttpRequest();
      } else {
      // code for IE6, IE5
      xhttp = new ActiveXObject("Microsoft.XMLHTTP");
      }
      xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
          document.getElementById("temp").innerHTML = xhttp.responseText;
        }
      };
      xhttp.open("GET", "temp_info.txt", true);
      xhttp.send();
    }
</script>
```

Figure 2.0: Script Used in Reading from Text Files

We tested this part by manually writing to *temp_info.txt*. The script will write to the element containing the id “temp” the information contained in the *temp_info.txt* file. The html tag containing this id is *<p id=”temp”>*</p>. Upon successfully testing this part, we tried to display real-time temperature data. To do this, we will be using JavaScript’s timing events (refer to Figure 3.0).



```
<body onload="setInterval(loadTemp, 1000)">
```

Figure 3.0: JavaScript Timing Events

The *onload* element tells the browser to run the timing event when the browser loads. The timing event is *setInterval*. In Figure 3.0, the *loadTemp* (see Figure 2.0) function will be invoked every 1000 milliseconds (or 1 second). In other words, the temperature information will be updated every second. To test this, we connected the LM35 to Arduino and the Arduino to the RBPi (via USB). The Arduino will be printing to the serial the temperature information and *listener.py* will be running on the RBPi and writing to *temp_info.txt*. Upon opening the website, we observed that the temperature had been successfully updated every 1 second. A similar approach was done for the light status, security status, and time of most recent breach.

Live Webcam Streaming on Website(RBPi):

The live webcam streaming was implemented by using the *iframe* html tag. This tag will be open to *http://localhost:8082/.* The port 8082 is the port where the webcam is streaming. One problem with this is that using *localhost* only streams for users on the RBPi. In other words, it will not stream the webcam feed if a user used another computer on the LAN. To resolve this we had to look at the ip of the RBPi by typing *hostname -I* into the terminal and use the that ip instead of *localhost* i.e. *http://172.20.10.5:8082/.*

Website(User Interface):

We first implemented the main page of our website. On the main page, we divided the screen into two sections. On the left side of the main page, we have a live stream video that can monitor the condition of the house through a webcam. We set the live stream to be on even when the security system is turned off. On the right side of the page, there were displays for the temperature, security and light status, together with the time of most recent breach. Then at the bottom, there was the buttons that are used to control the lights and the security system. How the display works is, when the light on button is clicked on, the green led turns on and then the light status changes to on and when off, the status changes to off. The same with the security on and off buttons. Then the the time of the recent breach displays when the alarm was last triggered so that the time information can be used to retrieve the corresponding video recorded from the webcam.

Then we realized that we should implemented a login page so that only the user who has the correct username and password can access the main page to monitor the house. We added the logic that authenticates the identity of the user in the login page's html file (*index.html*). We use javaScript to implement this logic because it was a simple interacted operation between the user and browser. We predefined a username and password pair and made the webpage check if it matches with what the user submitted, If they match, the browser will go to the main page (*main.php*), otherwise, it will show an error message at the top of the login page. However, we then realized that even if we were using this logic to prevent others to access the main page from the login page, by directly entering the URL of the main page, people could still get access to it and monitor the house. To avoid this issue, we deleted the javascript code and used php to define the inputs from the user(username and password) as global variables that can be used across multiple web pages so that whenever the browser goes to the main page, the main.php will check whether these two variables have been set correctly, if not, the browser will go back to the login page. However, as our group then decided to make the whole system running at localhost, there are no privacy issue any more, we changed them back to the javascript version.

Webcam:

To setup the webcam, we needed to install the *fswebcam* and *motion* package[4]. Following the instructions in a tutorial included in the references, we tried to get a video stream but were unsuccessful. We tried solving the problem by trying to edit the configuration files for lower quality video output and slowing the stream but it did not solve the problem. Upon trying different attempts, we tried rebooting the RBPi and it seemed to work after that.

For the webcam, it can be used to record both videos and pictures. Initially, the webcam takes pictures and stores them in a capture folder. Once the camera detects a movement, the pixel changes and it starts taking a video instead of pictures. Then the video is also recorded in the capture folder in the raspberry pi. The maximum length of the saved videos is 10 minutes. Another problem that we faced was at sometime, the webcam would crash after two minutes of streaming a video. This problem was solved by changing the configurations of the video stream. First we changed the screen size and reduced it from 640x430 to size 352x288. We realized that having a smaller screen also solves the problem of the video lag that we had. Then we also adjusted the frame rate, which is the number of pictures sent per time for the video. Decreasing this solved the crashing problem that we had. Therefore despite having a small video screen, we were able to make it very responsive and without a lag. After this, the webcam worked perfectly like we wanted it to.

Security System:

For this section, the code for each part was implemented separately then it was tested to make sure that we have a solid base.

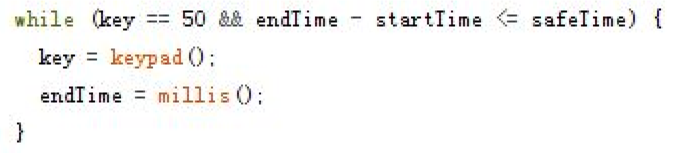
We started implementing the logic for the security system imitating it from the ones we are using in our houses. The logic is as follows:

- if any motion is detected near the door/window:

The ultrasonic sensor is triggered since the distance was decreased,the green led is on and the RGB is turned to yellow. The buzzer then sings to kindly remind the user to enter the

passcode.Within 30 seconds,the webcam in this case faces the door/window(whichever is detected opened):
*if the entered passcode is correct, the RGB led turns green for two seconds and the security system is turned off
*if an incorrect passcode is entered, then the buzzer starts beeping, RGB turns red then yellow when the buzzer sings allowing the user another chance to guess the passcode. Approximately, three chances are allowed to enter the correct passcode.
After those 30 seconds,RGB is turned red, buzzer starts beeping and the webcam starts scanning the place.
-If any motion is detected at both window and door:
Our logic is implemented in such a way that, there is a priority when both areas are triggered. The first priority is given to the window and the camera turns to the window first since that one is more likely to become an intruder. Also since the person at the door will be distracted while trying to enter the correct password in the keypad.

One problem faced earlier with this implementation was that, if the user does not press any key, the alarm is not going to beep even if the safe period is passed(20 seconds). After discussing the logic again and evaluating it, we figured out that it was because our keypad keeps waiting for an input. We fixed it just by adding a timing stamp in the while loop of the function that reads input from our keypad:



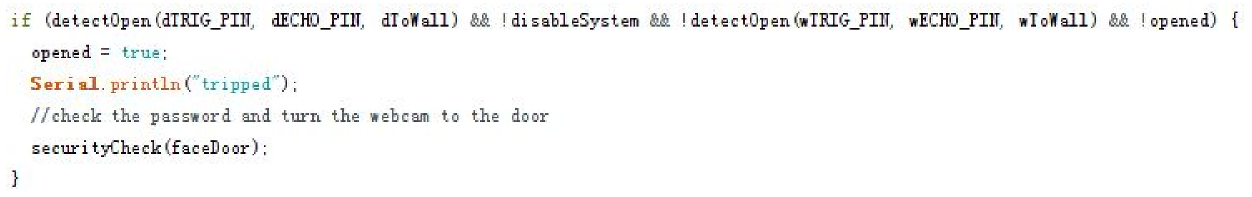
```
while (key == 50 && endTime - startTime <= safeTime) {
  key = keypad();
  endTime = millis();
}
```

So that the maximum waiting time of this program is 30 seconds then it will jump out of the while loop and start executing the next line.

For most of the functions, such as the ones for keypad, buzzer, and detect distance, we used similar logic as the one we had in the previous labs, with lots of improvements since we had to consider additional situations hence increasing the complexity of the code. For the *securityCheck()* function, if the distance that ultrasonic sensor detects is shorter that the distance between it and the wall, the system considers the door/window as open and runs *securityCheck()*. This function first turns on the light of the room, which makes it user friendly and also helps the Webcam to obtain a clear look of the possible intruder, then light up the RGB

LED(yellow) with the singing buzzer to remind the user to enter password. After the timing begins, the function prompts the servo to turn the Webcam to the door/window and start waiting for inputs from the keypad for maximum 30 seconds.

There are many flags for controlling the triggering conditions in this logic. We had many small bugs which were related to the *open* and *disableSystem* flag. This influenced the control from the website and it messed up with the light controlling. First we found that we did not set them back to *false* anywhere which made the controlling system not working with enabling or disabling the security system. After fixing that, we found out that there was a condition missing in the *if* and *else if* statement, which is the condition that *opened* flag to be *false* which means the door was closed before we run these lines:



