Smart Home

Submitted by:

Mariam Al Sadek 201305803

Hasan Beqai 201204466

Supervisor: Dr. Harag Margossian

School of Engineering

Department of Electrical and Computer Engineering

A final year project report presented to the Lebanese American University in partial fulfilment of the requirements of the degree of Bachelor of Engineering

Fall 2016

Table of Contents

Table of Figures

Table of Tables

Abstract:

This project includes the design and implementation of an individual controlled home automation system using ESP-12E, android and Arduino microcontroller. This automatic or semi-automatic system will allow the user to control and/or monitor his/her household appliances.

This project is an illustration of how to model and implement a multipurpose smart controlled system that can switch on or off any home appliance either manually via switches or remotely via mobile phone or laptop which are connected through ESP8266 WIFI module to Arduino microcontroller that controls the relays for automatic switching of appliances. The state of appliances is shown in the android application along with different options such as home power consumption and cost. The power supply will involve solar power system to reduce electric bill charges and help preserving the environment.

This project shows how powerful an Arduino microcontroller is. It can be used for building smart electronic devices

1. Introduction:

This report is describing our Capstone Design project. The project is entitled the "Smart Home". A Smart home can do a lot of things based on IoT "internet of Things" technology. It allows the user to choose the configuration that suites his/her needs, with automatic adjustment based on available conditions. These are the sort of things that an automated home can do, with endless applications. Smart homes are able to identify its owner through recorded Radio

Frequency Identification tags and automatically order shopping lists. This is what often referred to as a "Smart Home".

A smart home is a home that can act based on its owner preferences. It is linkage among Security, Telecommunication and Entertainment systems, which is integrated into active partner.

1.1. Automation

Automation is "the technique, method, or system of operating or controlling a process by highly automatic means, as by electronic devices, reducing human intervention to a minimum." as defined by dictionary.com

Automation is playing an increasingly important role manufacturing sector and in daily life aspects. Engineers succeed in creating a system that rapidly extends the range of applications and human activities through a combination of automated devices and mathematical and organizational tools. (IEEEXplore, 2006).

Many roles of human in industrial domain are within the scope of automation, including face detection and image synthesis. However, there still a wide range of activities that are exclusive to human, such as tasks that require subjective assessment of sensory data like scents and sounds and language production ability. These activities are beyond the current scope of automation.

The impact of automation is highly notable in large industries. "Once-ubiquitous telephone operators have been replaced largely by automated telephone switchboards and answering machines. Medical processes such as primary screening in electrocardiography or radiography and laboratory analysis of human genes, sera, cells, and tissues are carried out at much greater

speed and accuracy by automated systems. Automated teller machines have reduced the need for bank visits to obtain cash and carry out transactions. In general, automation has been responsible for the shift in the world economy from industrial jobs to service jobs in the 20th and 21st centuries." (Aptinex, 2014). As the decades passed, the scale of activity of automation dramatically increased and it was clear that quality of services that can be delivered by home automation is a good choice to invest in.

1.2. Home Automation

Home automation make it possible to have a "Smart Home". Juniper Research defines a Smart Home as one that is designed to deliver or distribute a number of services within and outside the home through a range of networked devices (2014). While high-speed Internet access is not essential for all devices deployed in the home, the full functionality of a Smart Home depends on the availability of a permanently accessible broadband Internet connection. There are many definitions that can be attributed to the Smart Home concept, and these change as time and technology progresses. Where previously a Smart Home may have been one with multiple devices independently connected to the Internet, a more accurate description today would be one where multiple devices might be connected in the same instance to a central hub. Where before several controllers were needed to communicate with each device, today's Smart Home is more geared for central control. The Smart Home scenario which will present service providers and vendors with the greatest revenue opportunities, is one where the connected devices in the home are all interconnected and have the potential to share information with one another; that is to say, they have become part of IoT (the Internet of Things).

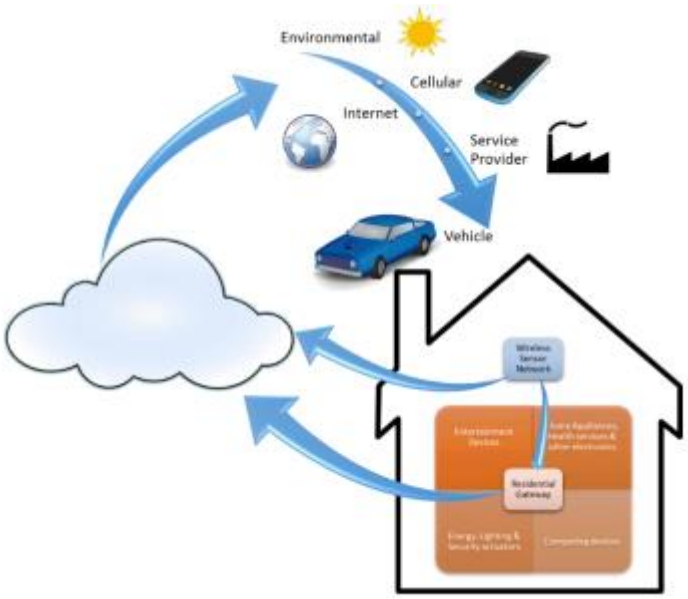


Figure 1 Smart Home Dataflow

1.3. Project Aim

The project aims to design and implement a home automation system with using solar power as alternative energy supply resource. This system will remotely switch on/or any household appliance connected, using Arduino as a microcontroller, android and html5 for requests and feedback.

1.4. Project Objective

The objective of this project is to construct a low cost, reliable, scalable and environmental friendly smart home system.

1.5. Project Scope and Limitation

This project work is almost complete on manually and automatically controlling operation of home appliances, with the possibility of controlling of multiple appliances. It also includes a standalone solar power system that will be used to supply the smart home. However, the implementation of automatic solar system supply is not completed.

1.6. Project Justification

This project is a contributory knowledge to the development and implementation of a smart home in Lebanon using low cost and reliable available resources like arduino and android app.

1.7. Report Layout

This report is composed of five chapters.

- Chapter 1, is an introduction to home automation.
- Chapter 2, is an extensive literature reviews of previous works on home automation systems.
- Chapter 3, is a description of methodology used in implementing this project, giving reasons for choice of specific components.
- Chapter 4, is a detailed description of the implemented project.
- Chapter 5, is a conclusion and recommendation section.

2. Literature Review

2.1. History of Home Automation

The television remote which is a simple home automation system was patented in 1893. “Since then different automation systems have evolved with a sharp rise after the Second World War. Its growth has been through various informal research and designs by technology enthusiasts who want a better way of getting things done at home without much effort on their part. The systems evolved from one that can automatically do routine chores like switch on and off security lights, to more sophisticated ones that can adjust lighting, put the television channel to favorite station and control doors”. (Manohar et al, 2015)

2.2. Home Automation Systems

Home automation may assign electronic framework that will automate home appliances. “The new stream of home automation systems has developed into a vast one and the current market is flooded with a flurry of home automation systems and device manufacturers.” (Sirisilla Manohar et al, 2015). Home automation systems have different categories based on their control systems.

- Individual Control Systems: first systems that hit the market. Each appliance in the home has its independent control.
- Distributed Control Systems: mainly featured by emergency shutdown. With this system several similar devices parameters can be changed.
- Central Control Systems: computerized systems to handle function of multiple utilities. You can connect to this central system through telephone or internet from anywhere in the world. (Sirisilla Manohar et al, 2015)

The categories based on their carrier system, are:

- Power line Carrier Systems: operates over existing wires or power line of the home. Ranging from X10-based lamp timers to more sophisticated systems that require trained professional. This is the least expensive type of home automation.
- Wireless Systems: utilize RF technology. Usually used to control lights.
- Hardwired Systems: most reliable and expensive system. These systems can operate over high-grade communications cable. They are planned when the house is constructed.
- Internet Protocol Control Systems: uses the internet, gives each device under its control an IP address, and creates a local area network (LAN).

2.3. Home Automation Standards

There are many industry standards for home automation systems and are implemented over the various carrier modes ranging from power line to wireless. Smart devices all run on a variety of different protocols. The popular and major standards are INSTEON, European Home Systems (EHS), ZigBee, KNX, Z-Wave, and X10, Lon Works, ONE-NET and Universal Power Line Bus (UPB).

