ELEC 3848: Integrated Design Project

Section 2B, 2016

香 港 大 學

THE UNIVERSITY OF HONG KONG

Group 20

iResidence 2020: Smart Home Final Report

GUPTA Sidhant

SINGHAL Rohak

CHAN Tin Ming

RIAZ Zain

Abstract:

iResidence is our vision of the homes of the future. More than just an automated home, iResidence changes the way its residents interact with their household objects. We believe that elegance and efficiency go hand in hand. Thus, our smart home aims at integrating easy to use and practical interfaces, built with the user in mind- with features such as a Smart Door Lock, A Security Camera with Motion Detection and Intelligent Lighting. The most attractive aspect about iResidence would be its central control panel, which is the Smart Mirror. Mirrors are essentially very personal objects; they are something which everyone uses daily, sometimes even unknowingly. Thus, they naturally are the best place to receive all your personal updates and key information about iResidence. The user gets notified about the data collected around iResidence: Status of the Home Door Lock, Camera Feed from the motion Sensor and various other statistics such as the temperature and humidity inside iResidence. The smart door lock can be accessed either through the residents' fingerprint or a pin-code, which is entered on a touch screen keypad by a guest. Automatic Fan Speed and Intelligent Lighting increases the residents' comfort, and Bluetooth control to set their respective intensities ensures convenience. A security camera can capture an image whenever a sensor detects motion in the backyard of the iResidence and sends it over to the Smart Mirror. But we believe that all these comforts must be accompanied by a sustainable and clean solution. Thus, we have also incorporated a Smart Solar Panel with a light tracker to power iResidence in a green and eco-friendly manner.

1. Project objectives:

1. To enhance the surveillance by a smart door lock with both pin access and fingerprint access.
2. To improve the household security by a security camera which will capture an image when motion is detected.
3. To design and build an automatic fan system which can control the fan speed as per the temperature, humidity, and pressure.
4. To develop an Intelligent Light control systems which adjust the luminance by the daylight and the people in the room.
5. To establish a control system with Bluetooth which allows the communication between different function modules.
6. To make a renewable energy source which can provide a stable DC power supply to all service modules.

2. Project Deliverables:

1. Smart Door Lock

Function	• Unlocks the door when the correct access code is entered on the touch screen. • Unlocks the door when a stored fingerprint is matched. • Displays a welcome message for the user. • Closes the door automatically.
Input	• Fingerprint Scanner (Adafruit 751) • TFT screen with touch (FT6206)
Output	• Servo connected to door – TowePro SG90 • Signal to Control Panel given -Serial/I2C

2. Backyard Security Camera

Function	• Uses a Motion Sensor system to take a photograph of the backyard if there is any motion in the backyard. • Displays the photograph on the central control panel.
Input	• Camera Module (Image Processing Based) • Camera Module (Adafruit 397)
Output	• Photograph File to Control Panel – Serial/I2C

3. Temperature Control and Weather Data

Function	• Controls the speed of a fan based on temperature values inside the house • Set the target temperature via Bluetooth. • Weather Data (temperature, pressure etc.) is displayed on central Control Panel.
Input	• Temperature and Pressure Sensor – HTU21DF System • Bluetooth – HC04 Serial over Bluetooth.
Output	• Fan Voltage Control (Analog Control) via OPAMP LM358 • Output data to central Control Panel – I2C/Serial

4. Lighting Management System

Function	• Controls the lighting intensity based on the lighting outside. • Smart web based used to turn on and turn off the lights.
Input	• Photo resistor • Web server for communication.
Output	• LED lights (0-12V) – Analog control through a MOSFET to adjust for voltage differences.

5. Solar Panel Charger with Light Tracking

Function	• The light tracking system points the solar panel towards the point of maximum light intensity. • The solar panel charges the emergency battery of the smart home.
Input	• Photo resistor on light tracker • Power from Solar Panel
Output	• Power connection to battery from panel. • Feedback system controls Servo Motor – MG995

6. Control Panel/Smart Mirror System

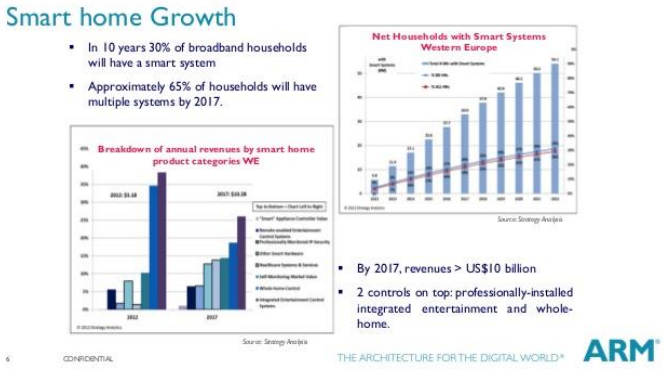
Function	• Displays the data on a web-page or mobile platform. • Allows for control of the individual components of the smart home via an app: a) Smart Lighting and Temperature Control via Homebridge. b) Traffic and News Data from Web API. c) Display of images from motion sensor security system. d) Use of Siri/voice commands to control smart home components.
Input	• Smart Door Lock • Security Camera • Weather Control System
Output	• Outputs data onto a screen which is placed behind a one-way mirror. • **Optional** output to web-server via Wi-Fi using ThingSpeak Platforms.

3. Project Team:

Role	***Name***	***Responsibility***
Project Coordinator	GUPTA Sidhant *(BEng – CE)*	Sidhant is the project coordinator. • Oversee Project Competition • Security Camera • Bluetooth Integration • Smart Door Lock • Integration of all other components to one central system.
Project Team	SINGHAL Rohak *(BEng – CE)*	In charge of developing: • Control Panel – smart mirror • Optional Web Server Integration • Integration of all other components to one central system
Project Team	RIAZ Zain *(BEng – EE)*	In charge of developing: • PID controlled fan – temperature sensor with Bluetooth temperature adjust. • PID controlled lighting system
Project Team	CHAN Tin Ming *(BEng -EE)*	In charge of developing: • Solar Light Tracker • Solar Charging System • Assembly and Finish

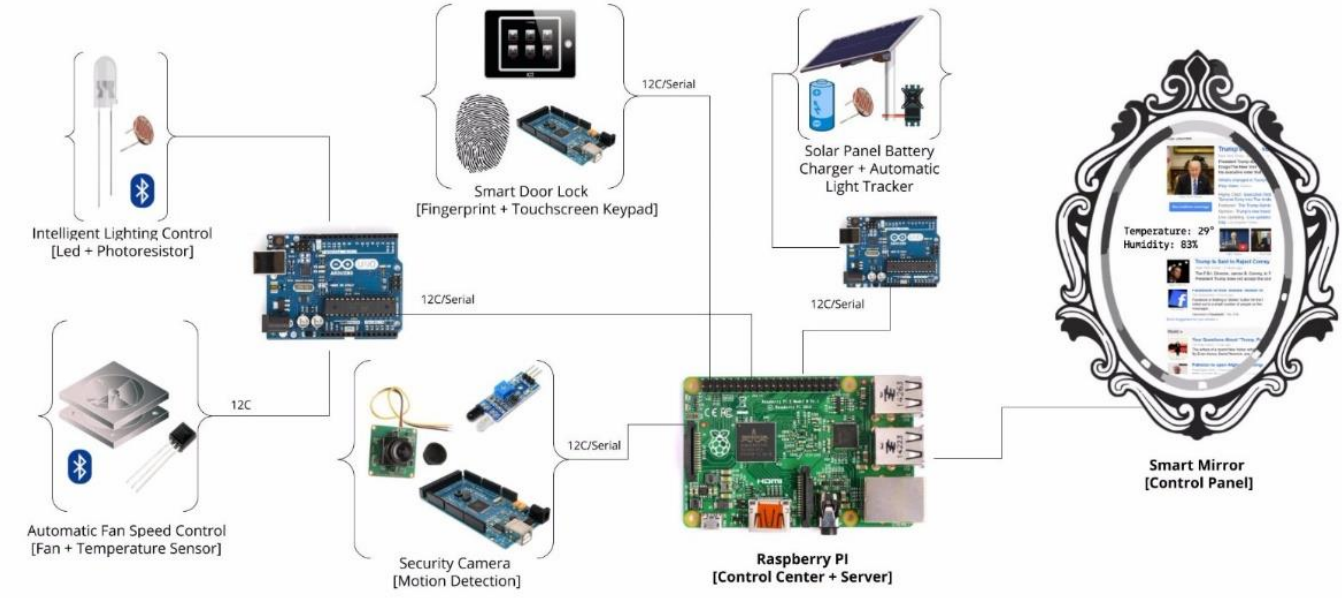
4. Project Background:

Housing is an essential to all people. Along with the improvement of a human's living standards; we have higher expectations from our house. Hence, the idea of a smart home is born. Smart House is defined as a convenient home setup where appliances and devices can be automatically controlled remotely from anywhere in the world using a mobile or another networked device. A smart home has its devices interconnected through the Internet, which control functions such as security access to the home, temperature, lighting, and home theater (1). It soon becomes popular since it was introduced to the market, and become a trend in modern society. As per ABI Research, 1.5 million home automation systems were installed in the United States by 2012(2). Our goal, as engineers is to evaluate and discover the secret behind smart home technology.



One of the leading technologies in the Smart home sphere is the wireless control of all appliances. With the advanced technology around us, we can communicate our smartphone, smart TV with other household appliances through Bluetooth or Wi-fi. So, we can control different electronic devices through our smart home, even if we are not at home. Also, smart homes have lots of automated control. It will sense the environment around the house, such as temperature, humidity, luminosity, etc., and the appliances will automatically run after it discovers a suitable environment. The biggest problem of the smart home is that its control may seem to be complex for those who are not familiar with technology. Users need to deal with the complex interface of the control panel. Hence in our project we will set up a Bluetooth control between all appliances and our control panel so that we can control our function modules wireless. Also, we will make our functional devices automatically run under suitable environment. We also plan to solve several existing issues with smart homes. To address the problem, we will develop a user-friendly interface for our users. With the interface, users can easily control different appliances; real-time conditions can also be observed on the control panel so that users can monitor the situation easily.

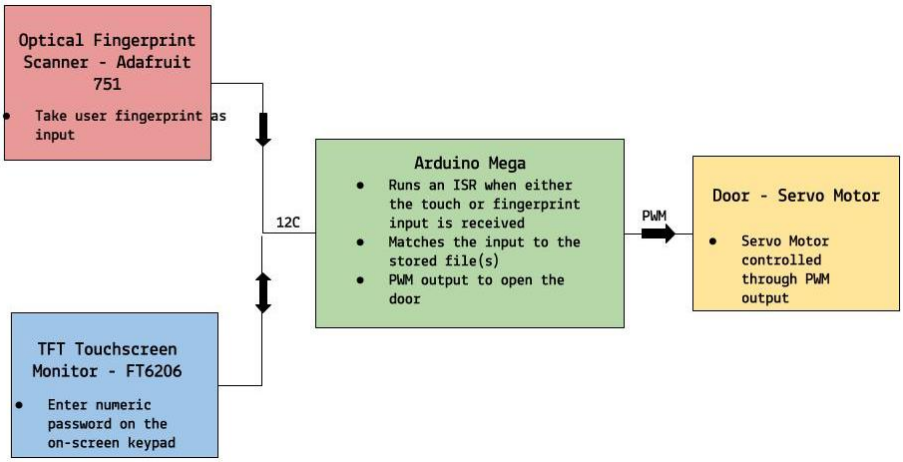
5. Project Overview and Milestones:



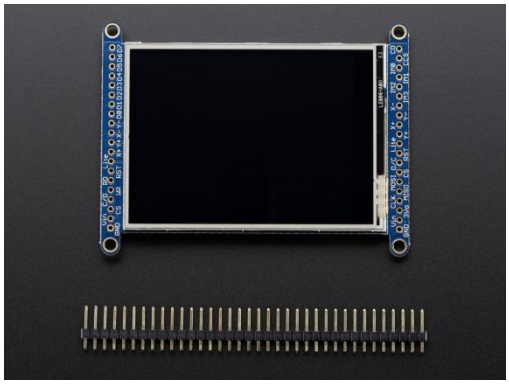
[Diagrammatic summary of our project layout]

1. Smart Door Lock

Fingerprint scanner- Adafruit 751, and TFT (Touch) monitor – FT6206 connected to an Arduino Mega. If either of the inputs is received, ISR is run. I2C is used as communication protocol here. Fingerprint compares the current fingerprint to the stored one and calls Arduino on successful match using digital signal (0/5V). Within the ISR the servo is given PWM output through Arduino pins and to open the door if the input is correct. Serial/I2C connection to central control panel.