```
if (detectOpen(dTRIG_PIN, dECHO_PIN, dToWall) && !disableSystem && !detectOpen(wTRIG_PIN, wECHO_PIN, wToWall) && !opened) {
  opened = true;
  Serial.println("tripped");
  //check the password and turn the webcam to the door
  securityCheck(faceDoor);
}
```

This made the program more stable when we control it from the website.
The only thing that might be edited is that we had to figure out a way to control the security on at the beginning(now it is by default on).The other debatable thing would having the ability to turn the security system off from the house itself but not only through the website. We thought it is better to keep our logic as it is for protection purposes.In case, the user is at work and there is something happening inn the house, the user would be able to turn it off just by clicking on the security off button, even if he/she at home(it is more efficient/faster).

D. References and bibliography:

REFERENCES:

[1]"Win32DiskImager – Raspberry Pi Projects", *Raspberry-projects.com*, 2016. [Online]. Available: http://www.raspberry-projects.com/pi/pi-operating-systems/win32diskimager. [Accessed: 08- Apr- 2016].

[2]"Communication between Arduino, RaspberryPi and Server", *YouTube*, 2016. [Online]. Available: https://www.youtube.com/watch?v=M2-nXbi3qmk. [Accessed: 08- Apr- 2016].

[3]"Servidor web con Raspberry Pi", *CoyaN.es*, 2012. [Online]. Available: http://www.coyan.es/Blog/2012-06/servidor-web-raspberry-pi/. [Accessed: 08- Apr- 2016].

[4]"Raspberry Pi IP Webcam Tutorial", *YouTube*, 2016. [Online]. Available: https://www.youtube.com/watch?v=7TQ1IBqreUY. [Accessed: 08- Apr- 2016].

[5]"Fritzing", *Fritzing.org*, 2016. [Online]. Available: http://fritzing.org/projects/raspberry-pi-3. [Accessed: 08- Apr- 2016].

[6]"Connect Raspberry Pi and Arduino with Serial USB Cable - Oscar Liang", *Oscar Liang*, 2013. [Online]. Available: http://blog.oscarliang.net/connect-raspberry-pi-and-arduino-usb-cable/. [Accessed: 08- Apr- 2016].

[7]"Super Mario theme song w/ piezo buzzer and Arduino! | PrinceTronics", *Princetronics.com*, 2016. [Online]. Available: http://www.princetronics.com/supermariothemesong/. [Accessed: 08- Apr- 2016].

[8]"adafruit/Fritzing-Library", *GitHub*, 2013. [Online]. Available: https://github.com/adafruit/Fritzing-Library/blob/master/parts/USB%20TTL%20Serial%20Cable.fzpz. [Accessed: 08- Apr- 2016].

[9]"Raspberry Pi Downloads - Software for the Raspberry Pi", *Raspberry Pi*, 2016. [Online]. Available: https://www.raspberrypi.org/downloads/. [Accessed: 08- Apr- 2016].

[10]2016. [Online]. Available: https://www.rpelectronics.com/sen06053b-mini-webcam-usb-2-0-300k-pixel.html. [Accessed: 08- Apr- 2016].