Standard	Details
KNX	A standardized EN50090, ISO/IEC 14543, OSI based network communications protocols for building automation, it defines several physical communication media, such as twisted pair wiring, power line networking, radio and Ethernet. It converges from: EHS, BatiBus and EIB protocols. It is administered by the KNX Association. It allows tree, line and star bus topologies.
WIFI	Boasting high bandwidth, Wi-Fi is already pretty much everywhere, so many manufacturers are enthusiastically making smart home devices to work with it. A multitude of homes in the U.S. already have wireless routers (which work on the Wi-Fi protocol), so obviously they've already got a central hub in place to which Wi-Fi compatible devices can be connected. This does, however, come with one key drawback: interference and bandwidth issues. If your house is full of Wi-Fi-connected gadgets (TVs, game consoles, laptops, tablets, etc.) then your smart devices will have to compete for bandwidth and may be slower to respond. Wi-Fi is also hungry for power; consequently, battery-operated smart devices such as locks and sensors get drained much sooner than in other wireless environments.
IP-Internet	Specifies the format of packets, and the addressing scheme. Most networks combine IP with a higher-level protocol called Transmission Control Protocol (TCP), which establishes a virtual connection between a destination and a source. IP by itself is something like the postal system. It allows you to address a package and drop it in the system, but there's no direct link between you and the recipient. TCP/IP, on the other hand, establishes a connection between two hosts so that they can send messages back and forth for a period of time. The current version of IP is *IPv4.* A new version, called *IPv6* is under development.

Http	The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Standards development of HTTP was coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs).

Table I. Standards Table

3. Methodology

In home automation design different platforms can be used. In order to build a reliable and flexible system, some choices were made on the type of platforms, software, hardware components and modes of operation of the smart home.

3.1. Preliminary Consideration

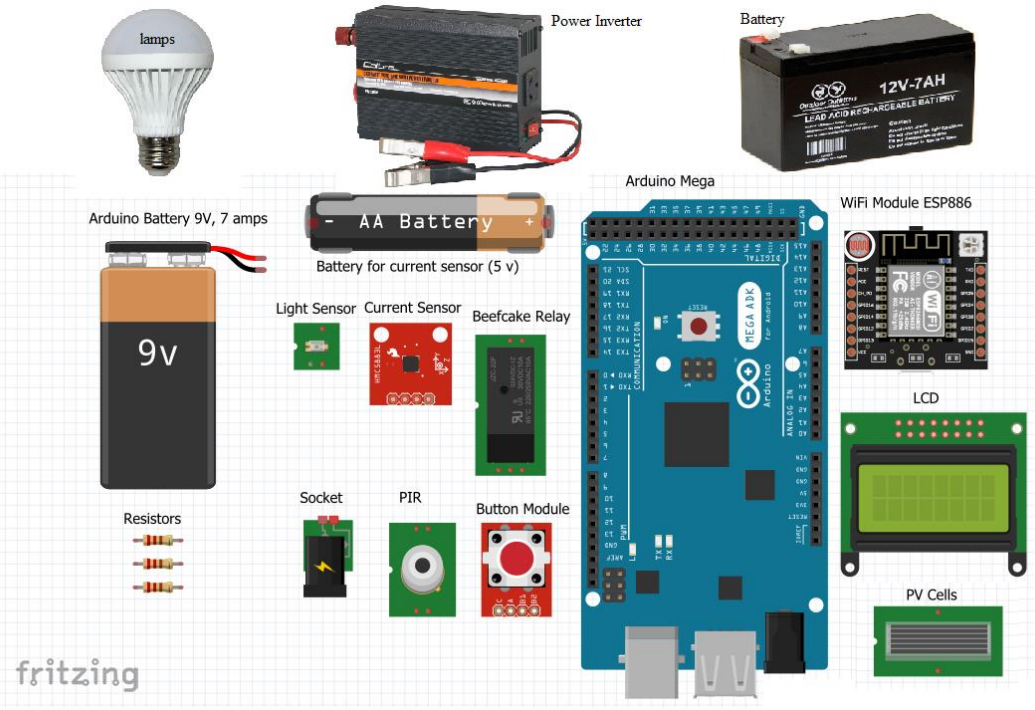
Before construction of the actual design, specific choices in selection of appropriate software and hardware were made. Priority was given to low cost availability, reliability, flexibility and simplicity in all these selections.

3.1.1. Selection of implantation platform

As shown in the previous section, there are many platforms and protocols that apply to home automation. Of the currently available platforms, Power line, Wi-Fi, http communication, Arduino Mega microcontroller, Android platform, .net framework are selected to be adopted through the implementation of project. The compilers used to implement the software programs are Arduino, Visual Studio and Android Studio.

3.1.2. Selection of hardware components

For hardware the components used are Ethernet cables, ESP8266 Serial WIFI Module, Beef Cake relays, sensors (PIR motion sensor, light sensor and current censor), android mobile phone for testing the software application.



For the microcontroller, Arduino Mega was found to be suitable due to its low cost, availability and available sample programs, simplicity and flexibility.

3.2. System Design

3.2.1. Arduino Mega

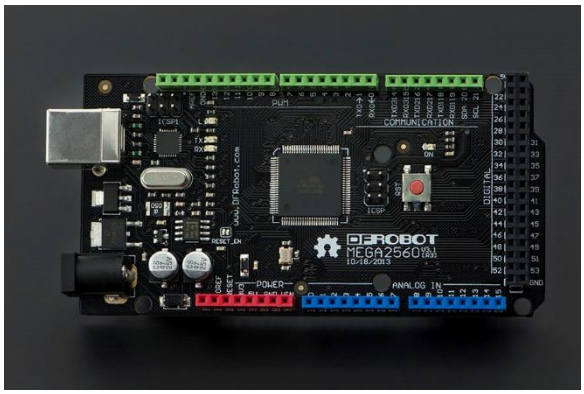


Figure 2. Arduino Mega

Description:

Arduino is an open-source physical computing platform based on a simple i/o board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software on your computer The open-source IDE can be downloaded for free (currently for Mac OS X, Windows, and Linux). The Arduino Mega is a microcontroller board based on the ATmega2560.

Arduino Structure:

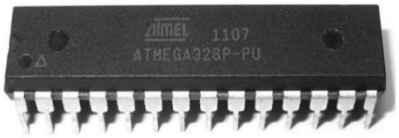


Figure 3. ATmega2560

Source	USB Connection	External Power Supply. External (non-USB) power can come either from an AC-to-DC adapter or battery.
Power Pins	VIN. The input voltage to the Arduino board when it's using an external power source	5V. The regulated power supply used to power the microcontroller and other components on the board. 3V3. A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA. GND. Ground pins.

Table II. Arduino Power

Note that the board can operate on an external supply of 6 to 20 volts.

Arduino Memories	Details	Information	Space
Flash memory	it is the program space where the Arduino sketch is stored	non-volatile	256k bytes (of which 8k is used for the bootloader)
SRAM	static random access memory) is where the sketch creates and manipulates variables when it runs	volatile	8k bytes
EEPROM	a read-only memory whose contents can be erased and reprogrammed using a pulsed voltage	non-volatile	4k byte

Table III. Arduino Memories

Arduino Inputs / Outputs	Pins Number	Function
Serial	0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX)	Used to receive (RX) and transmit (TX) TTL serial data
External Interrupts	2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3) and 21 (interrupt 2)	These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or change in the value
PWM	0 to 13	Provide 8-bit Pulse Width Modulation output with the analogWrite() function.
SPI	50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS)	These pins support SPI communication using the SPI library
LED	13	There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
AREF		Reference voltage for the analog inputs. Used with analogReference()
Reset		Bring this line LOW to reset the microcontroller

Table IV. Arduino Inputs / Outputs

Note that Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal analogReference() function.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

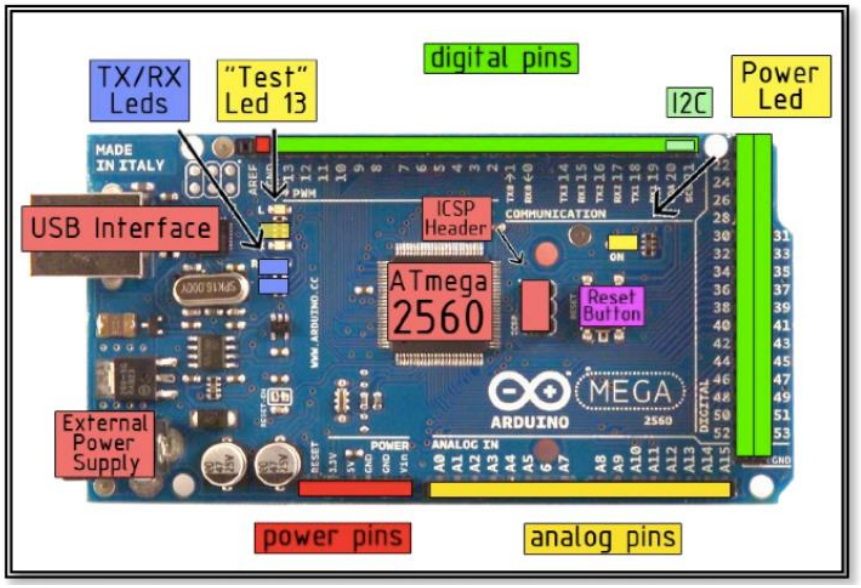


Figure 4. Arduino Mega Main Components and pins

→ Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Table V. Arduino Summary

3.2.2. ESP-E12 Wi-Fi Module

ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor. When ESP8266 hosts the application, and when it is the only application processor in the device, it is able to boot up directly from an external flash. It has integrated cache to improve the performance of the system in such applications, and to minimize the memory requirements.