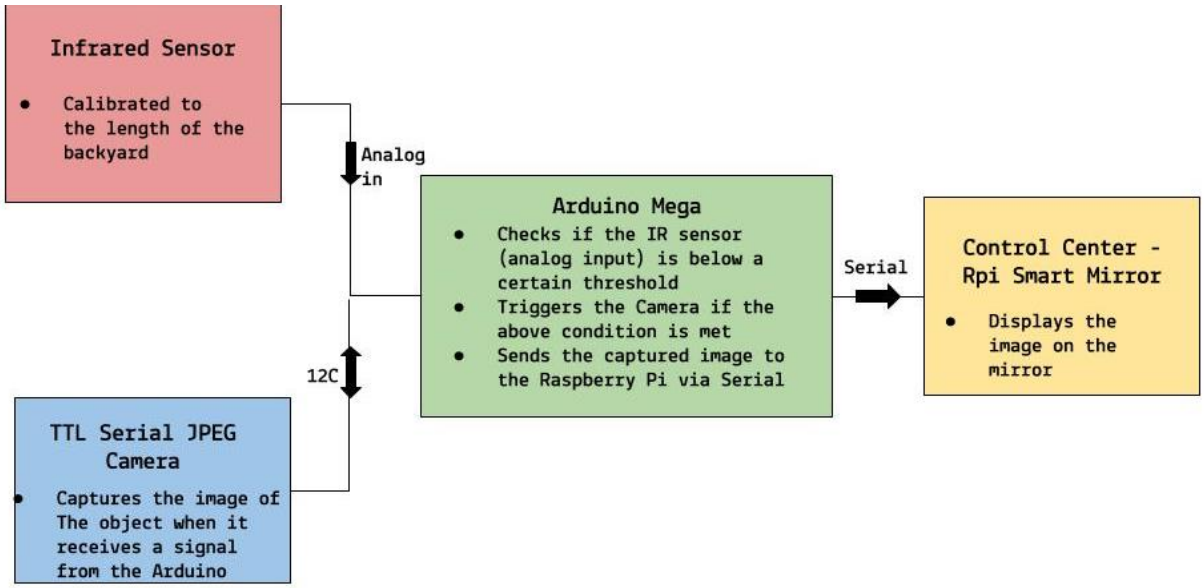
(Diagrammatic Flowchart)



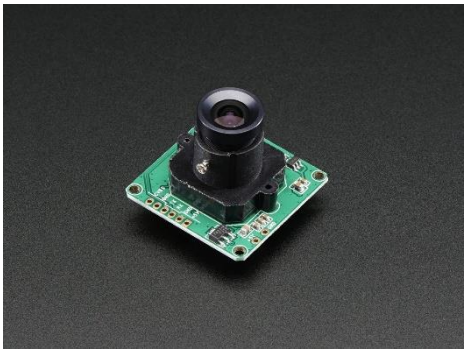
TFT (Touch) monitor – FT6206 & Fingerprint Scanner - Adafruit 751

2. Security Camera

IR sensor is calibrated for the length of the backyard. If any object comes in the way, the analog value (0-1023) map of the sensor changes, triggering the camera (via Arduino Mega) to take a photograph. Arduino Mega and camera use TTL Serial communication. The camera to be used is an Adafruit 397 module. The Mega sends the photo to the central control panel via Serial or I2C where it is displayed.



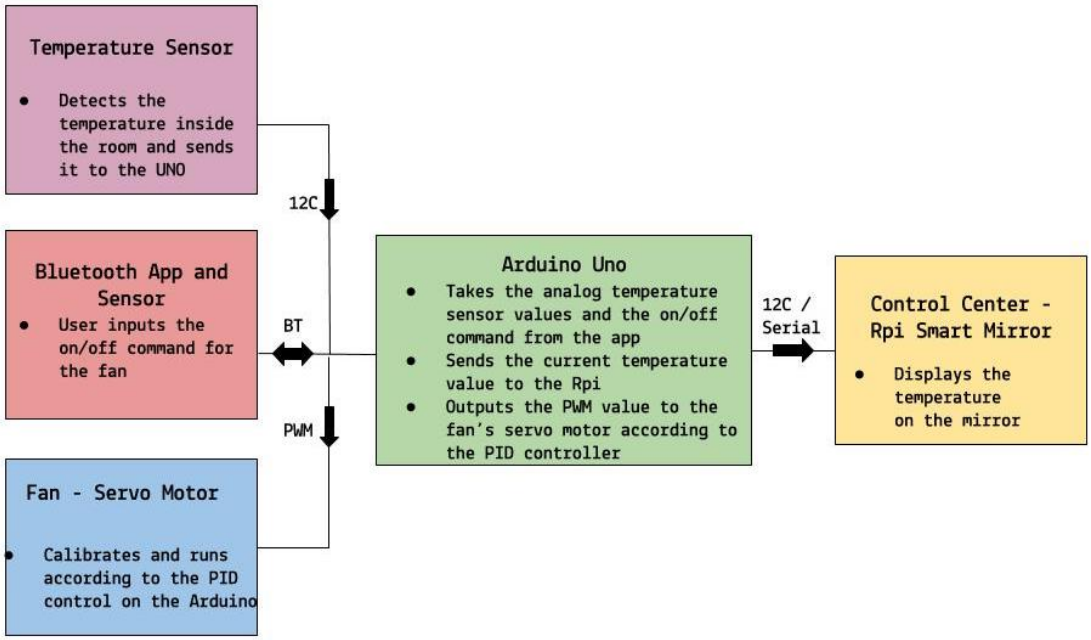
(Diagrammatic Flowchart)



(Adafruit 397 Camera Module)

3. Temperature Control and Weather Station

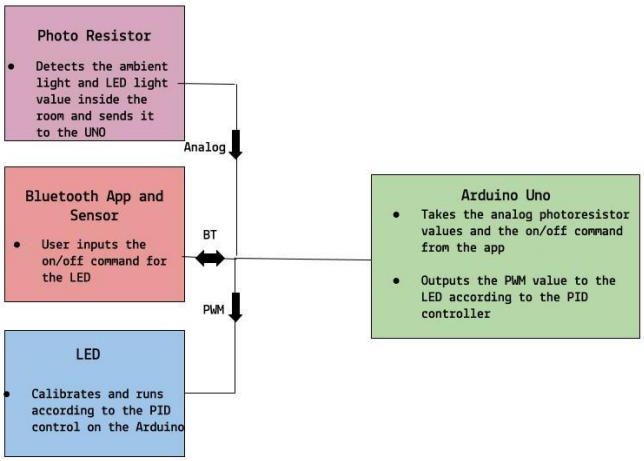
Temperature sensors communicate to Arduino UNO via I2C and Bluetooth is used to set the required temperature for the PID. The PID is run on the Arduino UNO. The fan is connected in parallel with a 470µF capacitor and 1N4001 diode via MOSFET IRF540 to a 12V supply. This allows control of the fan speed via the Arduino.



(Diagrammatic Flowchart)

4. Lighting Control System

Photo resistor sends analog data to the Arduino UNO (0-1024) on INPUT_PULLUP mode. PID is used to correct the lighting intensity inside the house accordingly. The system is designed so that the lights are turned on at night and off in full sunlight. Serial over Bluetooth (HC-04 module) connected via the Arduino UNO controls the ON and OFF state of the system.

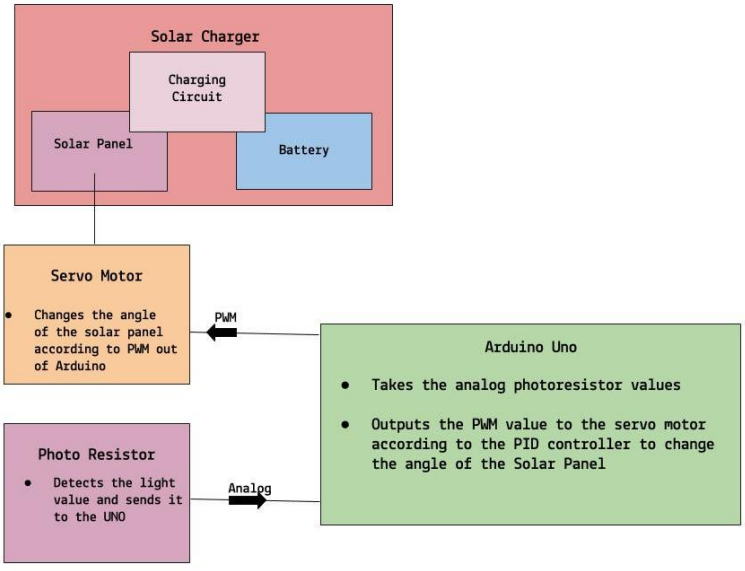


(Diagrammatic Flowchart)

(Light Dependent Resistor)

5. Solar Panel with Light Tracking

Solar Panel connected to the battery via a complex charging circuit. The light tracker uses feedback loop to find the area of maximum light intensity. PWM signal is used to control the servo motor. Arduino UNO is the preferred processing board.

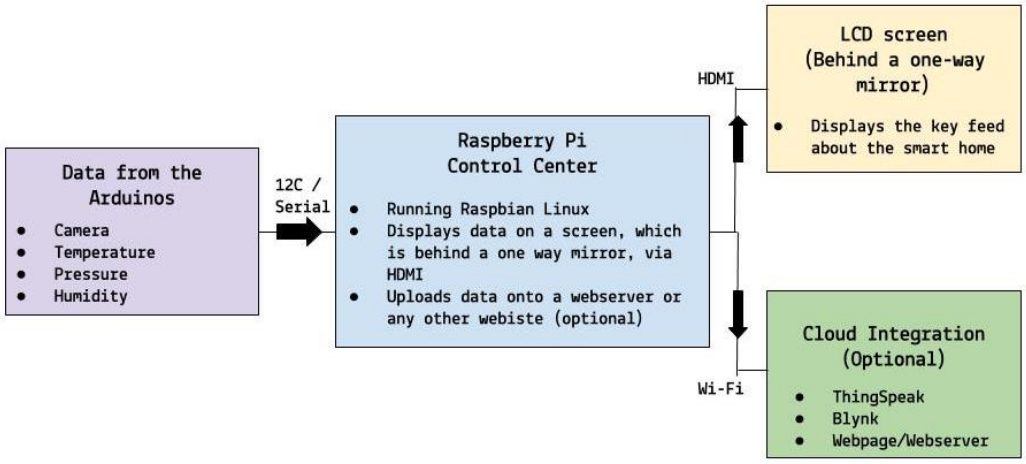


(Diagrammatic Flowchart)

6. Control Panel Smart Mirror (Optional: Web capability)

Uses a Raspberry Pi computer to communicate with all the previously mentioned systems. The data is then displayed on a screen which is placed behind a one-way mirror, hence making a smart mirror system for the home. HDMI is used to communicate with the screen. NOOBS OS is run on the Pi. We can also throw this data onto an online monitoring platform using *ThingSpeak.*

- This data can be displayed on a webpage so that it can be accessed universally by the users of the home.
- Using *ThingSpeak* we can map this data online and generate useful analytics to help the users increase efficiency:
  - Socket programming on the Pi via python or java is used to send data out.
  - MATLAB on the *ThingSpeak* platform is used for analytical work.
- Use of Homebridge and Siri to use iOS 10 to control smart home components.



(Diagrammatic Flowchart)

7. Equipment & Budget

Equipment	**Cost (HKD)**
Fingerprint Sensor	300
Raspberry Pi	300
Cloud Services	50
Mirror (One Way)	50
HDMI Screen	200
Home Building Material	100
Total	1000

8. Milestone Completion Record:

All tasks were completed on time. Since the project presentation was one week after our proposed week of completion, the final week was spent on making the smart home neat and presentable.

Prototype Design Cycle	Task Owner	Week	6	7	8	9	10
R&D	Whole Team						
Material Preperation							
Breadboard Design							
Design Specification							
Project Design							
1. Intelligent Door Lock	GUPTA Sidhant						
Design							
Implementation							
Testing							
2. Smart Security Camera	SINGHAL Rohak						
Design							
Implementation	GUPTA Sidhant						
Testing							
3. Temperature Control	RIAZ Zain						
Design							
Implementation							
Testing							
4. Lighting Control	RIAZ Zain						
Design							
Implementation	SINGHAL Rohak						
Testing							
5. Control Panel	SINGHAL Rohak						
Design							
Implementation							
Testing							
5. Solar Power and Tracker	CHAN Tin Ming						
Design							
Implementation							
Testing							
Complete Set	Whole Team						
Assembly							
Touch Up of Components							
Complete Prototype Test	Whole Team						
Functional							
Performance							
Final Touch Up							

7. Design details, Implementation, Prototyping:

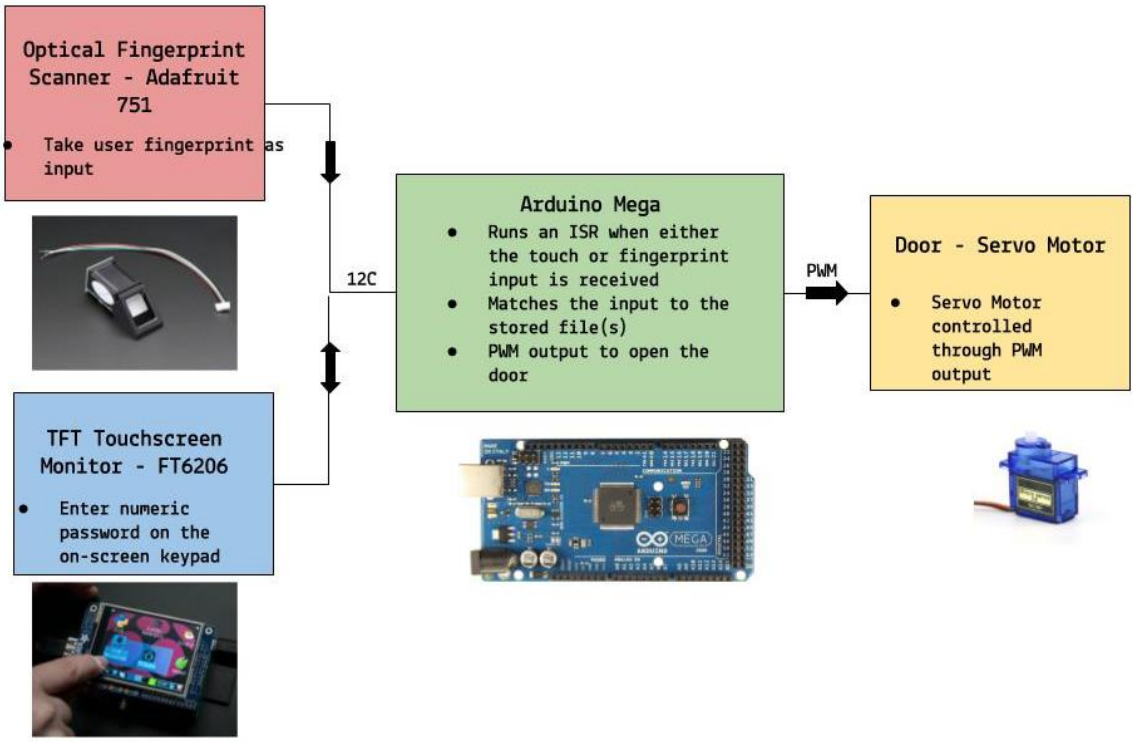
1. Smart Door Lock:

1.1 Design Concept:

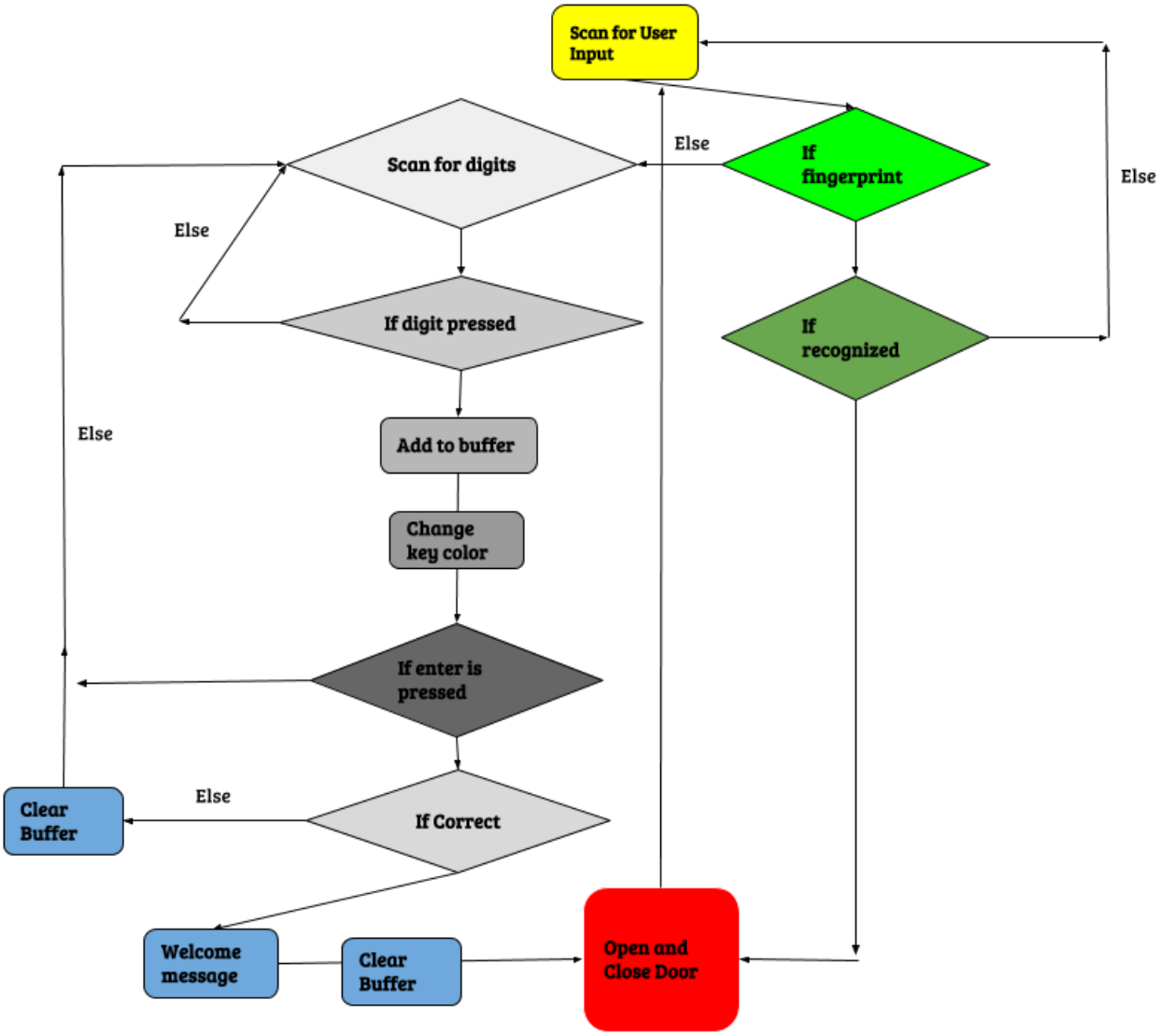
The design of the smart door lock had several requirements that had to be kept in mind. These include:

- Use of fingerprint scanner for home owner.
- Use of keypad for guests.
- Keypad must have any digit entry but four-digit password.
- Welcome message for guest if successful.
- Door must open and close automatically.
- Security system must not be active during door opening.

To make this design possible, we decided to use a servo motor to control the door of our smart home. The fingerprint scanner and TFT LCD were connected to an Arduino Mega. The programming paradigm would be such that if either of them turned a certain variable to HIGH, then the door would open and close. Further, the keys on the keypad must change color when pressed allowing the user to know which keys they have previously pressed. Hence there is a need to re-draw the keys that have been pressed. Also, to collect the data of the "number" typed by the user we need a buffer. Hence a buffer variable is created to store the current net user input. This works by adding the currently entered digit to ten times the previous sum. Now, in the situation that the passcode is not correct, the keys simply refresh once the enter key is pressed. In the situation that the right entry code is typed, the door is opened, a welcome message is displayed and the door then closes. The security system is now turned back on.

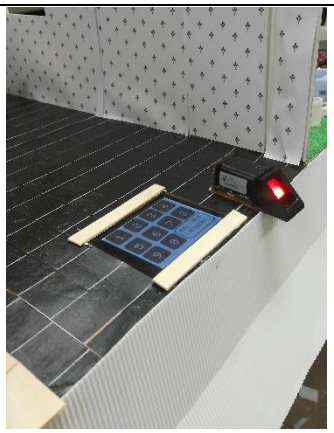


(Electronic Design Flowchart)



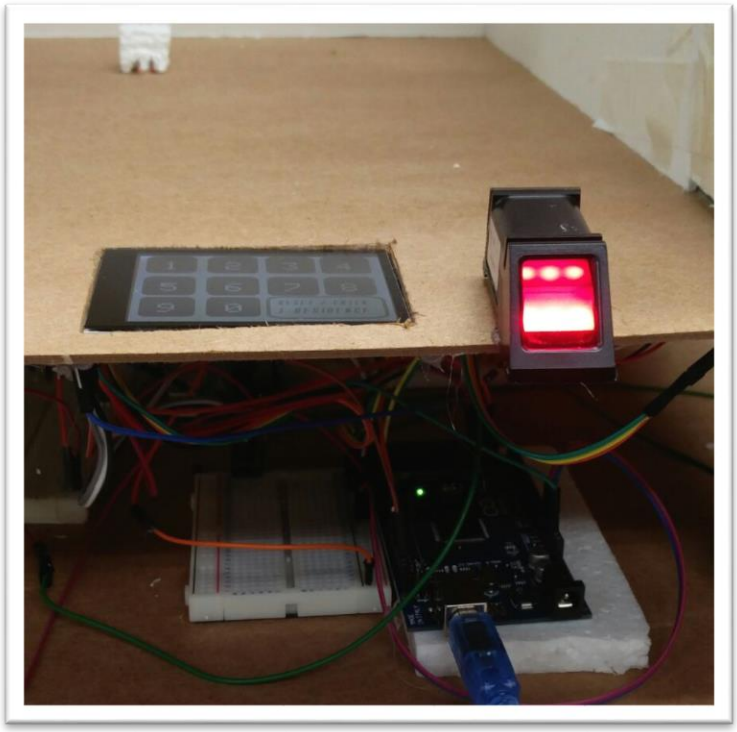
(Algorithm of operation)

1.2 Implementation:



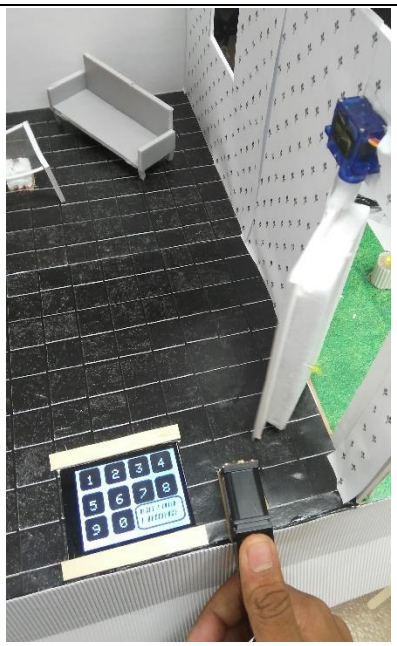
The security system installed)

To implement the smart security system, we first modified the architecture of our building such that the Arduino Mega and all other wiring would be below our smart home. This allowed us to finish all electronics underneath first, before cutting out our structure for the components. Also, this ensured that the wiring is neat and clean and can does not get in the way of our work, while simultaneously allowing serviceability of our electronics.



(Our two-layered design)

Next we needed to implement the fingerprint scanner. Here, we used the Adafruit 751 fingerprint scanners inbuilt library and SD card for operation. The fringerprint scanner communicated with the Arduino Mega via Serial at Serial3 of the Mega. The fingerprint scanner stored a fingerprint as a mathematical data matrix of values. Using the Mega, we read from the fingerprint scanner and save the fingerprint on it's SD card with an ID number. In our security system we then proceed to read the same mathematical data matrix and match the values with our stored one. Due to error and differences in position we accept 80% simiarity as the same fingerprint. This is also the same ballpark percentage used by other commercially availabile fingerprint scanners. If the fingerprint is wrong then no instruction is carried out by the smart security system. However, if the fingerprint is recognzed then a function called openandclosedoor() is called and the door is opened, kept open for a while and then closed automatically. New fingerprints can be stored by the user via a computer using Serial to the Arduino Mega. Further, the scan rate of the fingerprint system has been set to 100 ms, which means that it works quickly and efficiently. A faster system cannot do a very thorough verification while a slower system has too much lag as a result of whih it is not user friendly.



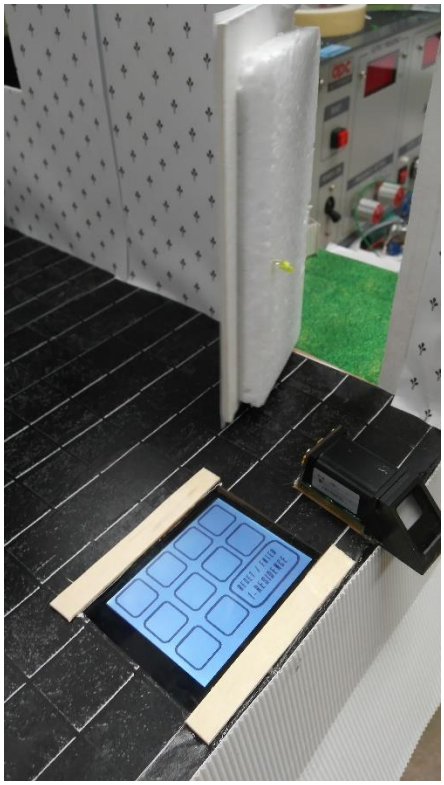
(Fingerprint door opening in operation)

The next component is the TFT touch monitor. This was the most complex part of the security system and required a lot of coding. The monitor first has a blank key image printed on it on startup. This image is stored on the SD card. This image acts like the blank image when the security system is not being used, such as when the door is open. After this we draw all the buttons on top of this image. The buttons are 65 pixels wide and are rounded rectangles. We draw the numbers inside them in white. For the enter key, we draw two colorless buttons which can be touched but not seen. This gives us the functionality of using the enter button of our image without destroying it's artistic value. The touch sensitive ares are also defined according to the dimensions and draw area of each button. If the buttons are touched the button is redrawn in the negative of it's original color. Each button can be pressed only once and not un-pressed until the enter button is pressed. If you touch a button twice, it does not change back to it's original form.

```
if ((px > Xc1-BUTTON_SIZE/2) && (px < Xc1+BUTTON_SIZE/2) && (py> Yc1-BUTTON_SIZE/2) && (py < Yc1+BUTTON_SIZE/2))
{ buttonKEY = 1;
  tft.fillRoundRect(Xc1-BUTTON_SIZE/2, Yc1-BUTTON_SIZE/2, BUTTON_SIZE, BUTTON_SIZE, BUTTON_RAD, ILI9341_WHITE);
  //tft.fillCircle(Xc1,Yc1,15,ILI9341_RED);
  tft.setCursor(Xc1-10,Yc1-10);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(4);
  tft.setTextWrap(false);
  tft.println(1);
}
else if ((px > Xc2-BUTTON_SIZE/2) && (px < Xc2+BUTTON_SIZE/2) && (py > Yc2-BUTTON_SIZE/2) && (py < Yc2+BUTTON_SIZE/2))
{buttonKEY = 2;tft.fillRoundRect(Xc2-BUTTON_SIZE/2, Yc2-BUTTON_SIZE/2, BUTTON_SIZE, BUTTON_SIZE, BUTTON_RAD, ILI9341_WHITE);
  //tft.fillCircle(Xc2,Yc2,15,ILI9341_RED);
  tft.setCursor(Xc2-10,Yc2-10);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(4);
  tft.setTextWrap(false);
  tft.println(2);}
else if ((px > Xc3-BUTTON_SIZE/2) && (px < Xc3+BUTTON_SIZE/2) && (py > Yc3-BUTTON_SIZE/2) &&(py < Yc3+BUTTON_SIZE/2))
{buttonKEY = 3;tft.fillRoundRect(Xc3-BUTTON_SIZE/2, Yc3-BUTTON_SIZE/2, BUTTON_SIZE, BUTTON_SIZE, BUTTON_RAD, ILI9341_WHITE);
  //tft.fillCircle(Xc3,Yc3,15,ILI9341_RED);
  tft.setCursor(Xc3-10,Yc3-10);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(4);
  tft.setTextWrap(false);
```

(Code snippet of button color change on press)

After the color is changed, the value of the button is collected by the arduino and stored into the buffer variable (which in our program is called *total*). When any of the enter keys are pressed the buffer variable value is compared to our passkey (which is 1234 in our program). If the passkey matches then the welcome graphic is drawn and then the door is opened and closed after a delay. At the end of this, the raw grid is drawn again and the buttons are re-drawn to restart the security system. To open the door we use a Towerpro SG90 micro servo motor which recieves a PWM signal from the Arduino. This changes it's position from 0 to 120 degrees (to open) and then after a delay of 3 seconds it is closed via the PWM signal as well. Attached below are some images of the completed implementation of the TFT keypad:



[Top Left]
The keys are turning white whenever they are pressed. Any number of keys can be pressed but no key can be pressed twice for one passkey attempt.