[11]"Raspberry Pi 3 Model B", *Canakit.com*, 2016. [Online]. Available: http://www.canakit.com/raspberry-pi-3-model-b.html. [Accessed: 08- Apr- 2016].

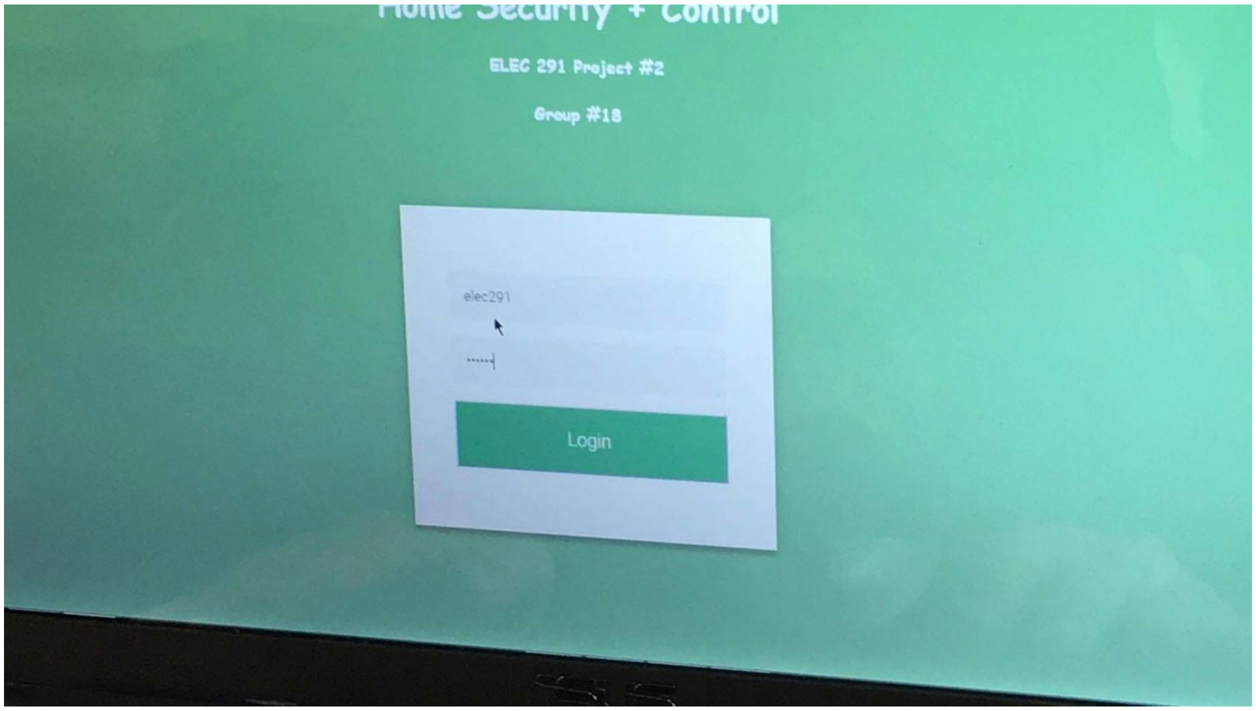
BIBLIOGRAPHY:

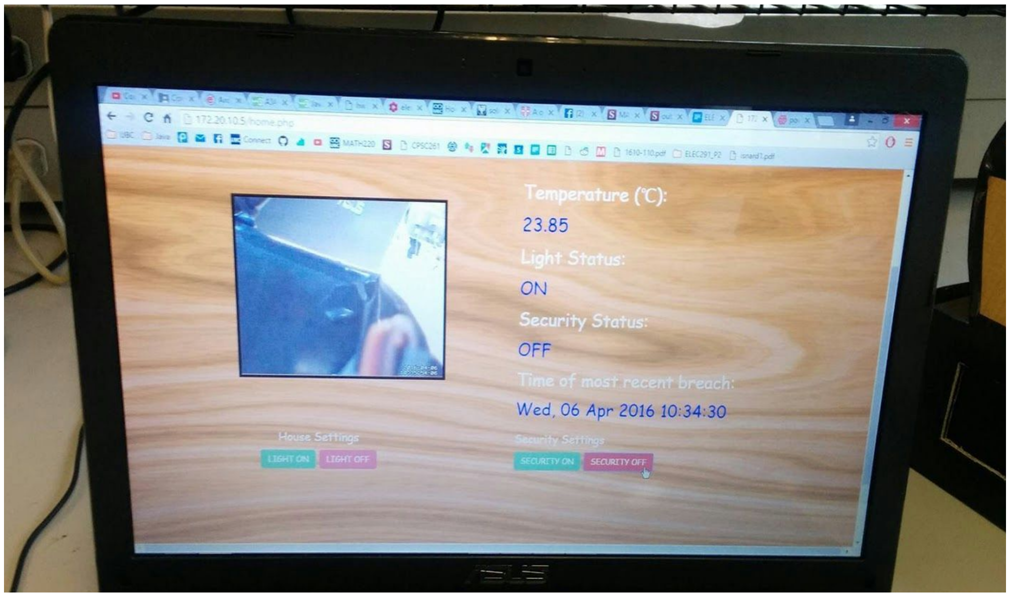
- Arduino file - house.ino containing the code that sends information to the serial port
- Fritzing file that shows the schematic of the circuit and how the components are connected.
- Index.php file that contains the code for the login page.
- Style.css file that contains the styles and settings used in the login page.
- Home.php file that contains the code for the main page of the website.
- Style_main.php file that contains the styles and settings for the main page of the website.
- Lightoff.php file that contains button controls for switching off the green LED
- Lightoff.py file containing the script that actually communicates with the serial in Arduino to switch off the green LED.
- Lighton.php file that contains button controls for switching on the green LED
- Lighton.py file containing the script that actually communicates with the serial in Arduino to switch on the green LED.
- Securityoff.php file that contains button controls for disabling the security system.
- Securityoff.py file containing the script that actually communicates with the serial in Arduino to disable the security system.
- Securityon.php file that contains button controls for enabling the security system
- Securityon.py file containing the script that actually communicates with the serial in Arduino to enable the security system.
- Listener.py file containing the script that communicates with the serial to recieve values that will be stored in the text files.
- Light_status.txt that stores information from the serial that will be displayed on the website, for the status of the green LED

- Security_status.txt that stores information from the serial that will be displayed on the website, for the status of the security.
- Temp_info.txt that stores information from the serial that will be displayed on the website, for the temperature of the house.
- Tripped_status.txt that stores information from the serial that will be displayed on the website, which is the time of most recent breach on website.

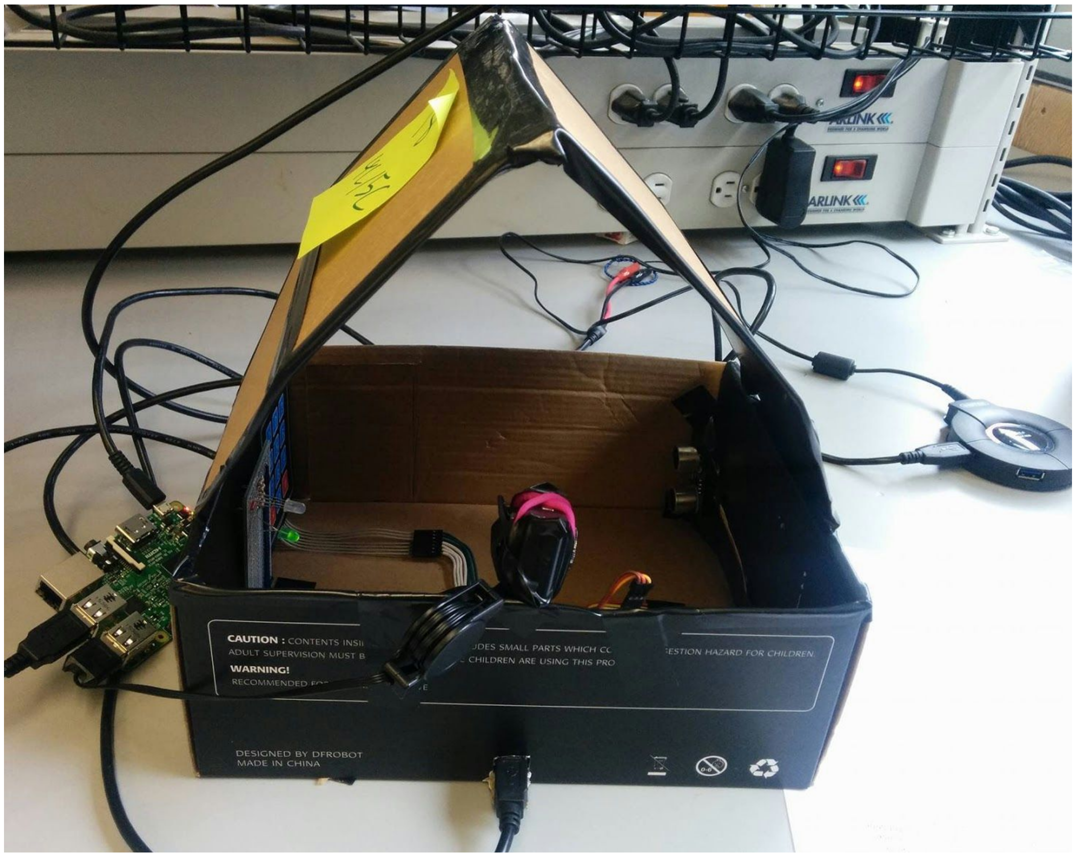
Appendix A – Project pictures

Login page:

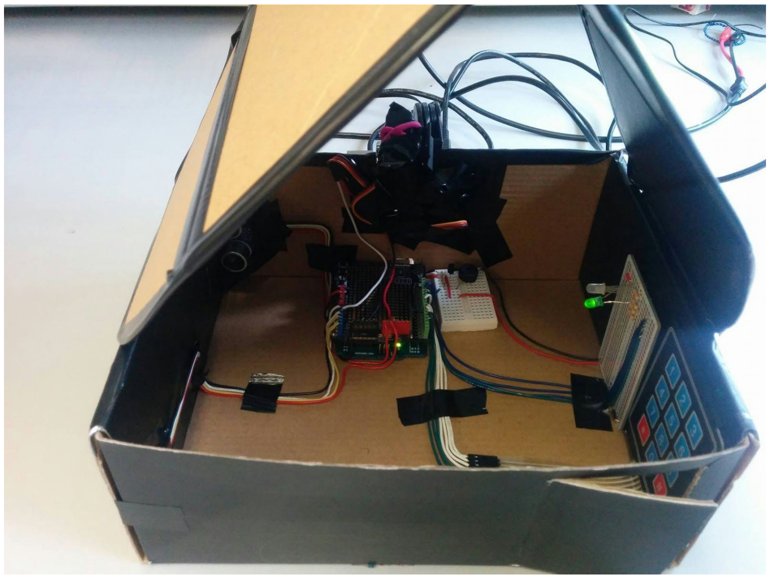


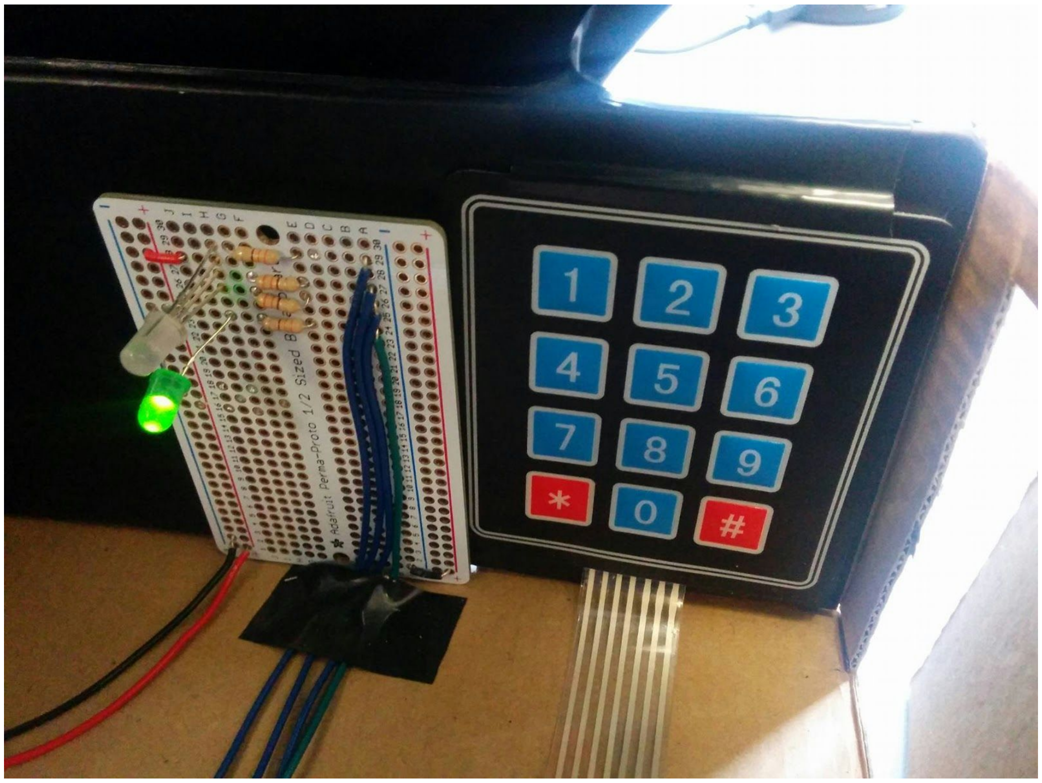
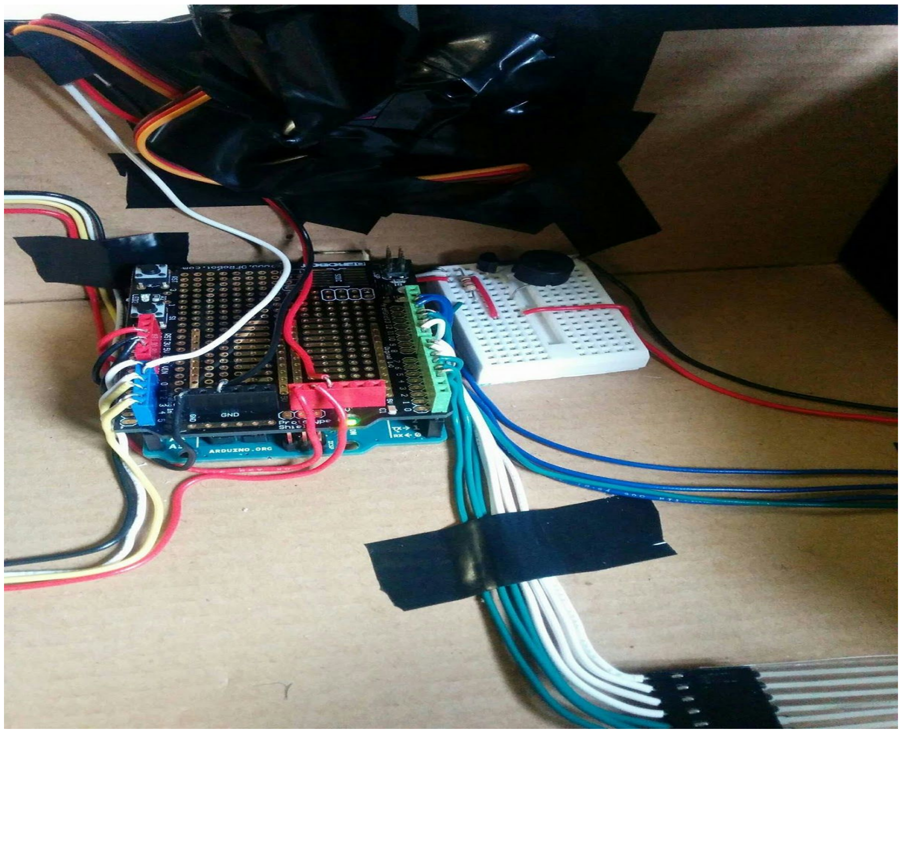

Temperature (℃):
23.85
Light Status:
ON
Security Status:
OFF
Time of most recent breach:
Wed, 06 Apr 2016 10:34:30

house:



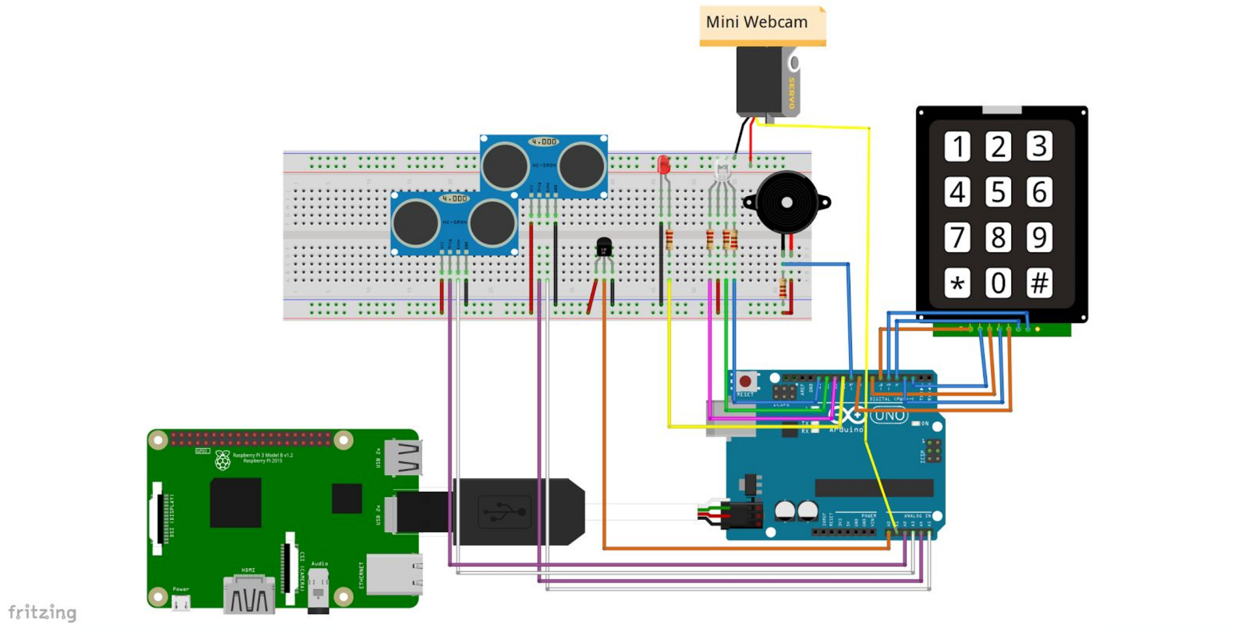
components:




Adafruit Perma-Proto 1/2 Sized
1
2
3
4
5
6
7
8
9
*
0
#

ARDUINO.ORC
GND

Appendix B - Fritzing

The fritzing schematic for the circuit is shown below:



Appendix C – Arduino Code