Figure 5. WIFI Module

Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any microcontroller-based design with simple connectivity through UART (universal asynchronous receiver/transmitter) interface or the CPU AHB (Advanced High-performance Bus) bridge interface.

ESP8266 on-board processing and storage capabilities allow it to be integrated with the sensors and other application specific devices through its GPIOs (General-purpose input/output) with minimal development up-front and minimal loading during runtime. With its high degree of on-chip integration, which includes the antenna switch balun, power management converters, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

Sophisticated system-level features include fast sleep/wake context switching for energy efficient VoIP, adaptive radio biasing for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

→ Characteristics:

- ✓ 802.11 b / g / n
- ✓ Wi-Fi Direct (P2P), soft-AP
- ✓ Built-in TCP / IP protocol stack
- ✓ Built-in TR switch, balun, LNA, power amplifier and matching network
- ✓ Built-in PLL, voltage regulator and power management components
- ✓ 802.11b mode + 19.5dBm output power
- ✓ Built-in temperature sensor
- ✓ Support antenna diversity
- ✓ off leakage current is less than 10uA
- ✓ Built-in low-power 32-bit CPU: can double as an application processor
- ✓ SDIO 2.0, SPI, UART
- ✓ STBC, 1x1 MIMO, 2x1 MIMO
- ✓ A-MPDU, A-MSDU aggregation and the 0.4 Within wake
- ✓ 2ms, connect and transfer data packets
- ✓ standby power consumption of less than 1.0mW (DTIM3)

AT Commands "Attention Commands"

These commands are used to configure and initialize the ESP Wi-Fi Module at baud rate 57600.

All comments

Commands	Description	Type	Set/Execute	Inquiry	test	Parameters	Examples
AT+RST	restart the module	basic	-	-	-	-	
AT+CWMODE	wifi mode	wifi	AT+CWMODE=<mode>	AT+CWMODE?	AT+CWMODE=?	1= Sta, 2= AP, 3=both	
AT+CWJAP	join the AP	wifi	AT+ CWJAP =<ssid>,< pwd >	AT+ CWJAP?	-	ssid = ssid, pwd = wifi password	
AT+CWLAP	list the AP	wifi	AT+CWLAP				
AT+CWQAP	quit the AP	wifi	AT+CWQAP	-	AT+CWQAP=?		
AT+ CWSAP	set the parameters of AP	wifi	AT+ CWSAP= <ssid>, <pwd>,<chl>, <ecn>	AT+ CWSAP?		ssid, pwd, chl = channel, ecn = encryption	Connect to your router: : AT+CWJAP="YOURSSID","helloworld"; and check if connected: AT+CWJAP?
AT+ CIPSTATUS	get the connection status	TCP/IP	AT+ CIPSTATUS				
AT+CIPSTART	set up TCP or UDP connection	TCP/IP	1)single connection (+CIPMUX=0) AT+CIPSTART= <type>, <addr>,<port>; 2) multiple connection (+CIPMUX=1) AT+CIPSTART= <id> <type>,<addr>, <port>	-	AT+CIPSTART=?	id = 0-4, type = TCP/UDP, addr = IP address, port= port	Connect to another TCP server, set multiple connection first: AT+CIPMUX=1; connect: AT+CIPSTART=4,"TCP","X1.X2.X3.X4",9999
AT+CIPSEND	send data	TCP/IP	1)single connection(+CIPMUX=0) AT+CIPSEND= <length>; 2) multiple connection (+CIPMUX=1) AT+CIPSEND= <id>, <length>		AT+CIPSEND=?		send data: AT+CIPSEND=4,15 and then enter the data
AT+CIPCLOSE	close TCP or UDP connection	TCP/IP	AT+CIPCLOSE=<id> or AT+CIPCLOSE		AT+CIPCLOSE=?		
AT+CIFSR	Get IP address	TCP/IP	AT+CIFSR		AT+ CIFSR=?		
AT+ CIPMUX	set mutiple connection	TCP/IP	AT+ CIPMUX=<mode>	AT+ CIPMUX?		0 for single connection 1 for mutiple connection	
AT+ CIPSERVER	set as server	TCP/IP	AT+ CIPSERVER= <mode>[,<port>]			mode 0 to close server mode, mode 1 to open; port = port	turn on as a TCP server: AT+CIPSERVER=1,8888, check the self server IP address: AT+CIFSR=?

AT commands on our code "initialization of ESP8266"

```
Serial3.write("AT+RST\r\n", 2000); // reset module

Serial3.write("AT+CWMODE=2\r\n", 1000); // configure as access point

Serial3.write("AT+CIFSR\r\n", 1000); // get ip address

Serial3.write("AT+CIPMUX=1\r\n", 1000); // configure for multiple connections

Serial3.write("AT+CIPSERVER=1,80\r\n", 1000); // turn on server on port 80
```

3.2.3. Sensors

Analog Ambient Light Sensor

→ Description:

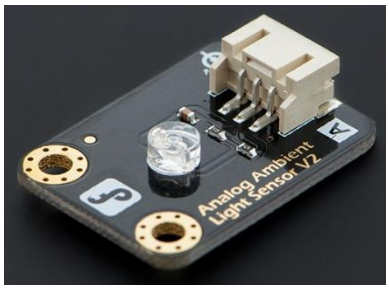


Figure 6 Light Sensor

It is a photodetector that converts light into current. This module detects the light density and reflects an analog voltage signal back to Arduino controller. It contains the environmental PT550 light sensor and it is more sensitive to light than normal one. It has three connections, ground, VCC and Vout which connected into the analog input of the Arduino.

→ Specification:

Supply Voltage: 3.3V to 5V

Interface: Analog

Size:22x30mm

→ Connection Diagram

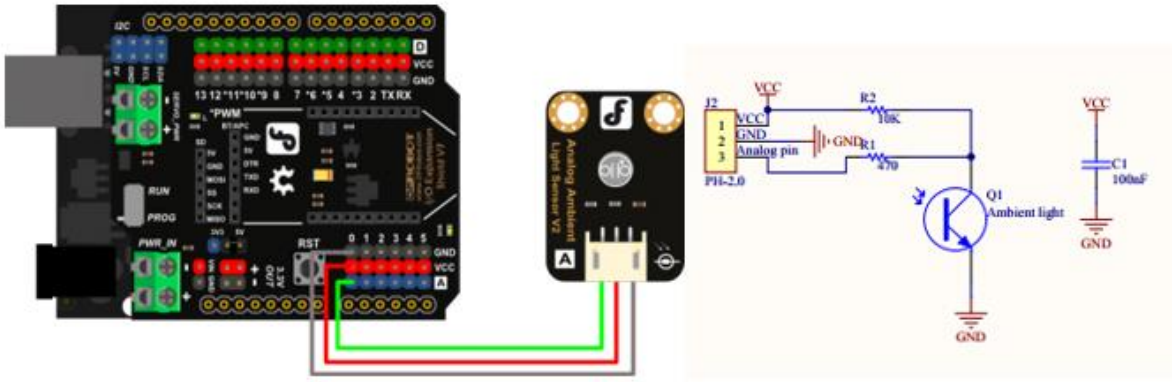


Figure 7. Light Sensor (Photodetector) Connected to Arduino

Gravity Analog 50A Current Sensor (ACDC) or ACS758 Current Sensor:

→ Description:

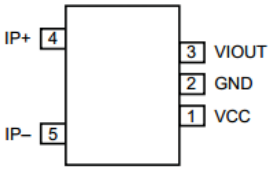
Figure 8. Current Senor

This is a breakout board for the fully integrated Hall Effect based linear ACS758 current sensor. The sensor gives precise current measurement for both AC and DC signals. The thickness of the copper conductor allows survival of the device at high overcurrent conditions. The ACS758 outputs an analog voltage output signal that varies linearly with sensed current. This current sensor enables users to monitor currents in their project every second for energy saving or circuit protecting purposes.