[Top Right]
Our welcome message is displayed when the user enters a correct passkey to unlock the house. Following this, the door is opened and left open for a fixed delay time.

[Left]
After the welcome message, the door opens. While the door is open, the security system is disabled and gets re-enabled only after the door closes again and is secure.

2. Automatic Temperature and Humidity Control:

One of the main features of the smart home is energy efficiency and cutting down on electricity bills. The cooling system is developed on this basis as well. It consists of a fan and a temperature sensor. The temperature sensor records the outside temperature which then determines if the fan is to be turned on or off. The amount of energy a fan uses i.e. the fan speed is also controlled depending on how hot the outside environment is. This method is not only convenient, since it saves the user the need of readjusting the thermostat, but it is energy efficient as well. The major features of the automatic temperature control are summarized below:

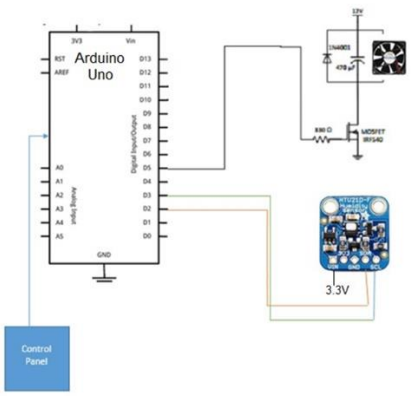
- Maintenance of house temperature as per default set temperature
- Manual control of set temperature to increase or decrease the overall temperature of house
- Permanently turn on or turn off the temperature control system, and thus permanently turn on or turn off the fan
- Very rapid response of fan incase house temperature increases a little bit higher than set temperature
- Cool mode where every component functions at its maximum capacity to make the house as cool as possible in the minimum time

The following sections discuss the design considerations of the automatic temperature control system. The hardware and software requirements are listed in the table:

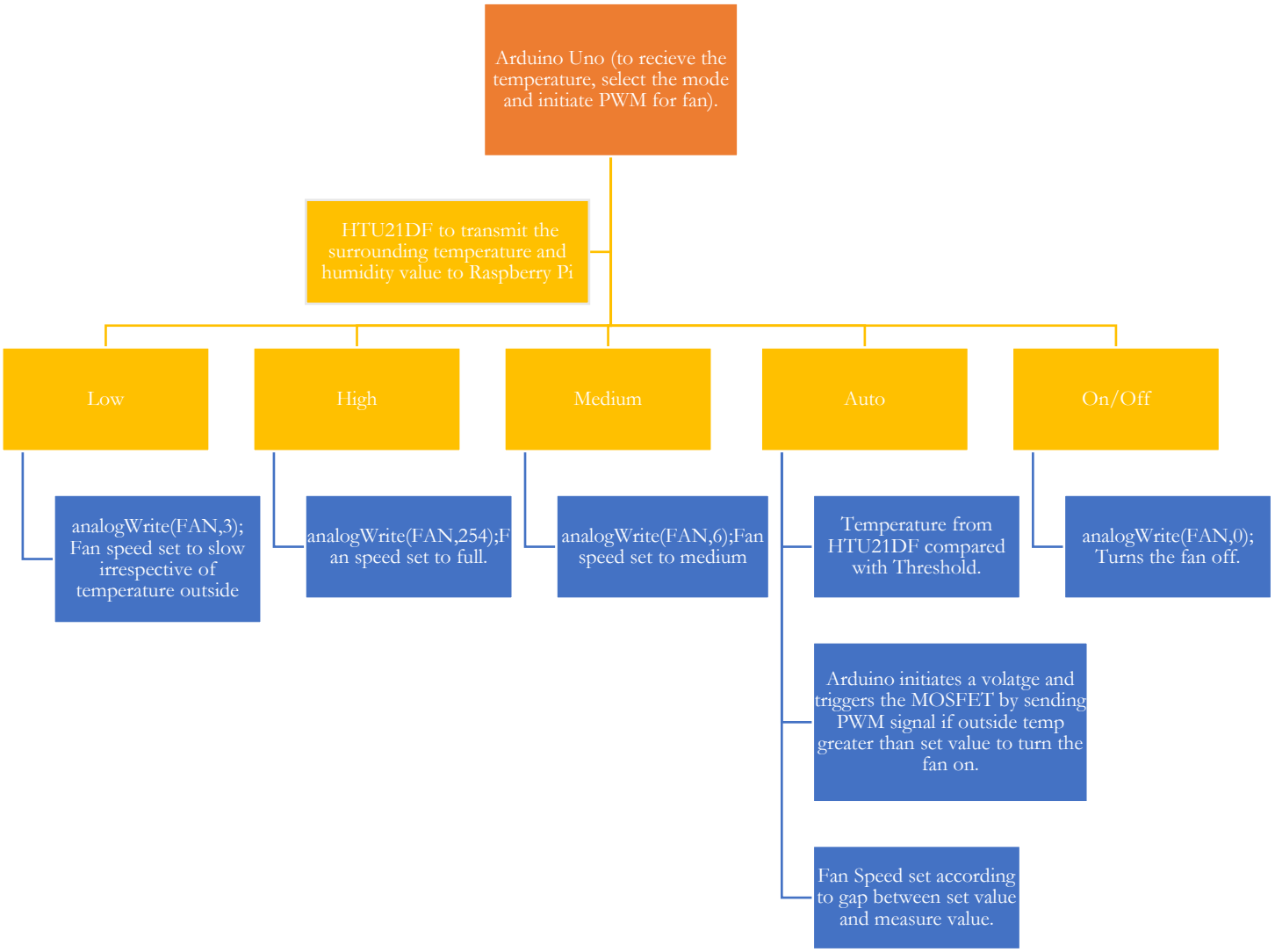
Hardware and software requirement for automatic temperature control system

Hardware	Software
Arduino Uno	Arduino Software (IDE)
HTU21DF Adafruit temperature and humidity sensor	
Fan (12V)	
Power supply (12V)	
MOSFET IRF540	
1N4001 diode	
330 Ω resistor.	
470 μF capacitor	
Wires/cables	

The circuit diagram and flow chart of the automatic fan speed control system is shown below:



(Circuit diagram of automatic fan speed control system)



(Flow chart of automatic fan speed control system)

The fan system is controlled by interrupts sent from the Arduino Uno. The temperature system has 4 modes;

1. Auto
2. Manual
3. PID
4. Off

This mode is selected from a control panel which sends the command to the Arduino Uno and the operation is then carried out. For the **auto mode**, the temperature sensor transmits the data about the temperature and humidity through its SDA and SCL pins to pin 4 & 5 (the SDA and SCL pins in Arduino Uno) of the Arduino Uno. This data for the temperature is continuously transmitted to the Arduino board which compares it with the threshold value set by the user. If the temperature value exceeds the set value, a PWM signal is sent through pin 5 of the Arduino Yun and turns on the MOSFET which completes the circuit and turns the fan on. Now, the speed of the fan is controlled as per the outside temperature. If the temperature outside is just above

the set temperature, the fan speed is slow. As the temperature gap increases, the fan speed increases. This is implemented by setting up various if loops to set different PWM values for different temperature ranges. The PWM value is between 0 and 255 and we set the pin 5 for different PWM for different temperature ranges above the set value. For example, for temperature between 24 and 26, the PWM value was set such that the voltage across the fan was 3V as a result, the fan speed was very slow, between 26 and 27.5 degree centigrade, the voltage was 6V across the fan and the fan speed was medium and above 27.5, the whole 12V was applied across the fan to run the fan at full speed. The fan system with the Adafruit HTU21DF temperature and humidity sensor is shown in the picture below:

(Physical placement of fan and sensor in the project)

In the **Manual mode** there were four options which could be selected, these modes set the value for the PWM irrespective of the temperature outside. If high is selected, the PWM is set to 255 to turn the fan on with full speed, medium sets the PWM to a lower value and turns the fan on with a lower speed and same goes for the low mode. The fourth option was the called 'cool', in this option the whole system performed its best to make the house as cool as possible, and thus everything worked on its full potential to bring the temperature to the lowest possible value.

The **PID mode** was perhaps the most difficult to implement and is surely the most important part in the automation of the temperature control system**,** in this mode full manual control of the temperature could be set by the user, so in return the user can successfully turn on the fan or turn it off, while successfully changing the set temperature in real-time so that the controller then compares the value from temperature sensor and the set value to decide if the fan should be on or off. As the name suggests, Proportional Integral Derivative control system was used to control the fan in this mode. This perhaps, proved to be the most advanced system and the most accurate and responsive one as well. The controller tries to keep the room temperature as close to the set temperature as possible and in the minimum possible time.

An **off mode** was integrated in the system so that if there is any malfunction the temperature system could be turned off. There can also be instances when one is going away from the house for a long time and thus it is very efficient to turn off the system to conserve energy and to cut down on electricity bills.

The main challenge for fan control was to vary the speed according the temperature in the auto-mode. The main realization to be made was that the voltage across the fan does not vary linearly with the PWM value. The curve is one with a decreasing rate of increase. So a PWM signal of 6 gives a 6V output across the fan (half of the total voltage), a PWM signal of 12 gives a 9V output and a 255 PWM value gives the full 12V across the fan. This realization of decreasing rate of increase helped set the values to control speed of the fan.

The control panel responsible for controlling the mode of the fan is connected to Arduino Uno via Raspberry Pie. The humidity and temperature values measured by the HTU21DF are transmitted to Arduino Uno via I^2C and then to the control panel via Wi-Fi. Therefore, any change in temperature and humidity are reflected on the control panel screen.

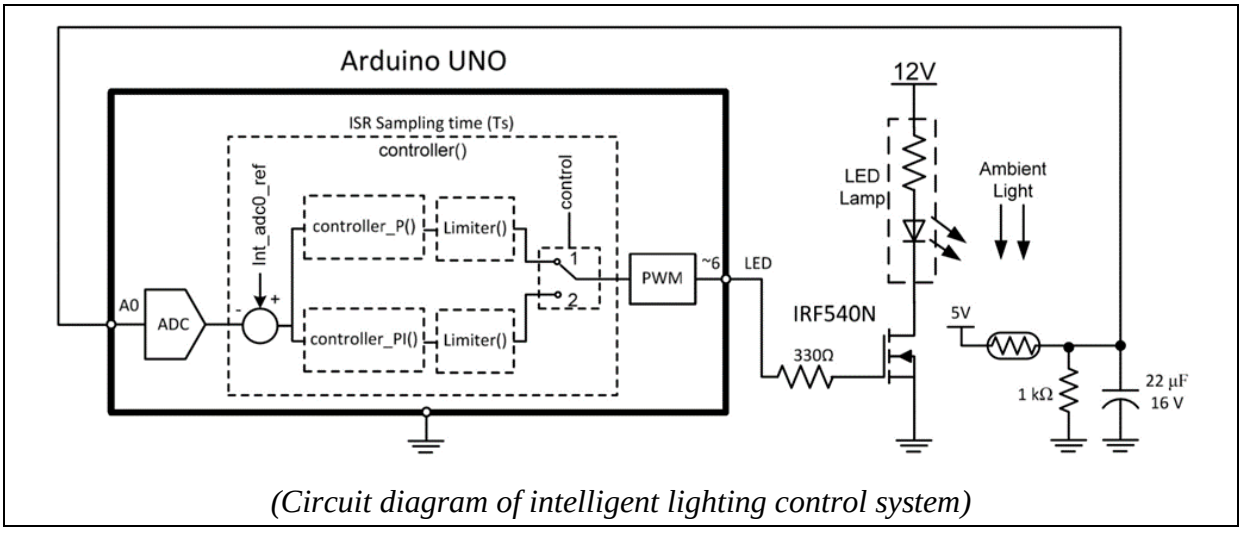
3. Intelligent Lighting Control

The intelligent lighting control system of the smart home provides a convenient way to not only regulate the lighting inside the home but also ensures maximum use of natural lighting. This is an efficient way of energy conservation. The lighting system has 6 modes; automatic, manual, high, medium, low and off. Under the auto mode, the light intensity of the LED adjusts itself according to the light intensity of the surroundings. If the surrounding is bright, the LED's brightness decreases and vice versa. On the other hand, low, medium and high modes can set the brightness of the LEDs irrespective of the ambient light. The manual and off mode are self-explanatory, and hence in the manual mode one can set any level of maximum light intensity and yet it will keep on adjusting itself according to ambient light, whereas in the off mode the light system will be turned off. The required hardware and software are listed below:

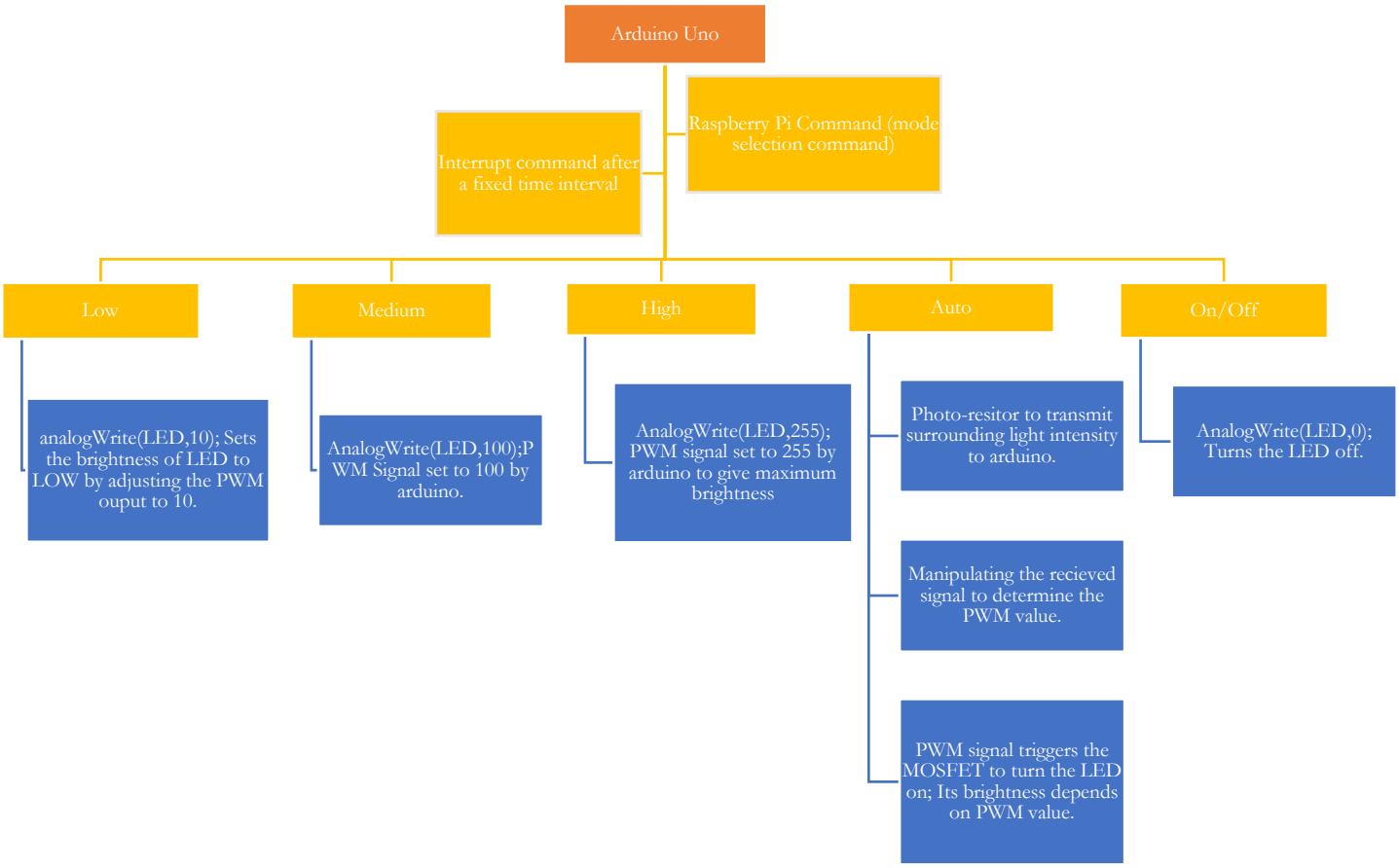
Hardware	Software
Arduino Yun	Arduino Software (IDE)
Arduino Uno	
Photo-resistor	
12V LEDs (×4)	
MOSFET	
Power Supply 12 V	
1N4001 diode	
330 Ω and 1 *k*Ω resistors.	
22 μF capacitor	
Wires/cables	

(Hardware and software requirement for intelligent lighting control system)

The circuit diagram and flow chart of the automatic light control system is shown below:



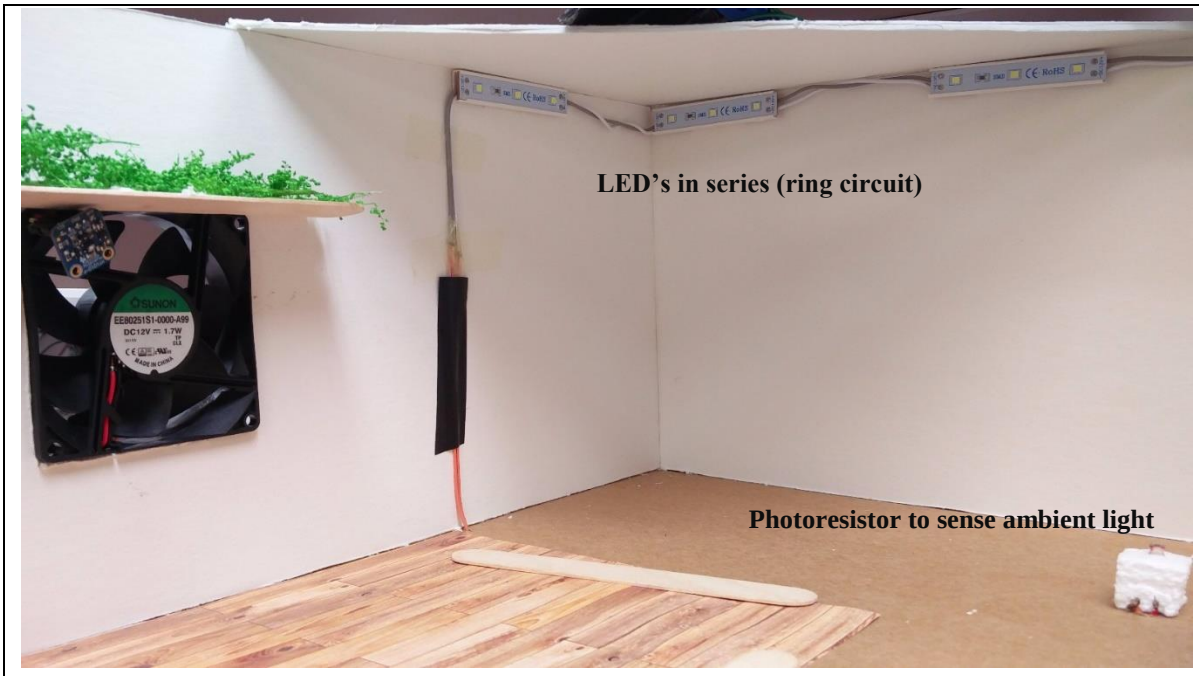
(Circuit diagram of intelligent lighting control system)



(Flow chart of intelligent lighting control system)

The Automatic lighting system has 4 LEDs connected in a ring circuit (as shown in the diagram below) and mounted all around the home. Ring circuit ensures that all the LEDs behave in the save manner and have the same brightness at each stage. The LEDs are linked to a photo-resistor via Arduino Uno and a breadboard. The photo resistor is sensitive to the light and is responsible for transmitting the light intensity of the surrounding to the Arduino board which then manipulates that light intensity value

according the uploaded program to generate a PWM signal and transmit it through pin 6 to the LED via MOSFET. When the light intensity of the surrounding is high, the resistance of the photo resistor decreases and it sends more current to the A0 pin which then uses this to value, manipulates it according to the program algorithm and produces a PWM signal to determine the brightness of the LED. The program running in the Arduino board sends an interrupt command after a fixed amount of time and registers the most updated value for the light intensity and this keeps the brightness in check.



(Picture of actual project showing the lighting system connected in a ring circuit)

The mode selection is done by sending a signal to the Arduino Uno through Raspberry Pi. The program in the Arduino board has options to select auto-mode, on/off, low, medium or high depending on the command sent from the Arduino Uno via I^2C. The program runs an interrupt after a certain time interval, each time updating the command from the Raspberry Pi and hence, making sure that the mode is updated regularly. The auto-mode adjusts the light intensity according to the surroundings light, with the maximum light intensity being 255, shining on the photo-resistor as described above. The manual mode allows to set the maximum value to any desired number, and yet letting the sensor control the lighting system relative to the ambient light. The high, medium and low modes turn the LEDs on with the maximum, intermediate and low brightness respectively, irrespective of the surrounding light. In the manual mode, the maximum. Finally, the off mode is integrated to have an efficient way to turn off the system in case there is malfunction, or when the house is not in use for a very long time. Any of the modes can be selected from anywhere in the world and hence the system is very user friendly.

The main challenges experienced were the fluctuating output of the LEDs, and controlling the mode through the control panel. For the fluctuations, the sensitivity of the photo-resistor was decreased by varying the values of k_p and k_i, and ensuring that the photo-resistor is in vertical position so it receives the same light-intensity from all the four sides. For dealing with control panel implementation issues, the entire

program for all the 4 modes and the on/off command was put inside the interrupt function. This way, every time the program runs and interrupt, it checks the mode selected and implements that part of the loop.

4. **Motion Detection Security Camera**

With increasing crime rates all over the world, iResidence aims to provide enhanced security features so that the user can monitor their home at their own convenience. To meet this goal, we have a Motion Detection Security System which is placed in the main Hall of iResidence.

When the camera detects motion, it sends the feed to the Smart Mirror and keeps updating the image. If no motion is detected after a while, the image is deleted which was displayed on the mirror. There is also a log of 1000 images maintained on board an SD card, controlled by the Arduino.

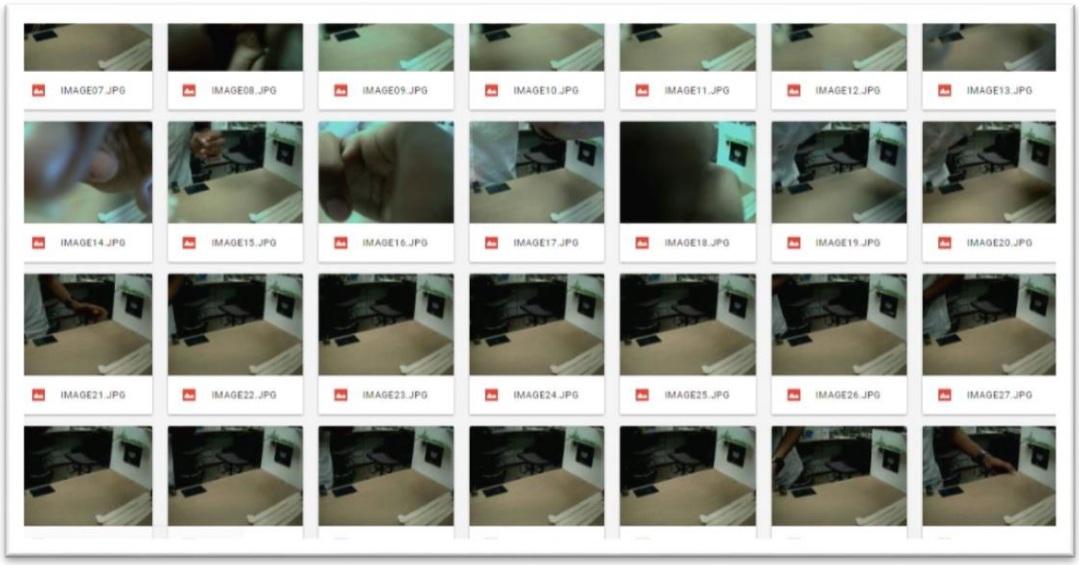
- **Adafruit TTL Serial Camera**

The Adafruit TTL Serial Camera has supported libraries on Arduino which take care of Setting the image size, Capture the image, Motion Detection etc.

- **Arduino UNO Program**

The Arduino program controls the camera, detects the motion and sends commands to Raspberry Pi to display the image on our Smart Mirror.

When motion is detected, Arduino sends a ‘YS’ command to the Raspberry Pi via Serial. Then it proceeds to send the image data to RPi, which is encoded in Base4 for efficient and quick transmission. This is done by using the Base64 library for Arduino. A multiple of 6 bits are read at once from the image, and converted to Base64.



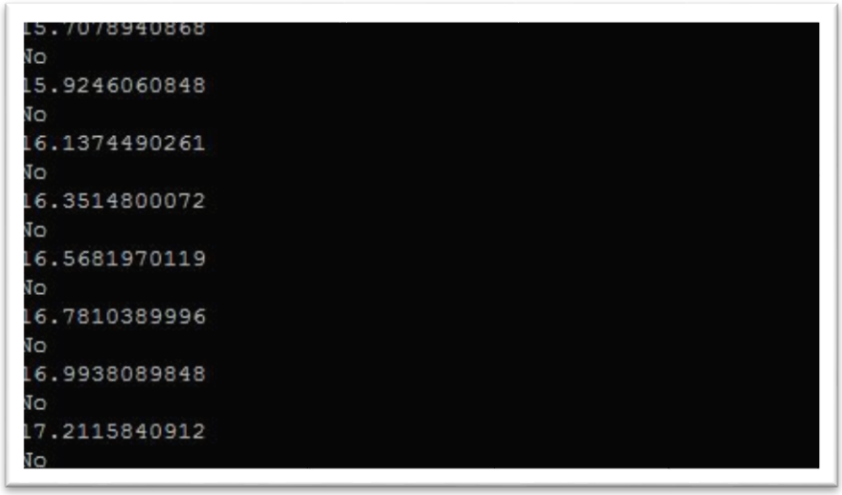
The Arduino program also stores the motion detected images on board an SD card (on the Ethernet shield) to maintain a log of images for a user to go through when they are not present in front of the Smart Mirror.
If no motion is detected, the Arduino sends a ‘NO’ command via serial.