```
//-------------buzzer-------------------------
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_C4  262
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_C5  523
#define forBuzzer 9
int underworld_melody[] = {
  NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
  NOTE_AS3, NOTE_AS4, 0,
  0
};
//Underwolrd tempo
int underworld_tempo[] = {
  18, 18, 18, 18, 18, 18, 18, 3
};
int song = 0;
//--------------------keypad---------------------
int g = 0;
byte h = 0;
byte v = 0;  //variables used in for loops
const unsigned long period = 50; //little period used to prevent error
```

```
unsigned long kdelay = 0;      // variable used in non-blocking delay
const byte rows = 4;           //number of rows of keypad
const byte columns = 3;          //number of columnss of keypad
const byte Output[rows] = {8, 7, 6, 5}; //array of pins used as output for rows of keypad
const byte Input[columns] = {4, 3, 2}; //array of pins used as input for columns of keypad
int password[5] = {1, 2, 3, 4, 14}; //password
int input[5] = {0, 0, 0, 0, 0}; //default input
//------------ultrasonic sensorWindow---------------
#define wTRIG_PIN A3
#define wECHO_PIN A2
int wToWall = 12; // the distance from the window sensor to the wall
//------------ultrasonic sensorDoor---------------
#define dTRIG_PIN A5
#define dECHO_PIN A4
int dToWall = 21; // the distance from the door sensor to the wall
//-------------detecting distance--------------
float duration;
float distance;
int reading = 0;
//------------mark whether the door or window is open
bool opened = false;
//------------Temperature Sensor--------------
#define TEMP_PIN A0
//---------------Servo----------------
#include <Servo.h>
Servo myservo;
#define SERVO A1
int pos = 40;
int faceDoor = 125;
int faceWindow = 35;
//-----------varibles for time-----------
unsigned long startTime = 0;
unsigned long endTime = 0;
const long safeTime =  30000;
int disableSystem = 0;
//--------------------RGBLed------------
int bluePin = 11;//RED
int redPin = 13;//BLUE
int greenPin = 12;//GREEN
//--------------------SingleColorLed---------
int led = 10;
//-------------------the code send from Raspberry Pi--------
char action;