→ Features and Specifications:

- ✓ Arduino interface compatible
- ✓ 120 kHz typical bandwidth
- ✓ Rise time to set up the current = 3 s
- ✓ Output voltage proportional to AC or DC currents
- ✓ Extremely stable output offset voltage
- ✓ Nearly zero magnetic hysteresis
- ✓ Operating Voltage(analog): 5V
- ✓ Peak Measuring Voltage:3000V(AC),500V(DC)
- ✓ Current Measuring Rang: -50~50A
- ✓ Sensitivity:40 mV/A
- ✓ Operating Temperature: -40~150°C
- ✓ Size:34x34mm

→ Connections:



Terminal List Table

Number	Name	Description
1	VCC	Device power supply terminal
2	GND	Signal ground terminal
3	VIOUT	Analog output signal
4	IP+	Terminal for current being sampled
5	IP–	Terminal for current being sampled

Figure 9. Current senor Pins

Passive Infrared Sensor "PIR"

→Description:

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. Normally this type of sensor would be used as a motion sensor. It detects or "reads" infrared radiation "emitted" from objects all around it within an angle 110 degrees.

Figure 10. Passive Inferred Sensor

PIRs are called "passive" since they are not assisted by any helpers to do their work. It's purely based on what the sensor can pick up out from the surrounding environment, what's being emitted by objects. PIR has two half sensors and it looks on the difference between the values of these sensors to trigger motion. This is done in a smart way, to avoid false positives caused for example by a brief flash or an increase in room temperature.

→ Specification:

- ✓ Type: Digital
- ✓ Supply Voltage:3～5V
- ✓ Current:50Ma
- ✓ Working temperature:0℃～＋70℃
- ✓ Output level(HIGH):4V
- ✓ Output level(LOW):0.4V
- ✓ Detect angle:110 Degree
- ✓ Average detection distance:7 meters
- ✓ Size:28mm×36mm
- ✓ Weight:25g

→ Connection:

- ✓ 1 → Digital Signal Out
- ✓ 2 → VCC
- ✓ 3 → GND
- ✓ 4 → Jumper Selection: Repeatable trigger and unrepeatable trigger selection.
- ✓ H: Repeatable trigger which indicates continues reading (default setting)
- ✓ L: Unrepeatable trigger which indicates discrete reading.
- ✓ 5 → Potentiometer: To adjust trigger latency from 0.5s to 50s.

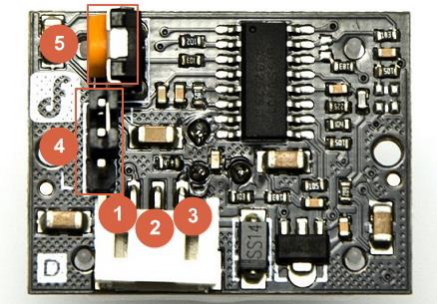


Figure 11. Labeled PIR

3.2.4. Beefcake Relay

Beefcake Relay

Description:

The Beefcake Relay Control Kit contains all the parts you need to get your high-power load under control. Only minimal assembly is required. The heart of the board is a sealed, SPDT 20A/10A Relay. The relay is controlled by 5V logic through a transistor, and an LED tells you when the relay is closed. This is a kit, so it comes as through-hole parts with assembly required, which makes for some nice soldering practice. Screw terminal connectors on either side of the board make it easy to incorporate into your project. There are some pretty beefy traces connecting the relay to the load pins, but the 3-pin terminals are only rated for 15A maximum. The beefcake relay has two main parts, the AC part and the control part. The AC part is where the source

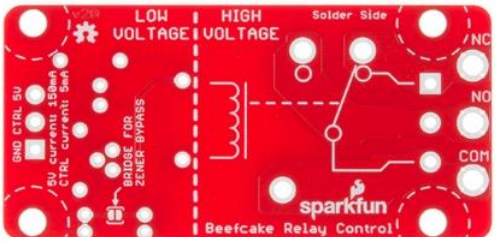


Figure 13. Beefcake Relay Structure

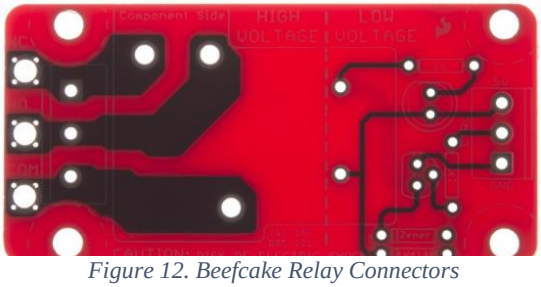


Figure 12. Beefcake Relay Connectors

and the load connected, this part is called the high voltage part. The other part is the low voltage part where VCC, ground and V-control are connected into the Arduino.

There is a lot of distance between the high voltage and low voltage sides. There's enough that you should be able to safely touch anything on the low voltage side, but it is preferring to avoid touching the relay while it is connected to the main because manufacturing has statistical failure rates.

Figure 14. Full Beefcake Relay

→ Features:

- ✓ Voltage Rating: 220VAC/28VDC
- ✓ VCC requirements: 4-6V, 150mA capable
- ✓ SPDT pins exposed (Form C)
- ✓ 14 AWG screw terminals for relay connections.
- ✓ 10 AWG solder lugs for relay connections.
- ✓ Flyback diode included
- ✓ Zener recovery diode included (decreases turn-off time)

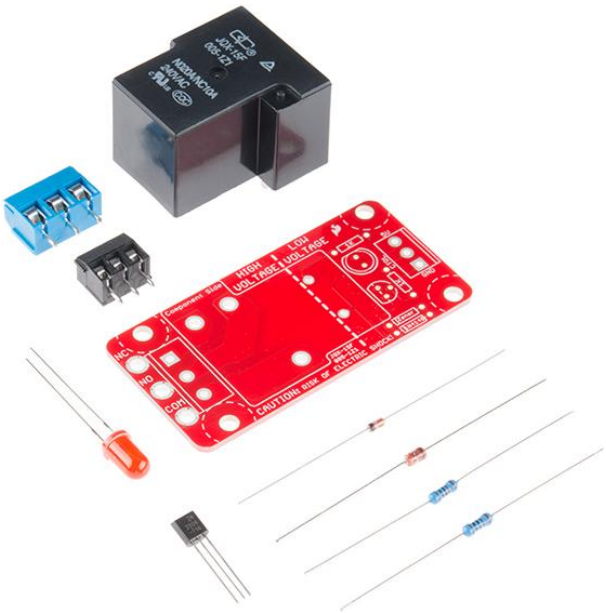


Figure 15. Beefcake Relay Components

Relay Assembly

The Beefcake Relay Control kit is relatively straight forward to assemble. This section outlines what tools will be needed and shows the assembly process.

The kit contains the following parts:

- ✓ An electromechanical Relay
- ✓ 2x Terminal blocks, light gauge for signal and heavy for output
- ✓ Coil-active LED
- ✓ Bipolar junction transistor (BJT)
- ✓ 2x Current limiting resistors
- ✓ Flyback arrestor diode (used to eliminate flyback, which is the sudden voltage spike seen across an inductive load when its supply current is suddenly reduced or interrupted.)
- ✓ Zener diode for discharge

The following tools are recommended:

- ✓ A soldering iron with 50 watts of power capability.
- ✓ Some solder, either leaded or lead-free.
- ✓ A magnifying glass or loupe.
- ✓ A vise to hold the PCB as you work.

→ Assembly Steps:

1- Start with the resistors. Bend the leads so they form 90 degree angles as close to the resistor body as possible.

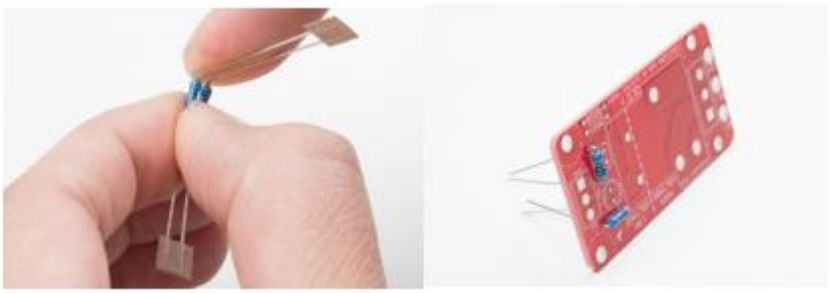


Figure 16. Fixing and Bending Resistor

2- Bend the diode leads. Insert the diodes. The fat diode is the 9.1V Zener. Place it over the silkscreen box “Zener” with the black band towards the white silkscreen stripe. The smaller diode is a standard high-speed diode. Place it over the box which is labeled N4148 with the black stripe towards the silk stripe.

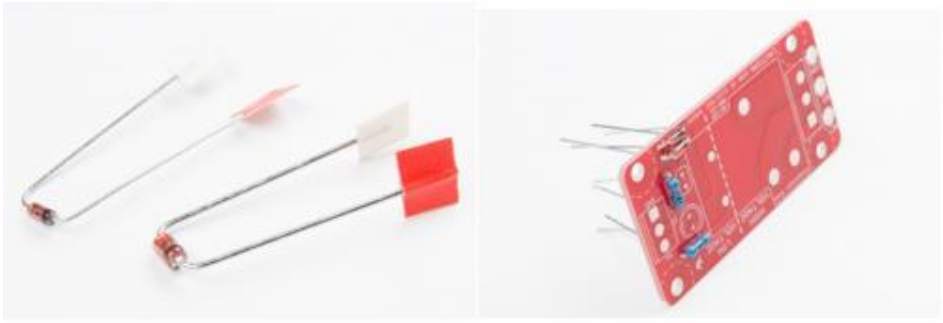


Figure 17. Connecting Diodes

3- Flip the board, and solder those components and remove the extra conductors.