- Python Program on the Raspberry Pi:

```
serialFromArduino = serial.Serial(port,9600)
serialFromArduino.flushInput()
while True:
        if (serialFromArduino.inWaiting() > 0):
                flag=str(serialFromArduino.read(2))
                if(flag=="ys"):
                        print("ys")
                        input = str(serialFromArduino.read(1))
                        img_data=""
                        while(input!="!"):
                                img_data+=str(input)
                                print(input)
                                input = str(serialFromArduino.read(1))
                        print(img_data)
                        f = open('IMAGE.JPG', 'w')
                        f.truncate()
                        f.write(img_data.decode('base64'))
                        f.close()
                        t1=time.time()
                        print("done")
                elif(flag=="No"):
```

The Raspberry Pi runs a Python Program which communicates to the Arduino via Serial. The Python Program receives either a ‘YS’ or a ‘NO’ according to the detection of motion.

When it receives a ‘YS’, it proceeds to receive a base64 encoded image, which it decodes by using the base64 library on Python. It then stores the image in a folder which the Smart Mirror can access and display on screen.

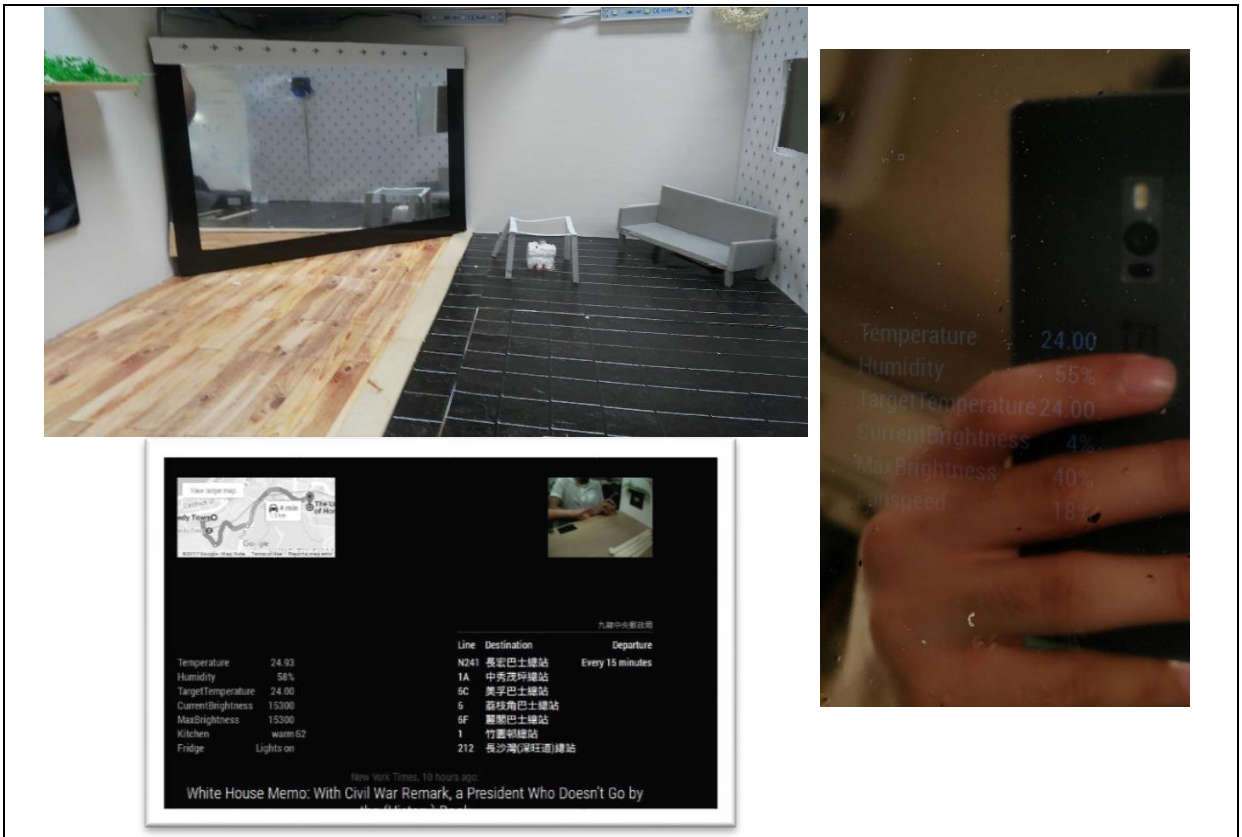


If no motion is detected for 30 seconds, the Python script deletes the image on the screen.

5. **Smart Control Panel:**

The Control Panel at iResidence comprises of 3 components. For providing the user with ease of access of iResidence data, we have chosen the display panel as a SmartMirror which can be placed in the user’s bedroom or bathroom. For simplicity of control, users can control multiple parameters such as the Lighting and Thermostat from their own iOS devices, using the ‘Home’ app which comes pre-installed. Moreover, users can access this data from anywhere in the world and monitor their iResidence with the help of ThingSpeak, a platform for data analytics.

a) Smart Mirror



(The display on our smart mirror)

MagicMirror[2], running on Raspberry Pi, is an open Source Smart Mirror Platform which supports numerous modules. It is a Node JS application, which is running the following modules to display our data:

- MMM-Rest

  This module collects data via HTTP calls and displays it on the mirror in a table.
  - Temperature
  - Humidity
  - Target Temperature
  - Max Brightness
  - Current Brightness
  - Fan Speed

- MMM-SimpleLogo

  This module displays the image captured by the Motion Detection program, onto the Smart Mirrror
  - Displays the image stored in the directory onto MagicMirror
  - Updates the image every 1000ms from the directory

- MMM-googlemaps

  Displays the distance and time between two locations, on the Smart Mirror. The

two locations are set in the config.js file of the Smart Mirror.

- Newsfeed
  Autoscrolls current newsfeed onto the Smart Mirror.

- MMM-HK-Transport
  Station Monitor for Hong Kong Public Transport. Updates the status of pre-configured routes in real time.

For updating iResidence's sensor data onto our Smart Mirror, we run a Python Script on the Raspberry Pi which:

Receives JSON data from Lighting and Temperature Control Arduino

- Updates the data onto a Webpage for the Smart Mirror to access.

The following data is handled by the Tornado Web Server:

- Retrieving the current status of
  - Photoresistor
  - Adafruit HTU21D – F Temperature and Humidity Sensor
  - Lighting
  - Fan

b) HomeBridge

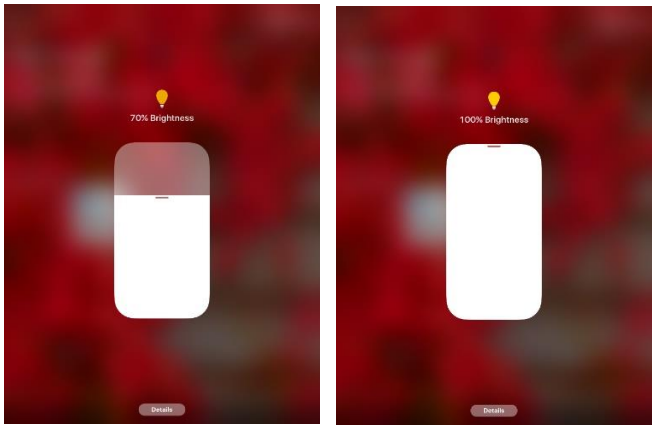
HomeBridge is an iOS HomeKit API emulator, which runs on NodeJS (NodeJS server). It supports numerous Plugins, in the form of Community-Contributed Modules, which provide a basic bridge from HomeKit to 3rd Party APIs.

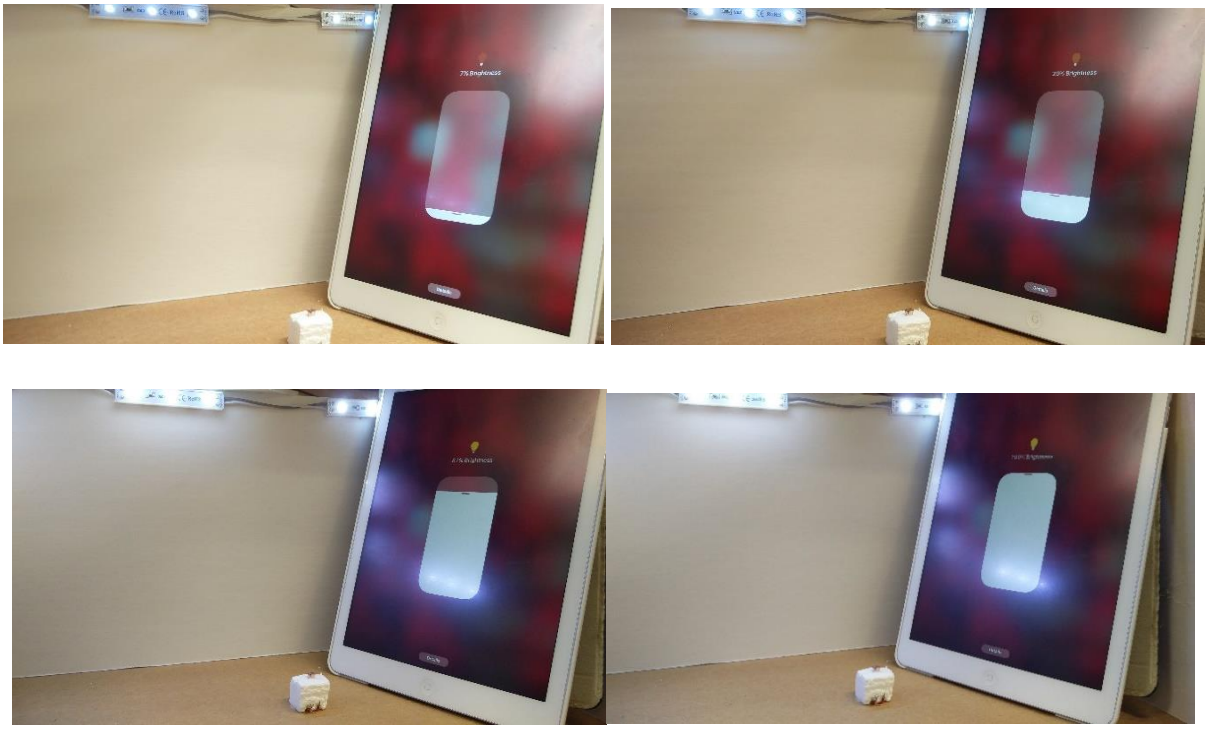
(Our home bridge app display interface)

For iResidence, we can control the following parameters through an iOS device with the Home app:

- Lighting:
  - On and Off control:

  - Brightness Control:

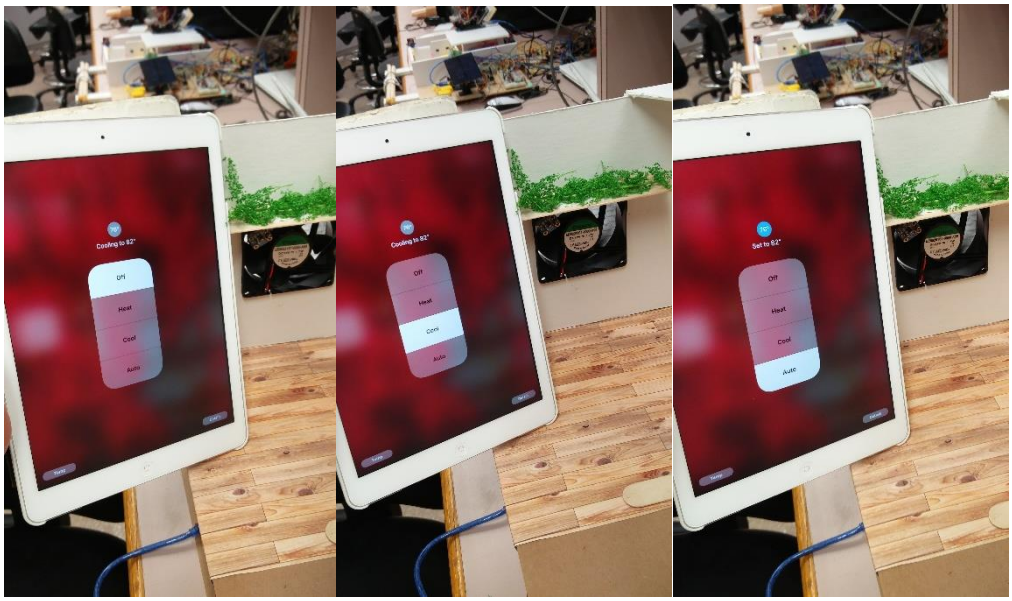




(Example of brightness control)

- Intelligent Ambient Light Adaptation
  - Lighting adjusts as per the ambient light