void setup() {
  //--------buzzer----------------------
  pinMode(forBuzzer, OUTPUT);
  //--------------------keypad---------------------
  Serial.begin(9600);
  // put your setup code here, to run once:
  for (byte i = 0; i < rows; i++) //for loop used to make pin mode of outputs as output
  {
    pinMode(Output[i], OUTPUT);
  }
  for (byte s = 0; s < columns; s++) //for loop used to makk pin mode of inputs as inputpullup
  {
    pinMode(Input[s], INPUT_PULLUP);
  }

  //-----ultrasonic sensorDoor---------------
  pinMode(dTRIG_PIN, OUTPUT);
  pinMode(dECHO_PIN, INPUT);
  //-----ultrasonic sensorWindow---------------
  pinMode(wTRIG_PIN, OUTPUT);
  pinMode(wECHO_PIN, INPUT);
  //------------Temperature Sensor--------------
  pinMode(TEMP_PIN, INPUT);
  //--------------servo------------------
  myservo.attach(SERVO);
  //------------------------RGBLed-------------------
  pinMode(redPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  //------------SingleColorLed---------
  pinMode(led, OUTPUT);
  //---------------Setting the RGB to off----------
  digitalWrite(greenPin, HIGH);
  digitalWrite(redPin, HIGH);
  digitalWrite(bluePin, HIGH);
}

void loop() {
  //read the command send from Raspberry Pi and change corresponding status
  if (Serial.available()) {
    action = Serial.read() ;
    switch (action) {
      case '0'://activate the system
        disableSystem = 0;
        opened = false;
        Serial.println("-100");
        break;

      case '1'://disable the system
        disableSystem = 1;
        opened = false;
        Serial.println("-200");
        break;

      case '2'://turn on the light
        digitalWrite(led, HIGH);
        Serial.println("-300");
        break;

      case '3'://turn off the light
        digitalWrite(led, LOW);
        Serial.println("-400");
        break;

      default:
        break;
    }
  }

  reading = 0;
  // to reduce noise, we take several sample readings and take their average
  for (int i = 0; i < 8; i++) {
    reading = reading + analogRead(TEMP_PIN);
    delay(30);
  }
  // get average of temperature samples
  reading = reading >> 3;

  // Calculate temperature from LM35
  float TemperatureC = reading * (3.3 / 1024.0) * 100;
  //calculate the effective speed of sound
  float speed = (331.5 + (0.6 * TemperatureC)) * 29 / 343;
  Serial.println(TemperatureC);

  // if the system is on, the door and windwo was not opened before, and the door is open, and
window is closed now
   if (detectOpen(dTRIG_PIN, dECHO_PIN, dToWall) && !disableSystem && !detectOpen(wTRIG_PIN,
wECHO_PIN, wToWall) && !opened) {
    opened = true;
    Serial.println("tripped");
    //check the password and turn the webcam to the door
    securityCheck(faceDoor);
  }
  //if the system is on, the window and door was not opened before, and window is open now
  else if (!disableSystem && detectOpen(wTRIG_PIN, wECHO_PIN, wToWall) && !opened) {
    opened = true;
    Serial.println("tripped");
    //check the password and turn the webcam to the window
    securityCheck(faceWindow);
  }
  //if the system is disabled then set the RGB to off, alarm to off, and put the webcam back
to the original position
  else if (disableSystem) {
    myservo.write(pos);
    Serial.println("-200");
    digitalWrite(greenPin, HIGH);
    digitalWrite(redPin, HIGH);
    digitalWrite(bluePin, HIGH);
    alarm(0);
  }
  //if window or door is open, and the alarm is on, start scan the room
  if (opened && !disableSystem)
    scan();
}