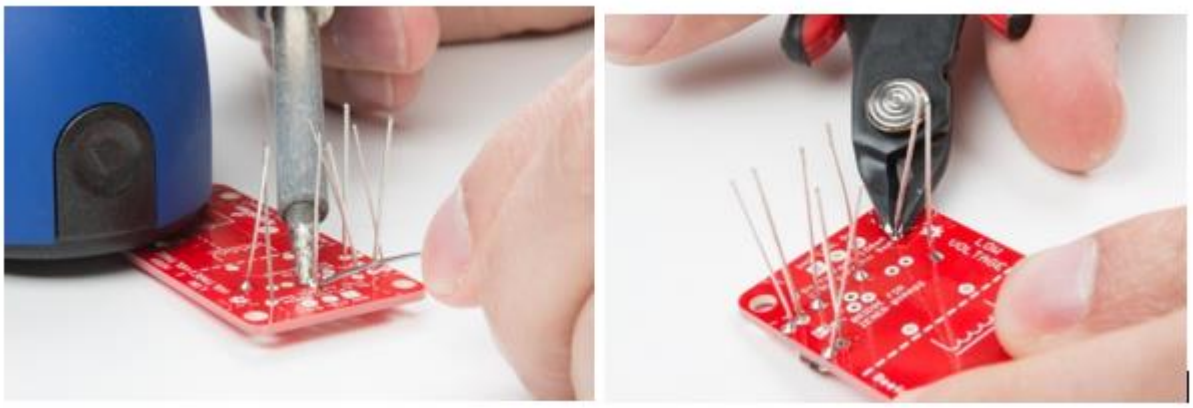


Figure 18. Soldering Components

4- Preparing the BJT. Leave the outer two legs straight and bend the inner. Fit the BJT and LED into the board, taking into consideration that the flat side of the LED matches the flat part of the silkscreen circle. Finally, solder them in place, and trim the leads.

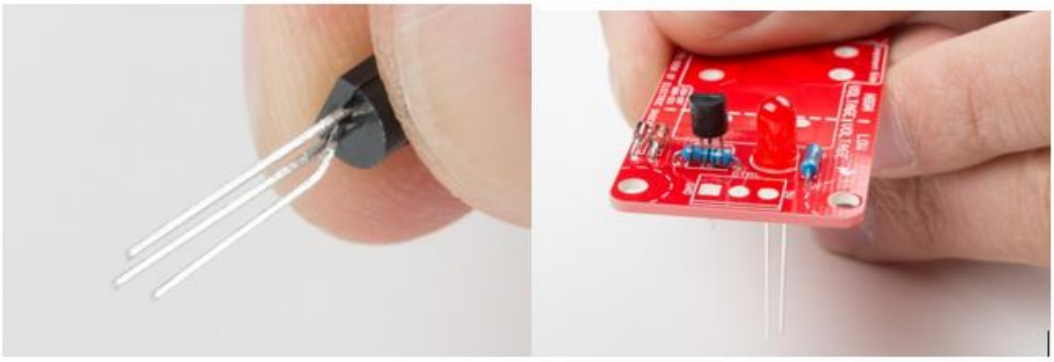


Figure 19. BJT and LED fixing

5- The last component is the relay. The relay has thick leads for current capacity, so they may need a little work getting inserted. So, the user should bond it correctly and solder it efficiently.

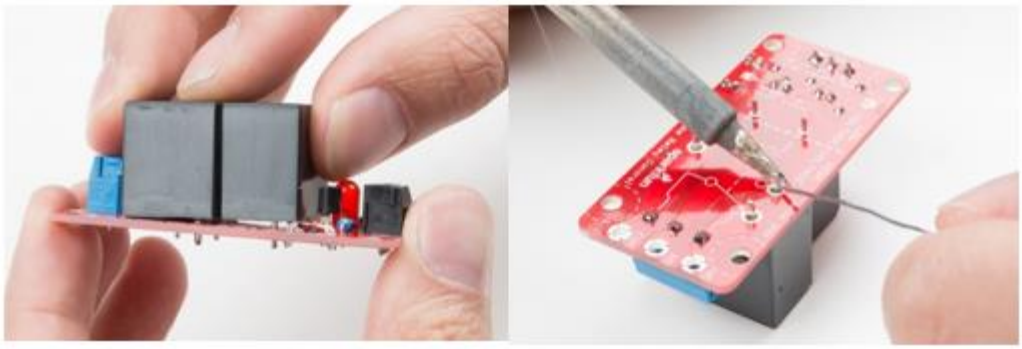


Figure 20. Fixing and soldering the relay

Connecting the relay into Arduino and into the main:

Connecting the relay into the Arduino is simple. The relay has a control signal pin in the middle that will be connected to a digital output pin in the Arduino. The other two pins is connected into the VCC and ground.

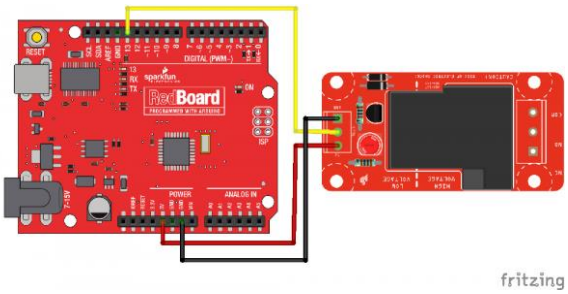


Figure 21. Relay connection to the Arduino

To connect the pins into the Arduino, user have to fix the wires using screws. On the other side, the high voltage side, two wires are connected into the relay from the main into the relay and then from the relay into the load. This relay is capable of holding two load, the first load is on and the second load is off when the relay is off and the first load is off and the second load is on when the relay is on. When the user uses the three pins in the high voltage side, he/ she should make the middle pin as the ground pin.

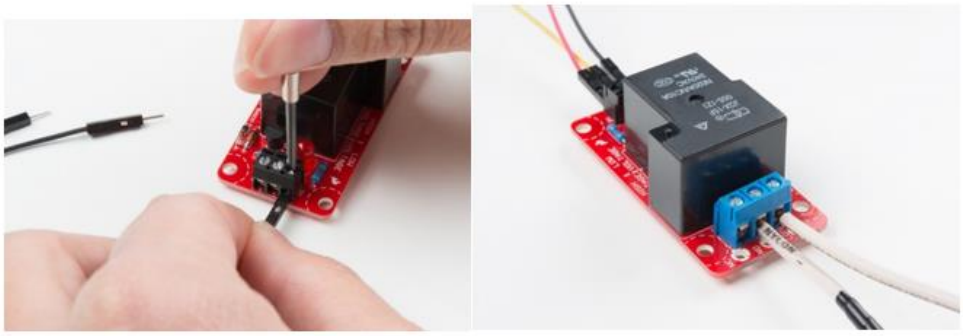


Figure 22. Connecting the Relay into Power and Control

Digital Push Button

Figure 23. Push Button Module

→ Description:

it is a digital push button acts as a switch. It has 3 connections which are the ground, VCC and Vout which is the control signal. When the button is pressed, a connection between Vout and VCC take place and a control signal is generated into the arduino.

→ Specification:

Supply voltage: 3.3V to 5V

Interface: Digital

Size: 22x30mm(0.87x1.18")

→ Connection:

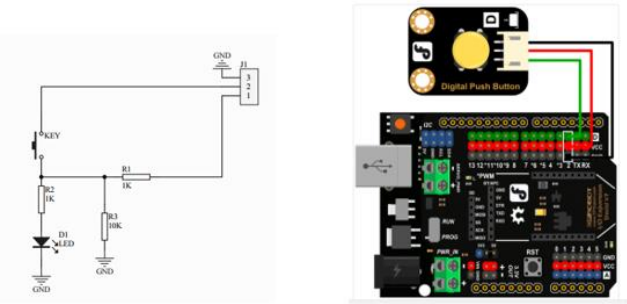


Figure 24. Button Circuit and its connection to the Arduino

IIC LCD1602(Arduino Compatible)

→ Description:

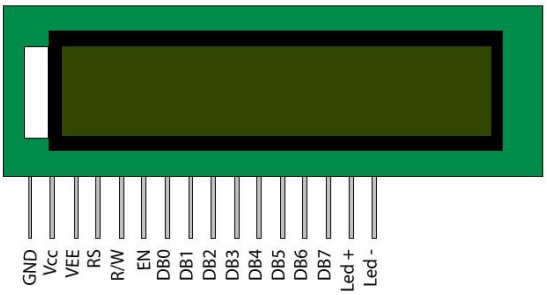


*Figure 25. 16*2 LCD screen*

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special and even custom characters (unlike in seven segments), animations and so on.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display and so on. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. The LCD has brightness and contrast levels can be specified by adjusting the potential-meter in the back of the screen.