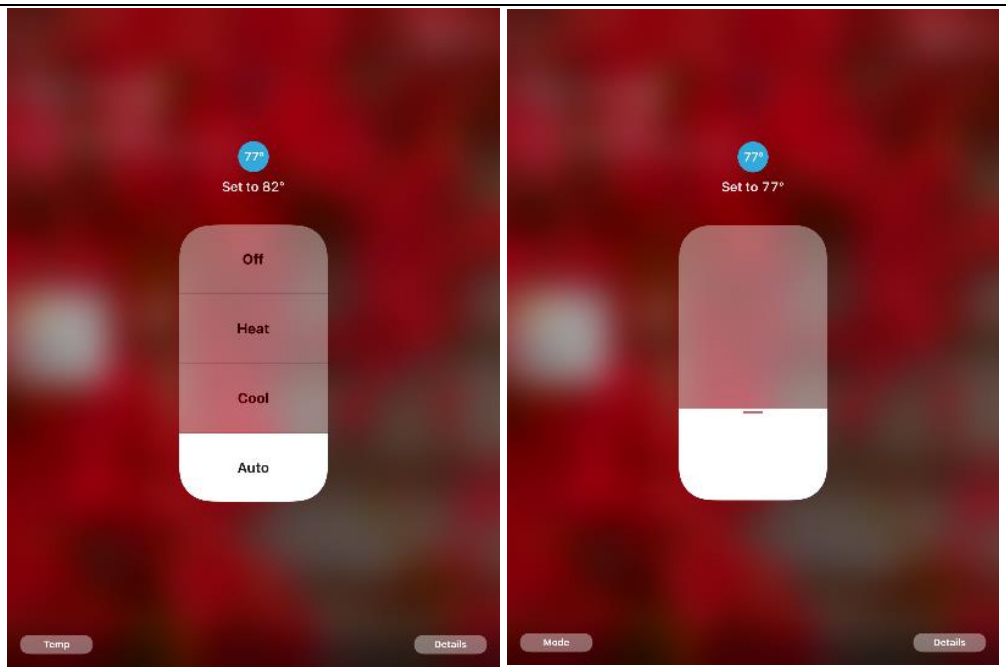
- Fan
  - On Off control



(The fan being controlled via the iPad)

(The options on the iPad)

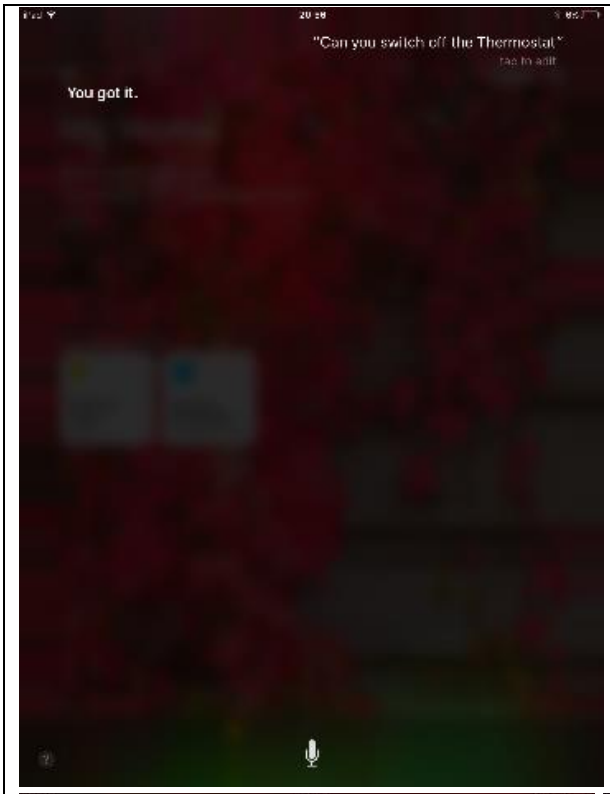
- Auto Temperature Control

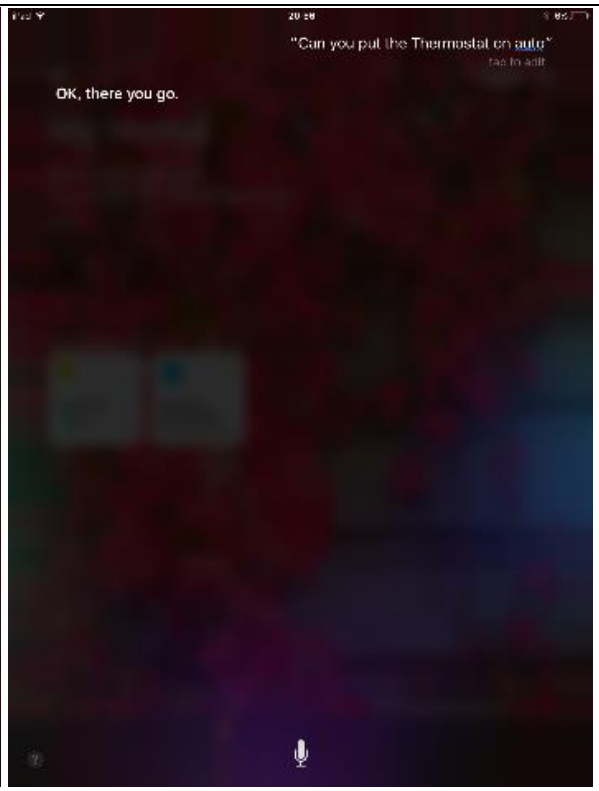


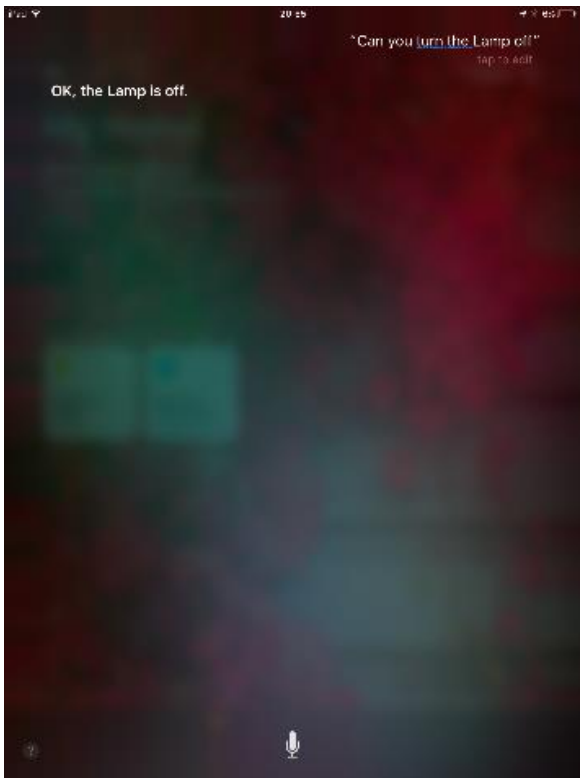
(The temperature setting and thermostat options on the iPad)

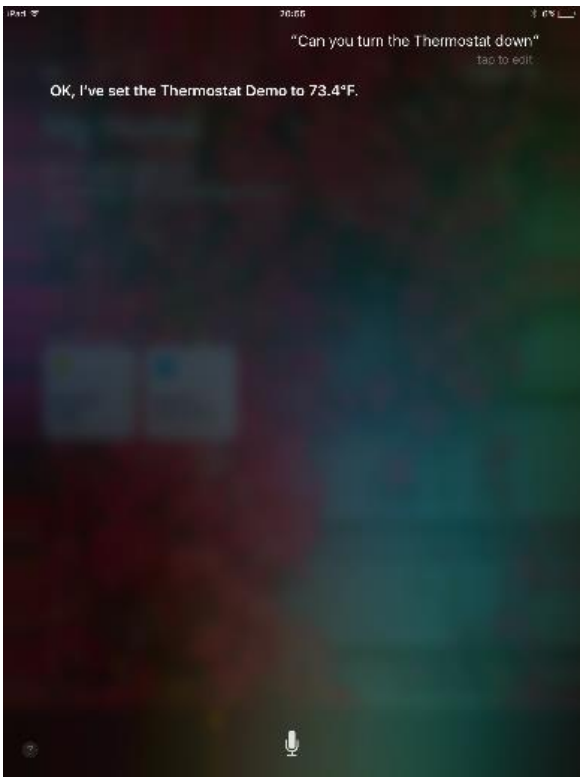
- Set Target Temperature of the home
- Fan Intensity automatically adjusts as per the current temperature and target temperature.

- Siri Support
  Since iOS Home app is integrated with Siri out of the box, Siri commands can be used to control features in the following way:









"Switch on the lights"
"Switch the Thermostat off"
"Turn down the Thermostat"

- For running Homebridge on RPi, we had to convert it into a Router and a Webserver. The Raspberry Pi is converted into a router with the name ‘mirror’ by making it:

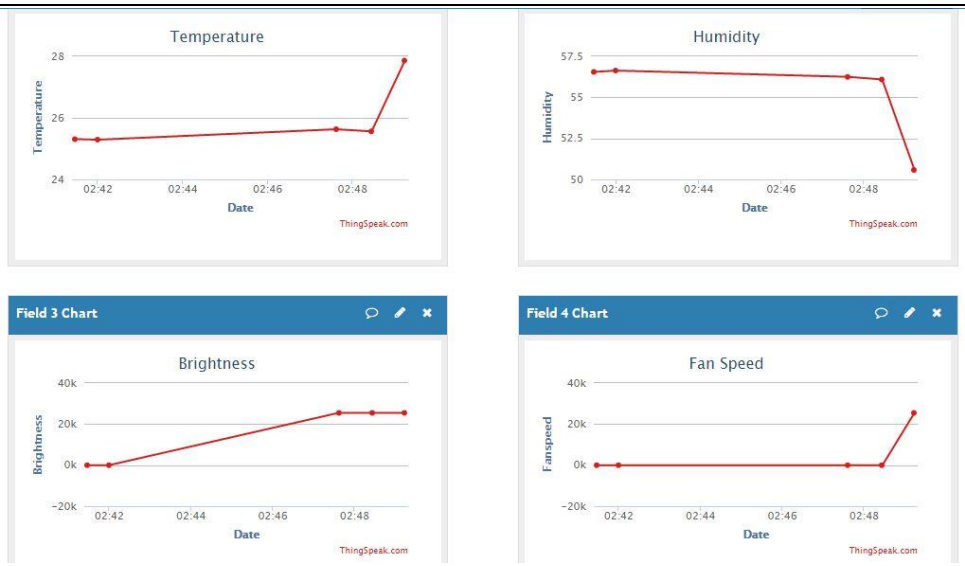
(Our Raspberry pi router available for connection)

  - A Wi-Fi access point (AP)
    - By installing the ‘Hostapd’ software.
  - DHCP and DNS server
    - By installing and configuring ‘Dnsmasq’.
  - IPv4 and IPv6 network forwarder
  - Network Address Translation (NAT)
    - Bridges the Internet Connection on eth0 to wlan0
    - By configuring ‘iptables’

The Webserver is created using Tornado on Python, which handles HTTP requests from both Homebridge. The following data is handled by the Web Server:

  - Retrieving the status of
    - Photoresistor
    - Adafruit HTU21D – F Temperature and Humidity Sensor
    - Lighting
    - Fan
  - Update the status of
    - On-Off status of:
      - Lighting
      - Temperature Control
    - Max Brightness Level
    - Target Temperature

c) ThingSpeak:



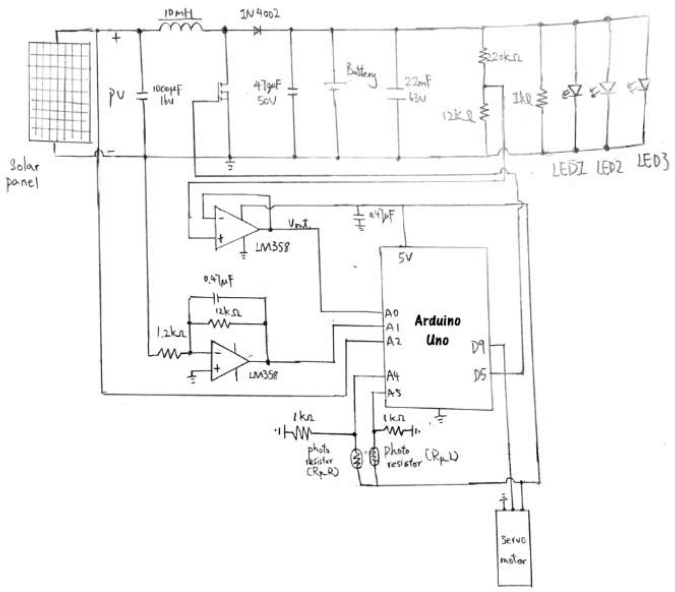
(The data of the smart home being plotted in real-time)

For meeting the goal of monitoring iResidence from anywhere, we log our sensor data on ThingSpeak, which is a platform for data analytics. The python programs sends data periodically (5 minute interval) to ThingSpeak via HTTP requests. The following data is sent:

- Temperature
- Humidity
- Brightness
- Fan Speed

The User can monitor home data from anywhere by using this link. The user can further enhance the monitoring by using MATLAB for more insight on the data.

6. Solar Panel and Solar Tracker:



(Schematic Diagram)

The purpose of the circuit is to harvest the maximum power from the solar panel and use it to supply the major energy consuming devices in the smart home. As we want to get the energy when the solar panel is not working, a battery is installed to store the energy extracted from the solar panel. Also, as we want to increase the efficiency of the solar charger, a light tracking system is installed. The light tracking system allows the solar panel to follow the path of sunlight and to get the maximum power.

So the solar panel is divided into 3 parts: solar charger (shown in the top left corner of the schematic, from solar panel to battery), power supplier (shown in the top right corner of the schematic, from battery to load) and the light tracking system (shown in the bottom of the schematic).