//check the input with the password
void securityCheck(int servoPos) {
  digitalWrite(led, HIGH);
  Serial.println("-300");
  digitalWrite(redPin, LOW);
  Serial.println("-100");
  digitalWrite(greenPin, LOW);
  startTime = millis();
  endTime = millis();
  sing(1);//remind the user
  sing(0);
  while (endTime - startTime <= safeTime) {//set certain time for the user to input password
    endTime = millis();
    myservo.write(servoPos);
    if (!disableSystem) {
      //if the input from the user is wrong
      if (!checkPassword()) {
        //set alarm on and turn on the red light
        digitalWrite(bluePin, HIGH);
        digitalWrite(greenPin, HIGH);
        digitalWrite(redPin, LOW);
        Serial.println("-100");
        alarm(1);
      }
      //if the input from the user is correct
      else {
        //disable the system and turn the green light on for two seconds
        disableSystem = 1;
        Serial.println("-200");
        digitalWrite(bluePin, HIGH);
        digitalWrite(redPin, HIGH);
        digitalWrite(greenPin, LOW);
        delay(2000);
        digitalWrite(greenPin, HIGH);
        //turn off the alarm
        alarm(0);
      }
        //if there is still more time, set the alarm off and remind the user to have another
attampt
      if (!disableSystem) {
        alarm(0);
        sing(1);
      }
       //if the user choose to disable the system during the safetime, it will set the alarm
off and not remind the user
      else {
        alarm(0);
        sing(0);
      }
    }
  }
  //after the safe time, if the correct password was not entered, set the alarm on
  if (!disableSystem ) {
    alarm(1);
    myservo.write(pos);
  }
}
//reminder
void sing(int s) {
  // iterate over the notes of the melody:
  song = s;
  if (song == 1) {
    int size = sizeof(underworld_melody) / sizeof(int);
    for (int thisNote = 0; thisNote < size; thisNote++) {
      int noteDuration = 1000 / underworld_tempo[thisNote];
      buzz(forBuzzer, underworld_melody[thisNote], noteDuration);
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);
      buzz(forBuzzer, 0, noteDuration);
    }
  }
}

void buzz(int targetPin, long frequency, long length) {
  digitalWrite(13, HIGH);
  long delayValue = 1000000 / frequency / 2; // calculate the delay value between transitions
  // 1 second's worth of microseconds, divided by the frequency, then split in half since
  // there are two phases to each cycle
   long numCycles = frequency * length / 1000; // calculate the number of cycles for proper
timing
  // multiply frequency, which is really cycles per second, by the number of seconds to
  // get the total number of cycles to produce
  for (long i = 0; i < numCycles; i++) { // for the calculated length of time...
    digitalWrite(targetPin, HIGH); // write the buzzer pin high to push out the diaphram
    delayMicroseconds(delayValue); // wait for the calculated delay value
    digitalWrite(targetPin, LOW); // write the buzzer pin low to pull back the diaphram
    delayMicroseconds(delayValue); // wait again or the calculated delay value
  }
  digitalWrite(13, LOW);
}

//alarm which makes noise controlled by the parameter sent to it
void alarm(int a) {
  analogWrite(forBuzzer, LOW);
  if (a == 1) {
    tone(forBuzzer, 700);
    delay(3000);
  }
  else
    noTone(forBuzzer);
}

// function used to detect which button is pressed
byte keypad() {
  static bool no_press_flag = 0; //static flag used to ensure no button is pressed
   for (byte x = 0; x < columns; x++) // for loop used to read all inputs of keypad to ensure
no button is pressed
  {
    if (digitalRead(Input[x]) == HIGH); //read evry input if high continue else break;
    else
      break;    if (x == (columns - 1)) //if no button is pressed
    {
      no_press_flag = 1;
      h = 0;
      v = 0;
    }
  }
  if (no_press_flag == 1) //if no button is pressed
  {
    for (byte r = 0; r < rows; r++) //for loop used to make all output as low
      digitalWrite(Output[r], LOW);
    for (h = 0; h < columns; h++) // for loop to check if one of inputs is low
    {
       if (digitalRead(Input[h]) == HIGH) //if specific input is remain high (no press on it)
continue
        continue;
      else    //if one of inputs is low
      {
        for (v = 0; v < rows; v++) //for loop used to specify the number of row
        {
          digitalWrite(Output[v], HIGH);  //make specified output as HIGH
            if (digitalRead(Input[h]) == HIGH) //if the input that selected from first sor loop
is change to high
          {
            no_press_flag = 0;              //reset the no press flag;
            for (byte w = 0; w < rows; w++) // make all outputs as low
              digitalWrite(Output[w], LOW);
            return v * 4 + h; //return number of button
          }
        }
      }
    }
  }
  return 50;
}

//read the input from the keypad and return it
int readKeypad() {
  delay(200);
  if (millis() - kdelay > period) //used to make non-blocking delay
  {
    byte key = keypad();  // key is set to be which button was pressed
    kdelay = millis();   //capture time from millis function

    // waiting until one of the buttons are pressed
```

```
while (key == 50 && endTime - startTime <= safeTime) {
  key = keypad();
  endTime = millis();
}

switch (key)  //switch used to specify which button
{
  // in case user pressed 1
  case 0:
    {
      g = 1;
      //          Serial.println(g);
    }
    break;
  // in case user pressed 2
  case 1:
    {
      g = 2;
      //          Serial.println(g);
    }
    break;

  // in case user pressed 3
  case 2:
    {
      g = 3;
      //          Serial.println(g);
    }
    break;

  // in case user pressed 4
  case 4:
    {
      g = 4;
      //          Serial.println(g);
    }
    break;
  // in case user pressed 5
  case 5:
    {
      g = 5;
      //          Serial.println(g);
    };
    break;
```

```
// in case user pressed 6
case 6:
  {
    g = 6;
    //          Serial.println(g);
  }
  break;

// in case user pressed 7
case 8:
  {
    g = 7;
    //          Serial.println(g);
  }
  break;

// in case user pressed 8
case 9:
  {
    g = 8;
    //          Serial.println(g);
  };
  break;
// in case user pressed 9
case 10:
  {
    g = 9;
    //          Serial.println(g);
  };
  break;
case 12:
  {
    g = 12;
    //          Serial.println("Star");
  }
  break;
// in case user pressed 0
case 13:
  {
    g = 0;
    //          Serial.println(g);
  }
  break;
// in case user pressed #
case 14:
        {
          g = 14;
          //          Serial.println("numberSign");
        }
        break;

      default:
        ;
    }
  }
  return g;
}

//get the input from the user
void getInput() {
  int i = 0;
  while ( i < 5) {
    input[i] = readKeypad();
    i ++;
  }
}

//check if the input matches the password and return true or false
boolean checkPassword() {
  int count = 0;
  getInput();
  for (int i = 0; i < 5; i ++) {
    count ++;
    if (input[i] != password[i])
      return false;
    else if ( count == 5)
      return true;
  }
}

//----------------------detect if door or window is open---------------------
bool detectOpen(int trig, int echo, int toWall) {
  reading = 0;
  // to reduce noise, we take several sample readings and take their average
  for (int i = 0; i < 8; i++) {
    reading = reading + analogRead(TEMP_PIN);
    delay(30);
  }
  //initialize the trigger pin by doing a digitalWrite() to LOW for 50ms
  digitalWrite(trig, LOW);
  delay(50);
  //start the trigger sequence by performing a digitalWrite() of HIGH for 10ms
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  //set to LOW on the same pin
  digitalWrite(trig, LOW);

  // get average of temperature samples
  reading = reading >> 3;

  // Calculate temperature from LM35
  float TemperatureC = reading * (3.3 / 1024.0) * 100;
  //calculate the effective speed of sound
  float speed = (331.5 + (0.6 * TemperatureC)) * 29 / 343;
  //Serial.println(TemperatureC);
  //obtain the echo pulse duration (in microseconds)
  duration = pulseIn(echo, HIGH);
  //get a more accurate distance value using speed of sound
  distance = duration / (2 * speed);
  if (abs(distance - toWall) > 2 && distance > 0.00) {
    return true;
  }
  else {
    return false;
  }
}