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

Table VI. LCD Pins

→ Specification:

- ✓ I2C Address:0x20-0x27(0x20 default)
- ✓ Back lit (Blue with white char color)
- ✓ Supply voltage: 5V

- ✓ Interface: I2C/TWI x1, Gadgeteer interface x2
- ✓ Adjustable contrast
- ✓ Size:82x35x18 mm (3.2x1.4x0.7 in)

→ Connection

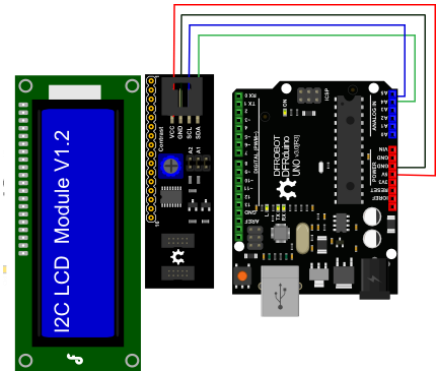


Figure 26. LCD connection to the Arduino

→Library Support Functions

- ✓ LiquidCrystal_I2C() //set the LCD address for a 16 chars and 2 line display
- ✓ init() //Initialization for the LCD
- ✓ clear() //clear display, set cursor position to zero
- ✓ home() //set cursor position to zero
- ✓ createChar() //Fill the first 8 CGRAM locations with custom characters
- ✓ setCursor() //set the position of the cursor
- ✓ cursor() //Turns the underline cursor on
- ✓ noCursor() //Turns the underline cursor off
- ✓ blink() //Turn on the blinking cursor
- ✓ noBlink() //Turn off the blinking cursor
- ✓ display() //Turn the display on(quickly)
- ✓ noDisplay() //Turn the display 0ff(quickly)
- ✓ backlight() //Turn the backlight on
- ✓ noBacklight() //Turn the backlight off
- ✓ scrollDisplayLeft() //Make the display scroll left without changing the RAM
- ✓ scrollDisplayRight() //Make the display scroll right without changing the RAM
- ✓ autoscroll() //This will 'right justify' text from the cursor
- ✓ noAutoscroll() //This will 'left justify' text from the cursor
- ✓ leftToRight() //This is for text that flows Left to Right
- ✓ rightToLeft() //This is for text that flows Right to Left

4. Design and Implementation

The design of this project involved connecting different hardware components and testing at different stages. As well as implementing both Android and net software projects.

4.1. Connecting different hardware components

first of all, we started by connecting the relays into the microcontroller. We wrote a code that send a signal into the output pin that is connected to the control pin of the beefcake relay after a certain delay in order to test the relays.

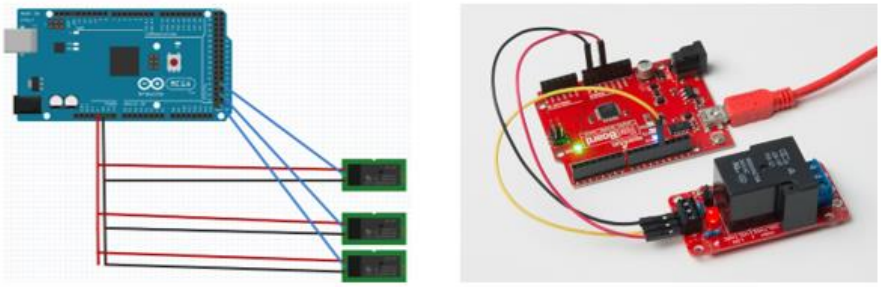


Figure 27. First Step

The second step was to add the push buttons into the circuit. We connect these buttons into the Arduino where each push button has three pins, VCC, GND and Vout which is the control signal. We wrote a program that receive a signal from the Vout pin and this signal will transfer into the control signal that will turn-on the relay. We had a problem in this section. The problem was that we had to keep pressing on the push button to keep the relay working. Then, we solve this problem via coding. We define in the code a previous state, which is the state of the relay initially. Then pressing the button will change only the state and preserve the new state.

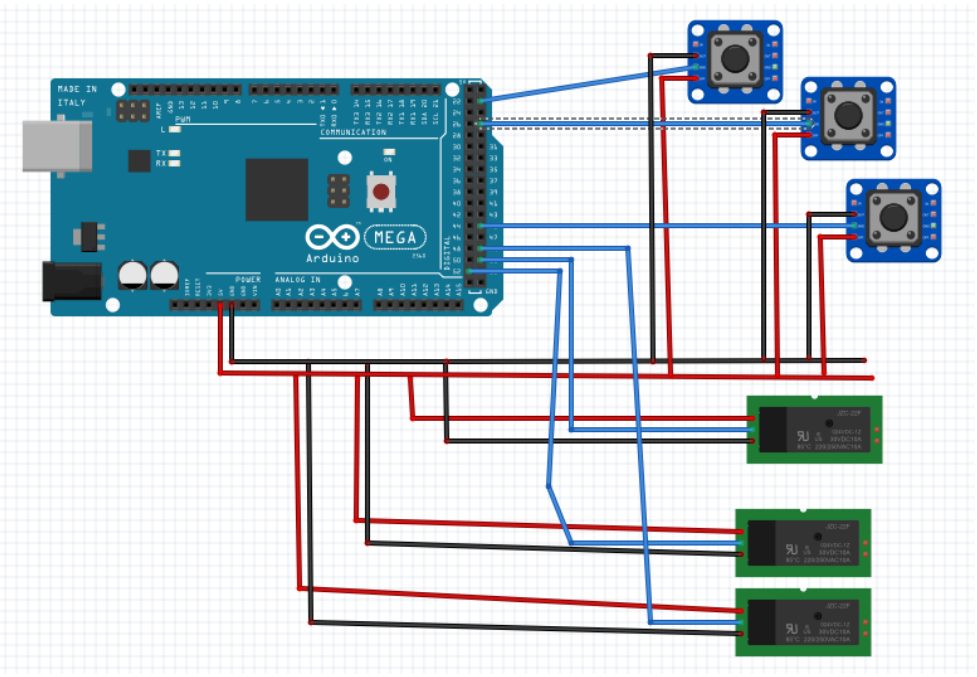


Figure 28. Second Step

The third step was connecting the current sensor and loads at each load after each relay. Therefore, we calibrated the current sensor based on the load measurements. Then we connect the circuit as shown in the diagram below. When we tried to run the code, the Arduino power decreases sharply and the Arduino turn off. So, we realized that current sensors consume large amount of power.

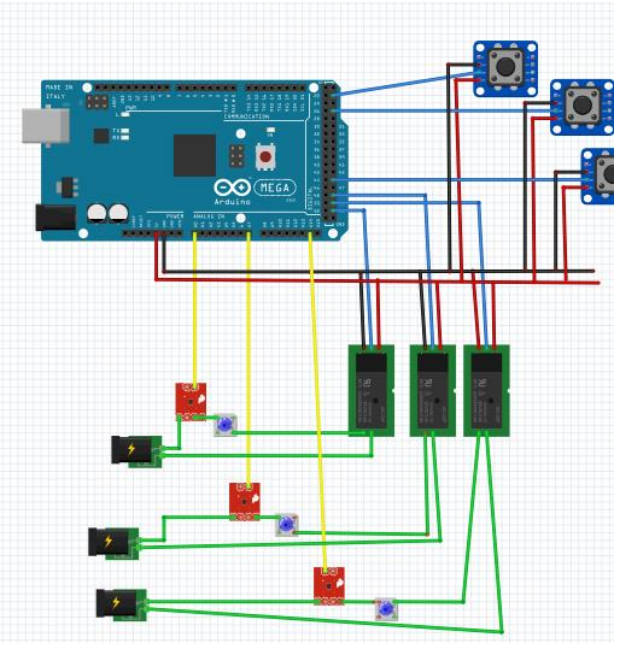


Figure 29. Third Step

Therefore, we solve this problem by connecting a 5-volt battery for each sensor as follow.