(Our solar lights powered by the panel)

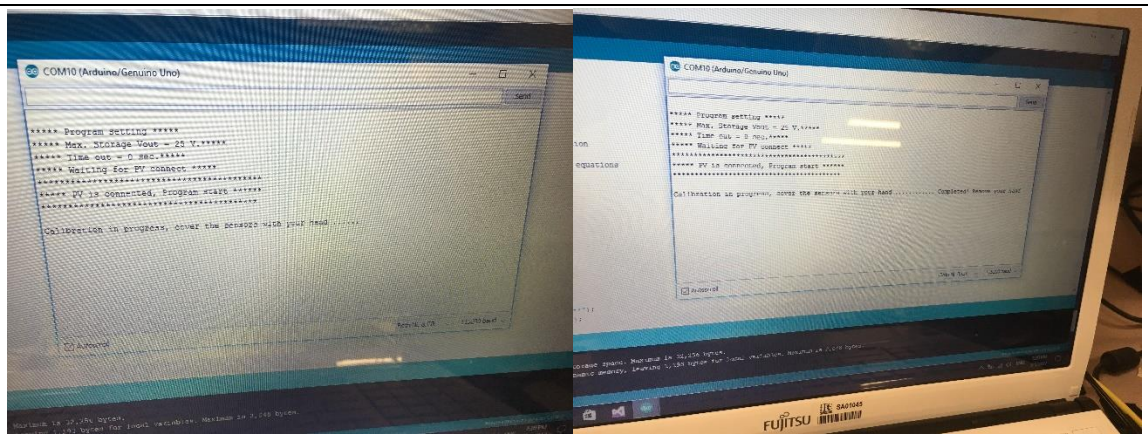
Firstly, we talk about the solar charger. It is the most difficult part of the whole system. The major consideration in building the circuit is how to stabilise the unstable voltage provided by a solar panel and step down the voltage of solar panel to charge the battery. The open circuit voltage of the solar panel is around 10 -12 V, depending on the light intensity. But the storage is a Li-MH battery produced by Samsung, and its maximum voltage is around 3.7V which can support the LED lighting. Thus, a buck converter is used to step down the solar panel (10V to 12V) to charge the battery pack (3V-4V).

Then the battery will be connected to the led lighting outside the house to light up the garden of the house. As lighting in the garden is useless in the morning, so an energy storage must need to supply the energy to the lighting. In the morning, the solar panel will be responsible

for charging the battery. At night, solar panel stops working, so the battery will start working and supply the energy to the lighting system.

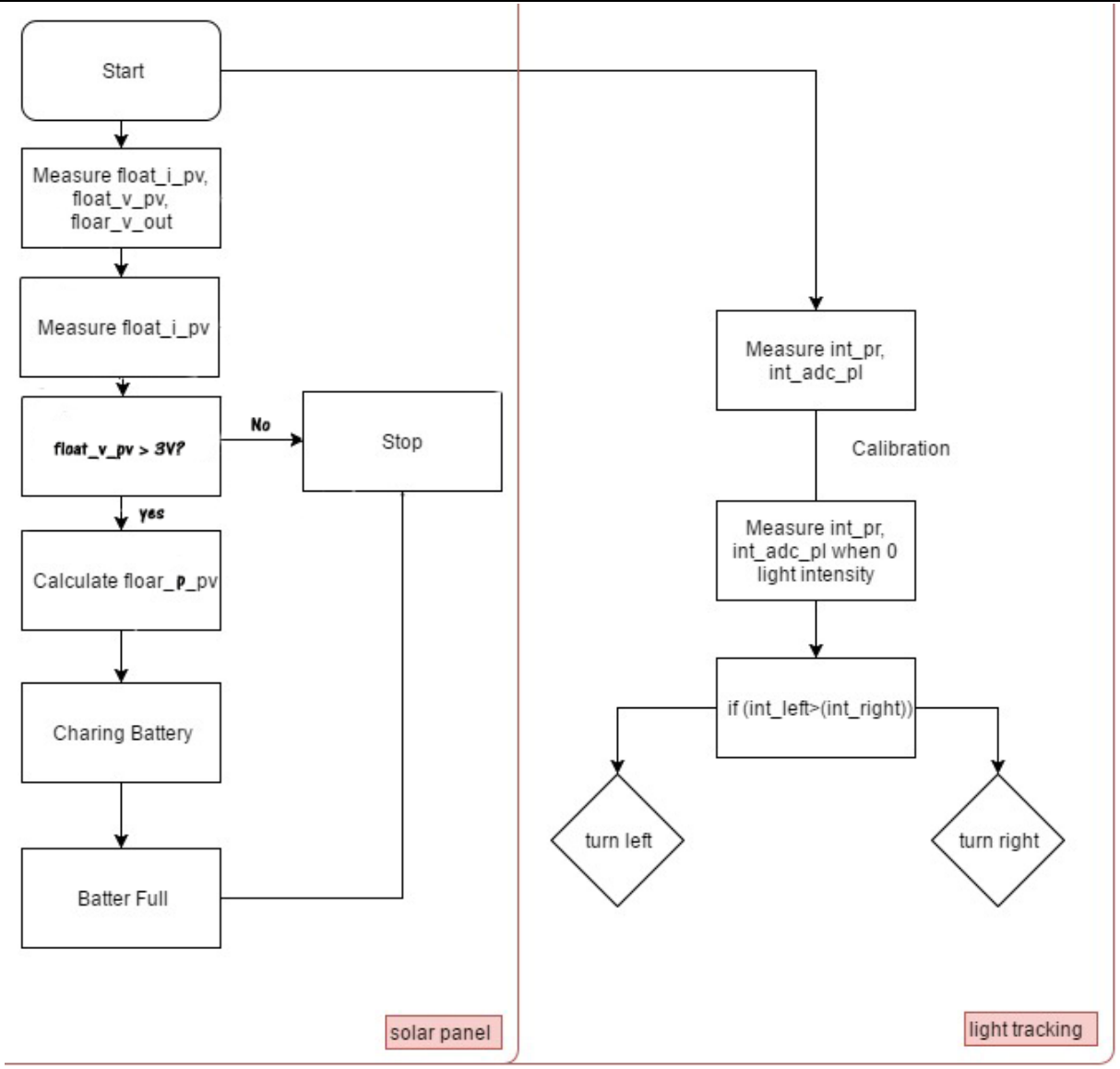
(The solar panel and light tracker)

For the light tracking part, it aims at the simulate the daily operation of a real solar panel. As the sun rise and fall every day, so the solar panel need to keep tracking of the movement of the sun to gain maximum power from it. There are two photo-resistor at the left side and right side of the solar panel respectively. Photo-resistor can sense the light intensity, and its resistance will change according to the light intensity. The resistance of a photoresistor decreases with increasing incident light intensity. We can make use of the properties of photo-resistor. When the resistance of two photo-resistor is the same, it means that the solar panel is facing the sun.



(Voltage and intensity data being fed into our serial monitor for debugging)

First, the program will measure the voltage and current of solar panel and the output voltage. Then it will check whether the power of solar panel is larger than 3V. At the same time, the light intensity of two photo-resistor will be measured. Then, calibration starts, we need to measure the resistance when zero intensity. Then there will be two output, which is the light intensity of left and right photo-resistor based on the value measured during calibration. Two values will be compared to decide which side should the solar panel turns.



]

(Solar panel and light tracker operational algorithm)

8. Prototype performance:

1. **Motion Detection Security Camera:**

(Prof. Lee inspecting our Automatic Temperature Control System)

We were able to integrate the Motion Detection Security Camera System to the utmost satisfaction. The key to swift and efficient data transmission from the Arduino to Raspberry Pi (Control Panel) was to base64 encode the image before the communication. We also overachieved compared to our initial proposal by integrating a two-level security system in the form of SD card photo logging.

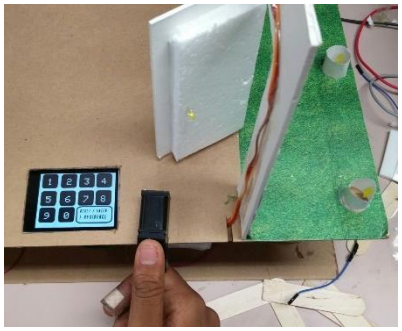
2. **Automatic Temperature Control:**

The Automatic Temperature Control in our prototype exceeded our expectations as well. Initially, we had planned to only display Temperature, Humidity and Fan Speed data onto our SmartMirror, but we outperformed our proposal by integrating Homebridge with the Temperature Control. By choosing the target temperature, on or off values on any iOS device, users can control temperature regulation in iResidence at their own convenience.

3. **Intelligent Lighting System:**

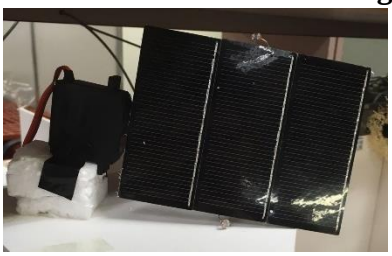
Much like Automatic Temperature Control, The Intelligent Lighting System in our prototype exceeded our expectations as well. Not only does our SmartMirror display the current brightness level and max brightness level, we can also control these parameters using any iOS device connected to our Raspberry Pi router.

4. **Smart Door Lock:**



The Smart Door Lock performed as per our expectation as well. In the prototype, we were able to accomplish the integration of a Fingerprint Scanner and Keypad Lock, which was in the form of a touchscreen LCD.

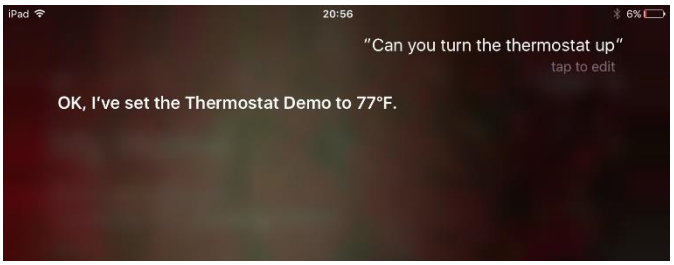
5. **Solar Panel + Automatic Light Tracker:**



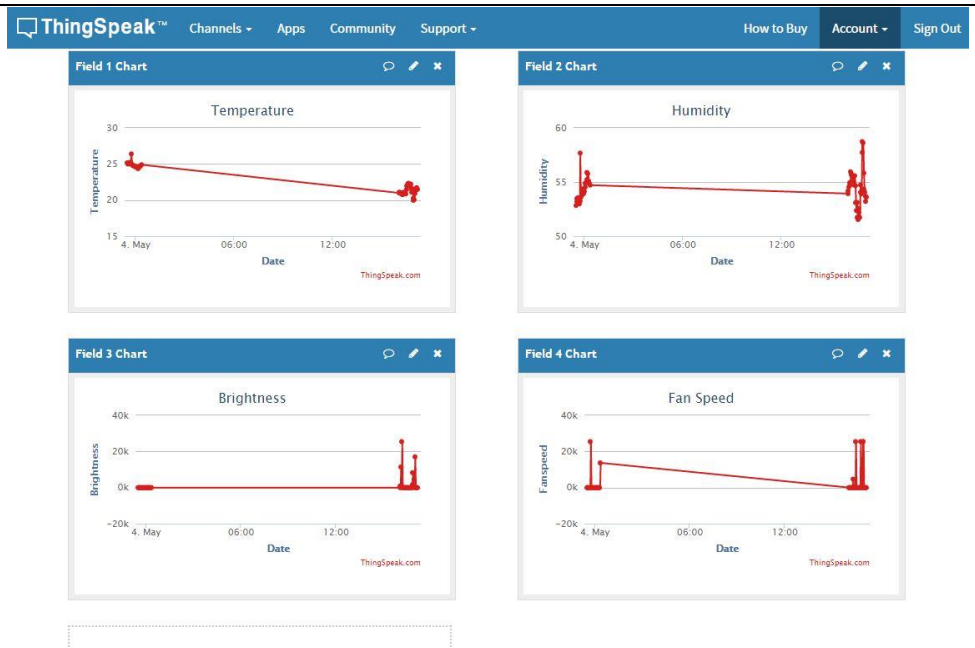
The Solar Panel and Automatic Light Tracker in the prototype performed as per our expectations. The Solar Panel not only tracks the position of the light, but also gives a substantial amount of power which we utilize to power LEDs in the backyard of our prototype.

6. **Control Panel (SmartMirror + Homebridge + Thingspeak):**

In the prototype of iResidence, we were extremely satisfied with the integration of all the aspects of our Control Panel. Not only did we achieve our initial goal of monitoring iResidence from the SmartMirror, we also successfully integrated HomeBridge into our prototype for controlling Temperature and Lighting inside iResidence.



We also accomplished to integrate Thingspeak into iResidence to monitor the various parameters inside the Smart Home and run data analytics onto it.



(iResidence data being mapped live on the webpage)

9. Discussion and future work:

During the course of our project, we realized the importance of integrating the various components of our Smart Home together. Although we have outperformed each target set for ourselves as per our initial proposal, there is always scope for improvement. We have combined the Temperature Control, Lighting Control and the Motion Detection System together on our Smart Mirror and Control Panel, but in the future, the Smart Door Lock and the Solar Tracking System can also be integrated. A heating system can also be accommodated inside the thermostat, in addition to the pre-existing cooling system in iResidence. For the smart door lock, a touchscreen LCD pin lock and fingerprint sensor is beyond satisfactory, but further features such as another camera outside the front door and voice commands to open the door lock can be integrated as well. For the Solar Panel system, we can use a bigger Solar Panel and a Battery with a higher voltage, so that the system can supply electricity to the entire house.

Finally, we feel it is important to stress on the need for a smart plug and play solution for smart homes today. Rather than integrating different components one by one, we need a single system that readily accepts and connects to smart home "modules" based on a single standard communication protocol. This will greatly benefit smart homes in the future and shall help take our project out of the lab and into the real world.

References

1) Smart Home Definition | Investopedia. (n.d.). Retrieved from http://www.investopedia.com/terms/s/smart-home.asp

2) 1.5 Million Home Automation Systems Installed in the US This Year ... (n.d.). Retrieved from https://www.abiresearch.com/press/15-million-home-automation-systems-installed-i

3) Image Source: https://image.slidesharecdn.com/shcp13gould-131119070740-phpapp01/95/sensinode-arm-smart-homes-cleanpower-2013-cambridge-uk-via-cir-wwwhvmukcom-6-638.jpg?cb=1384844912