//----------------Scan function for the servo and webcam------------
void scan() {
  for (pos = 0; pos <= 140; pos += 1) {
    delay(60);
    myservo.write(pos);
  }
  for (pos = 140; pos >= 0; pos -= 1) {
    delay(60);
    myservo.write(pos);
  }
}
```

Appendix D – Website Code (HTML/PHP/CSS)

index.html

```
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css">

</head>

<script>


function validateForm()
{
var user=document.login.username.value;
var user=user.trim();
var pass=document.login.password.value;

if(user == 'elec291' && pass == '111111')
{
location.href = "home.php";
return false;
}
else
{
document.getElementById('errormessage').innerHTML="password not correct";
return false;
}
}

</script>



<body>


     <div id="title">
            <h1 id="title" class="text-center">Home Security + Control</h1>
            <h4 class="text-center">ELEC 291 Project #2</h4>
            <h4 class="text-center">Group #18</h4>
    </div>

<div class = "login-page">
  <div class="form">
   <form name="login" onsubmit="return validateForm();">
     <err><p id="errormessage"></p></err>
       <input type ="text" name="username" placeholder="username: elec291"><br>
```

```
        <input type ="password" name ="password" placeholder="password: 111111"><br>
       <button name="submit" value = "login" type="submit"\>Login</button><br>
   </form>




</div>
</div>

</body>
</html>
```

style.css

```
@import url(https://fonts.googleapis.com/css?family=Roboto:300);

.login-page {
  width: 360px;
  padding: 8% 0 0;
  margin: auto;
}
.form {
  position: relative;
  z-index: 1;
  background: #FFFFFF;
  max-width: 360px;
  margin: -50 auto 100px;
  padding: 45px;
  text-align: center;
  box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
}
.form input {
  font-family: "Roboto", sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 100%;
  border: 0;
  margin: 0 0 15px;
  padding: 15px;
  box-sizing: border-box;
  font-size: 14px;
}

.form button {

 font-family: "Roboto", sans-serif;
  background: #4CAF50;
  width: 100%;
  padding: 20px;
  color: #FFFFFF;
  font-size: 18px;
  -webkit-transition: all 0.3 ease;
  transition: all 0.3 ease;
  cursor: pointer;

}
.form button:hover,.form button:active,.form button:focus {
  background: #43A047;
}

.form err{
  color: #FF0000;
}

body {
  background: #76b852; /* fallback for old browsers */
  background: -webkit-linear-gradient(right, #76b852, #8DC26F);
  background: -moz-linear-gradient(right, #76b852, #8DC26F);
  background: -o-linear-gradient(right, #76b852, #8DC26F);
  background: linear-gradient(to left, #76b852, #8DC26F);
  font-family: "Roboto", sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

#title {
       font-family: "Rouge Script", cursive;
  text-align: center;
  padding-top: 20px;
  padding-bottom: -10px;
       color: #FFFFFF;
       padding-top:5px;
}
```

home.php

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
                                                                 <link        rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
```

```
                                                                                                  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
                                                                                                  <script
src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
        <link rel="stylesheet" type="text/css" href="style_main.css">
        <script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
        <script type="text/javascript">
            $(document).ready(function() {
                $('#lightON').click(function() {
                    var req = new XMLHttpRequest();
                    req.open("GET", "lighton.php");
                    req.onreadystatechange=function() {
                        if (req.readyState == 4) {
                            if (req.status == 200) {
                            } else {
                                alert("HTTP ERROR");
                            }
                        }
                    }
                    req.send();
                });
                $('#lightOFF').click(function() {
                    var req = new XMLHttpRequest();
                    req.open("GET", "lightoff.php");
                    req.onreadystatechange=function() {
                        if (req.readyState == 4) {
                            if (req.status == 200) {
                            } else {
                                alert("HTTP ERROR");
                            }
                        }
                    }
                    req.send();
                });
                $('#securityOFF').click(function() {
                    var req = new XMLHttpRequest();
                    req.open("GET", "securityoff.php");
                    req.onreadystatechange=function() {
                        if (req.readyState == 4) {
                            if (req.status == 200) {
                            } else {
                                alert("HTTP ERROR");
                            }
                        }
                    }
                    req.send();
                });
                $('#securityON').click(function() {
                    var req = new XMLHttpRequest();
                    req.open("GET", "securityon.php");
                    req.onreadystatechange=function() {
                        if (req.readyState == 4) {
                            if (req.status == 200) {
                            } else {
                                alert("HTTP ERROR");
                            }
                        }
                    }
                    req.send();
                });
            });
        </script>
        <!--AJAX scripts used in displaying real-time data on the website-->
        <script>
            function loadTemp() {
              var xhttp;
              if (window.XMLHttpRequest) {
              // code for modern browsers
              xhttp = new XMLHttpRequest();
              } else {
              // code for IE6, IE5
              xhttp = new ActiveXObject("Microsoft.XMLHTTP");
              }
              xhttp.onreadystatechange = function() {
                if (xhttp.readyState == 4 && xhttp.status == 200) {
                  document.getElementById("temp").innerHTML = xhttp.responseText;
                }
              };
              xhttp.open("GET", "temp_info.txt", true);
              xhttp.send();
            }
        </script>
              <script>
            function loadSec() {
              var xhttp;
              if (window.XMLHttpRequest) {
              // code for modern browsers
              xhttp = new XMLHttpRequest();
              } else {
              // code for IE6, IE5
              xhttp = new ActiveXObject("Microsoft.XMLHTTP");
              }
              xhttp.onreadystatechange = function() {
                if (xhttp.readyState == 4 && xhttp.status == 200) {
                  document.getElementById("securityStatus").innerHTML = xhttp.responseText;
                }
              };
              xhttp.open("GET", "security_status.txt", true);
              xhttp.send();
            }
        </script>
              <script>
            function loadLight() {
              var xhttp;
              if (window.XMLHttpRequest) {
              // code for modern browsers
              xhttp = new XMLHttpRequest();
              } else {
              // code for IE6, IE5
              xhttp = new ActiveXObject("Microsoft.XMLHTTP");
              }
              xhttp.onreadystatechange = function() {
                if (xhttp.readyState == 4 && xhttp.status == 200) {
                  document.getElementById("lightStatus").innerHTML = xhttp.responseText;
                }
              };
              xhttp.open("GET", "light_status.txt", true);
              xhttp.send();
            }
        </script>
        <script>
            function loadBreach() {
              var xhttp;
              if (window.XMLHttpRequest) {
              // code for modern browsers
              xhttp = new XMLHttpRequest();
              } else {
              // code for IE6, IE5
              xhttp = new ActiveXObject("Microsoft.XMLHTTP");
              }
              xhttp.onreadystatechange = function() {
                if (xhttp.readyState == 4 && xhttp.status == 200) {
                  document.getElementById("breachTime").innerHTML = xhttp.responseText;
                }
              };
```

```
              xhttp.open("GET", "tripped_status.txt", true);
              xhttp.send();
            }
        </script>
        <script>
            function runListener() {
              var req = new XMLHttpRequest();
              req.open("GET", "listener.php");
                    req.onreadystatechange=function() {
                        if (req.readyState == 4) {
                            if (req.status == 200) {
                            } else {
                                alert("HTTP ERROR");
                            }
                        }
                    }
              req.send();
            }
        </script>
    </head>
              <body    onload="setInterval(loadTemp,    1000);    setInterval(loadSec,    1000);
setInterval(loadLight, 1000); setInterval(loadBreach, 1000)">
        <div id="outermost">
            <div>
                <h2 id="title" class="text-center">Home Security + Control</h2>
                <h6 class="text-center">ELEC 291 Project #2</h6>
                <h6 class="text-center">Group #18</h6>
                <div class="text-center clockBox">
                         <iframe src="http://onlinehtmltools.com/clocks/digital-clock/clock3/"
name="iframe985426"     width="170px"     height="100px"     scrolling="No"     frameborder="0"
align="center"></iframe><br/>
                </div>
            </div>
            <div class="row">
                <div id="stream" class="col-sm-6">
                       <iframe src="http://172.20.10.5:8082/" style="border: 5px solid black;"
width="362" height="299"></iframe><br/>
                </div>
                <div id="infoBox" class="col-sm-6">
                    <p>Temperature (&#8451;): </p>
                    <p class="displayFont" id="temp"></p>
                    <p>Light Status: </p>
                    <p class="displayFont" id="lightStatus"></p>
                    <p>Security Status: </p>
                    <p class="displayFont" id="securityStatus"></p>
                    <p>Time of most recent breach: </p>
                    <p class="displayFont" id="breachTime"></p>
                </div>
            </div>
            <div class="row">
                <div class="sendBox col-sm-6 text-center">
                    <p>House Settings</p>
                             <button type="button" class="btn btn-success" id="lightON">LIGHT
ON</button>
                             <button type="button" class="btn btn-danger" id="lightOFF">LIGHT
OFF</button>
                </div>
                <div class="receiveBox col-sm-6 text-center">
                    <p>Security Settings</p>
                        <button type="button" class="btn btn-success" id="securityON">SECURITY
ON</button>
                        <button type="button" class="btn btn-danger" id="securityOFF">SECURITY
OFF</button>
                </div>
            </div>
        </div>
    </body>
</html>
```