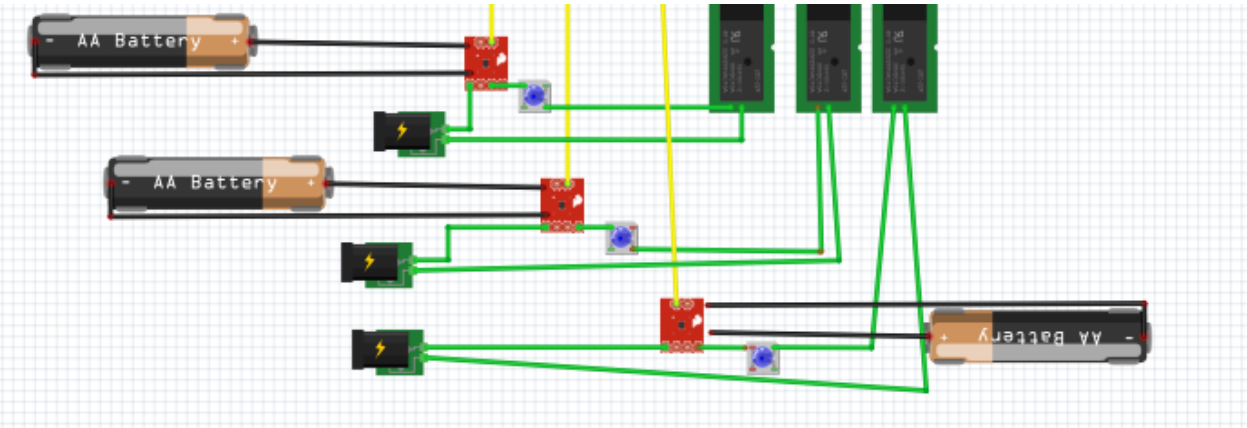


Figure 30. Battery added into Current Sensors

Finally, the code works and the analog readings appears on the computer at IDE software.

The fourth step was the most important step. In this step we communicate with the Arduino via mobile application. The module that satisfy this connection is ESP8266. This module acts as an access point that links between the mobile application and the Arduino. Therefore, the mobile application sends order into the ESP8266 and then it will transmit this order into the Arduino that will do an action. Also, the Arduino will send the analog reads into the ESP8266 which will store these that in its own server (serial 3). Therefore, when the mobile application needs any requirement data it will ask the ESP8266 to find it.

This process had two problems. The first problem was the selection of a Wi-Fi module. First we brought Wi-Fi Bee which was not suitable for this kind of communication. The Wi-Fi Bee needs two modules to communicate with each other's, not to communicate with a mobile application. So, we change the Wi-Fi Bee into ESP8266. The second problem is the initialization of ESP8266. The solution for this problem was the AT commands or the attention command which are commands specified in the code to identify the kind of communication, IP address, creating a server, set multiple connections and close the connection.

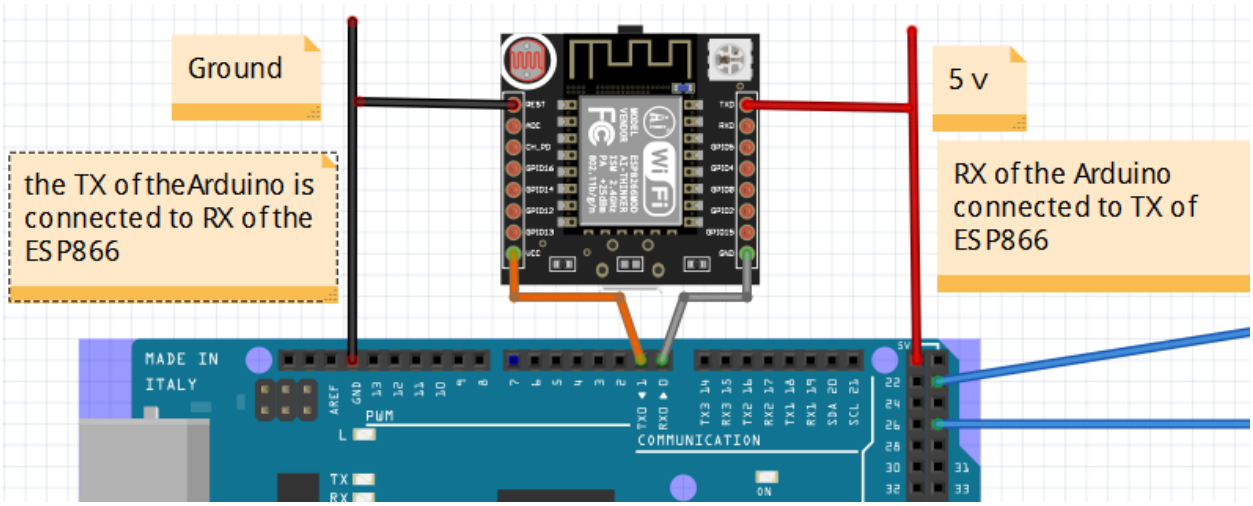


Figure 31. Fourth Step

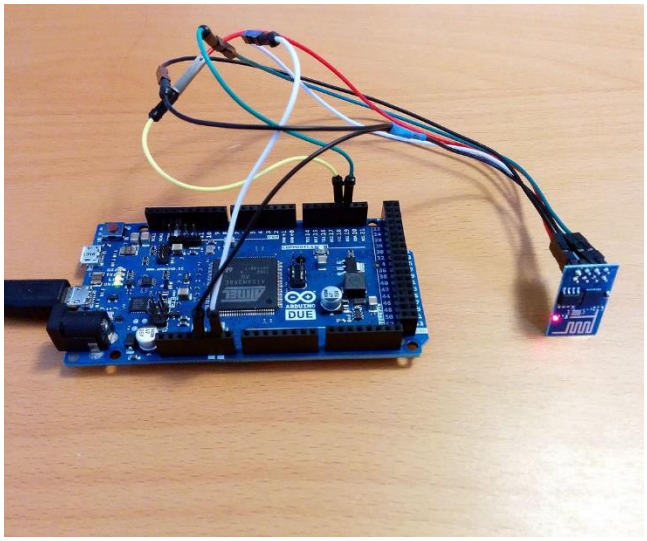


Figure 32. Connecting Arduino into ESP8266

The fifth step was connecting the PIR and light senor. Each element calibrated alone and connected through VCC, GND and input analog pin that reads the data from the surrounding medium. The light sensor calibrated by fixing a threshold value. If the light detected create voltage above the threshold value then it will work indicating Day mode, otherwise it will indicate night mode. Also, the PIR calibrated in the same way, however, this sensor detects

motion. So, if there is a motion then it will trigger day mode, otherwise, it will stay in the night mode.

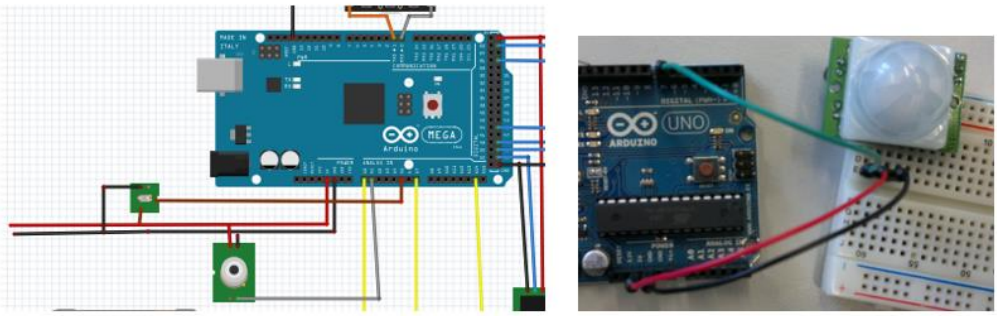


Figure 33. Fifth Step

The sixth step was connecting the LCD into the Arduino. This done according to the diagram below.

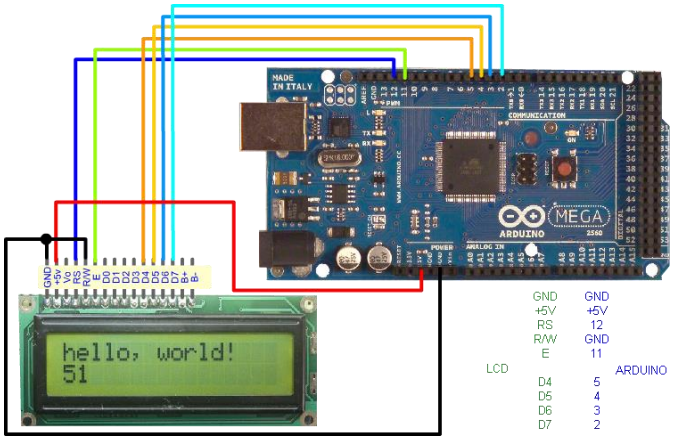


Figure 34. Sixth Step

The final, step was preparing the solar cells circuit. It is a simple circuit composed of solar panels, voltage regulator, battery and power inverted. The PV cells will charge the battery. However, since the solar energy is considering as backup energy then we need to store it in battery in regulated way. So, we need a regulator to avoid charging the battery with high voltage.

Then this DC power need to be converted into AC power in order to use it with an AC loads. This process done through a power inverter. Note that the power inverter has frequency equal to 50 Hz ± 3 Hz. Therefore, the relays will perform less efficiently.