style_main.css

```
/*NOTE: The borders are there for debugging*/
#outermost {
       background-image:
url('http://www.myfreetextures.com/wp-content/uploads/2014/10/seamless-wood-background-1.jpg');
       background-color: #474e5d;
       /*border: 5px solid black;*/
       height: 1080px;
       font-family: "Rouge Script", cursive;
       color: #FFFFFF;
       padding: 10px;
 }

#title {
       padding-top: 20px;
       padding-bottom: -10px;
}

#stream {
       /*border: 5px solid black;*/
       padding-top: 5%;
       padding-left: 15%;
}

#infoBox {
        height: 435px;
        /*border: 5px solid black;*/
        font-size: 30px;
        padding-top: 30px;
}

/*House Settings*/
.sendBox {
        /*border: 5px solid black;*/
        padding-top: 20px;
        font-size: 20px;
}

/*Security Settings*/
.receiveBox {
        /*border: 5px solid black;*/
        padding-top: 20px;
        font-size: 20px;
        text-align: left;
}

.displayFont {
        color: blue;
}

.clockBox {
        /*border: 5px solid black;*/
}
```

lightoff.php

```
<?php
        system("python /var/www/html/lightoff.py");
?>
```

lightoff.py

```
#!/usr/bin/env python

import serial
import time
import subprocess

try:
    print "Searching for Arduino..."
    dev = subprocess.check_output('ls ../../../dev/ttyACM*', shell = True)
except:
    print "Check if Arduino is connected."
    exit(1)
print dev

try:
    print "Trying to open serial port..."
    ser = serial.Serial(dev.strip(), 9600)
    time.sleep(2) #allow time for serial to open
    ser.write('3')
    print "Successfully connected to serial port."
except:
    print "Cannot open serial port."
```

lighton.php

```
<?php
        system("python /var/www/html/lighton.py");
?>
```

lighton.py

```
#!/usr/bin/env python

import serial
import time
import subprocess

try:
    print "Searching for Arduino..."
    dev = subprocess.check_output('ls ../../../dev/ttyACM*', shell = True)
except:
    print "Check if Arduino is connected."
    exit(1)
print dev

try:
    print "Trying to open serial port..."
    ser = serial.Serial(dev.strip(), 9600)
    time.sleep(2) #allow time for serial to open
    ser.write('2')
    print "Successfully connected to serial port."
except:
    print "Cannot open serial port."
```

securityon.php

```
<?php
        system("python /var/www/html/securityon.py");
```

```
?>
```

securityon.py

```
#!/usr/bin/env python

import serial
import time
import subprocess

try:
    print "Searching for Arduino..."
    dev = subprocess.check_output('ls ../../../dev/ttyACM*', shell = True)
except:
    print "Check if Arduino is connected."
    exit(1)
print dev

try:
    print "Trying to open serial port..."
    ser = serial.Serial(dev.strip(), 9600)
    time.sleep(2) #allow time for serial to open
    ser.write('0')
    print "Successfully connected to serial port."
except:
    print "Cannot open serial port."
```

securityoff.php

```
<?php
        system("python /var/www/html/securityoff.py");
?>
```

securityoff.py

```
#!/usr/bin/env python

import serial
import time
import subprocess

try:
    print "Searching for Arduino..."
    dev = subprocess.check_output('ls ../../../dev/ttyACM*', shell = True)
except:
    print "Check if Arduino is connected."
    exit(1)
print dev

try:
    print "Trying to open serial port..."
    ser = serial.Serial(dev.strip(), 9600)
    time.sleep(2) #allow time for serial to open
    ser.write('1')
    print "Successfully connected to serial port."
except:
    print "Cannot open serial port."
```

listener.py

```
 #!/usr/bin/env python

import serial
import time
import subprocess

while 1:
    dev = subprocess.check_output('ls ../../../dev/ttyACM*', shell = True)
    ser = serial.Serial(dev.strip(), 9600) #open serial port
    reading = ser.readline() #read from serial
    reading = reading.strip() #strip newline
     if reading != '-100' and reading != '-200' and reading != '-300' and reading != '-400' and
reading != 'tripped': #if not special values, then it's a temperature value
           ft = open('temp_info.txt', 'w') #open temp_info.txt to store temperature values from
arduino
        ft.write(reading) #write temperature value
        ft.close()
    elif reading == '-100':
         fs = open('security_status.txt', 'w') #open security_status.txt to store current status
of home security (ON/OFF)
        fs.write("ON") #write current security status to "ON"
        fs.close()
    elif reading == '-200':
         fs = open('security_status.txt', 'w') #open security_status.txt to store current status
of home security (ON/OFF)
        fs.write("OFF") #write current security status to "OFF"
        fs.close()
    elif reading == '-300':
         fl = open('light_status.txt', 'w') #open light_status.txt to store current light status
in home (ON/OFF)
        fl.write("ON") #write current light status to "ON"
        fl.close()
    elif reading == '-400':
         fl = open('light_status.txt', 'w') #open light_status.txt to store current light status
in home (ON/OFF)
        fl.write("OFF") #write current light status to "OFF"
        fl.close()
    elif reading == 'tripped':
           ftr = open('tripped_status.txt', 'w') #open tripped_status.txt to store time of most
recent breach
        ftr.write(time.strftime("%a, %d %b %Y %H:%M:%S")) #write time of most recent breach
        ftr.close()

    print reading #for debugging
```