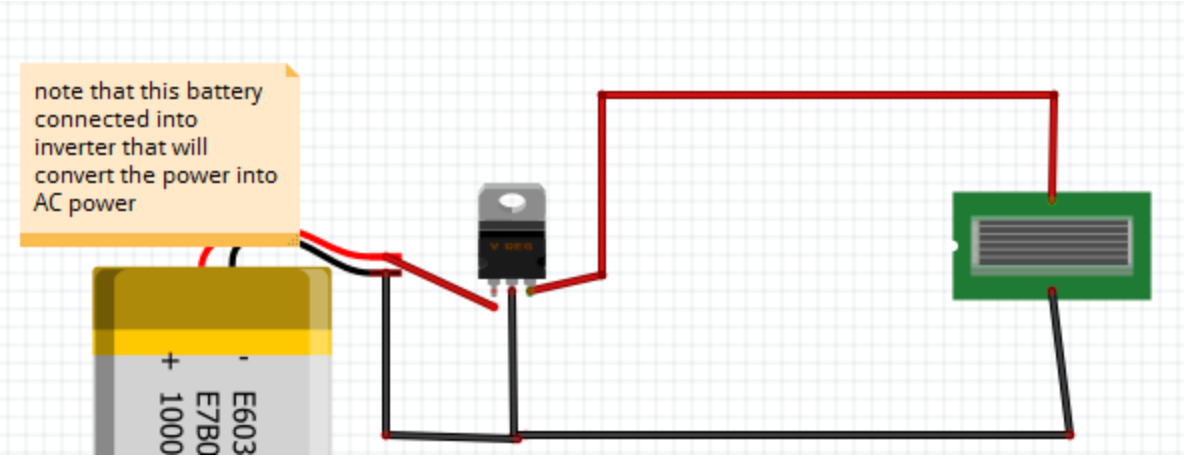


Figure 35. Solar Power

To sum up, the circuit below is the whole project.

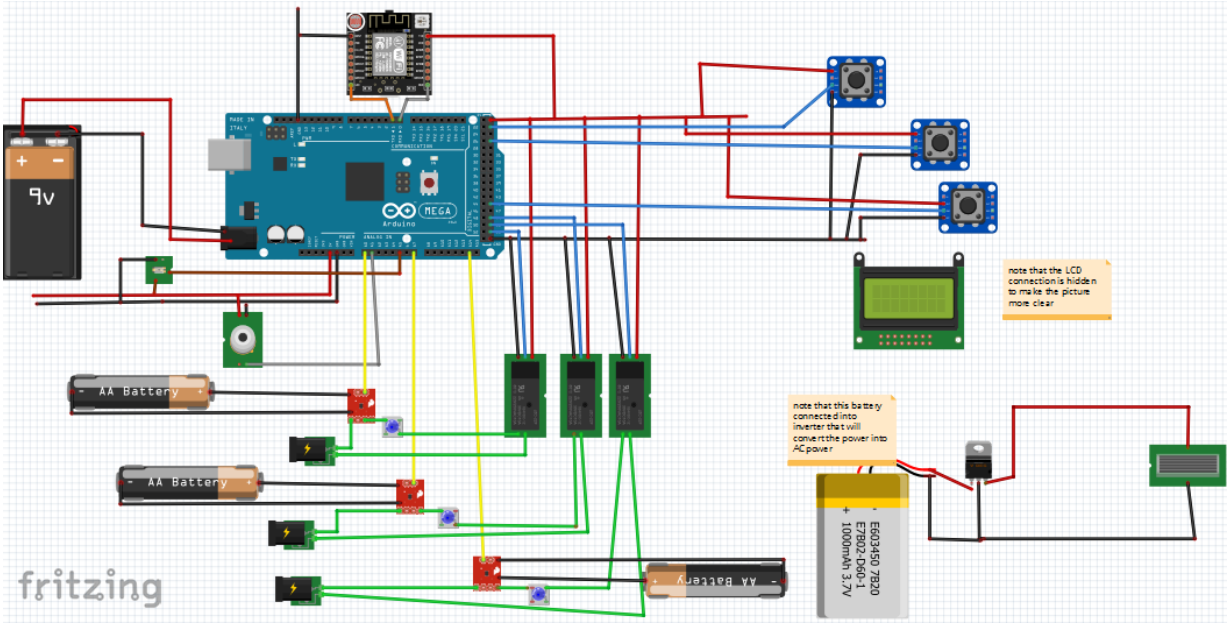


Figure 36. Final Project

4.2. Programming the Arduino Mega Microcontroller

Programming the Arduino Mega is done via Arduino IDE. The Arduino code is listed in the Appendix.

4.3. Implementing the Software Application

Android is the mobile operating system and working environment in which the mobile application was built. Android Nougat (Android 7.0) is the target version with SDK version 24, the software application is compatible with devices that have minimum SDK version 19 which is Android KitKat (Android 4.4). Android Studio is the IDE used for development of the android mobile application.

The mobile application called "Smart Home", gives the user multiple options, Manual Configuration, Power Consumption and switch mode. It also shows the user the current mode the home is operating in. the mobile application is given the permission to the internet so that it can communicate with the wifi module.

For the webApp, HTML5, CSS3 and jQuery where used. Microsoft Visual Studio .NET is the IDE for development of the asp.Net web application. The webApp gives simple user interface for basic functions to interact with home, mainly controlling home appliances and switching mode of operation.

The codes used for implementing both software applications are listed in the Appendix

4.4. Building power supply based on solar panels

Power Inverter:

→ Description:

It is a power inverter that inverts DC power from the battery into AC current.

→ Specification:

Frequency 50 Hz

Output Voltage 230 v

Figure 37 Power Inverter

Battery:

→ Description:

It is a rechargeable battery that are based on the lead-lead dioxide systems. It provides 12 volts and 7 amperes per hours

Figure 39 Lead Acid Battery

→ Specification:

- ✓ Nominal Voltage 12V
- ✓ Number of cell 6
- ✓ Design Life 5 years
- ✓ Nominal Capacity 77oF(25oC)
- ✓ 20 hour rate (0.35A, 10.5V) 7Ah
  5 hour rate (0.68A, 10.5V) 6.8Ah
  5 hour rate (1.13A, 10.5V) 5.65Ah
  1 hour rate (4.56A, 9.6V) 4.56Ah
- ✓ Internal Resistance: fully Charged battery 77 degree F(25 degree C) 28mOhms
- ✓ Self-Discharge: 3% of capacity declined per month at 20 degree C(on average)
- ✓ Operating Temperature Range
  Discharge -20~60oC
  Charge -10~60oC
  Storage -20~60oC

Solar Panel

→ Description:

Solar panel or a PV cells provide electricity by converting the solar power (radiated light) into electrical power (current). Therefore, it provides a clean energy. However, this energy is not reliable, especially when there is no light. Therefore, it used as a backup source when it connected into battery.

Figure 40 PV Cells

→ Specification:

- ✓ Working voltage: 9V
- ✓ Operating Current: 220mA
- ✓ Open circuit voltage: 9.6V
- ✓ Short circuit current: 500mA
- ✓ Power: 15W
- ✓ Performance: corrosion, moisture
- ✓ Size: Length 135mm X width 125mm

→ The connection between the solar panel, battery and inverter:

Figure 41. Getting Solar Power Diagram

Knowing that this charge controller is the Arduino where it will controller both overcharging and over discharging. It can controller overcharging by sensing the level of voltage (current sensor) at the output of the battery. If the battery above 90 % of its maximum voltage, then it will stop charging by sanding a control signal to make the relay open circuit. Otherwise, it will continue charging and the relay will stay short circuit. The same algorithm used to control over discharging but we sense the voltage level after the battery output. Also, the relay placed after the battery.

4.5. Testing the complete design

After all the hardware and software are done, several tests are made on the design to prove it working correctly and reliably.

Screen shot for the final circuit, the mobile application and the Web application are listed in Appendix.

5. Conclusion and Future Work

In this highly developing era, almost everything is becoming directly or indirectly dependent on computing and information technology. Our implementation of “Smart Home” proves to be economic, smart and efficient solution for home automation. It is obvious from this project that smart home automation can be cheaply made from low cost locally available components.

A smart home can be a simple grouping controls or heavy automated appliances that are controlled remotely. Ongoing costs include electricity to run the control systems, maintenance costs for the control and networking systems, including troubleshooting, and eventual cost of upgrading as standards change. Increased complexity may also increase maintenance costs for networked devices.

Future Work

This project represents a prototype of a smart home, which will be expanded to achieve more functionalities. The implemented project is a part of larger smart system.

Some features are to be integrated into this system later, including Smart Solar Management system which will gives the smart home the control over the power supply. The system will decide either to get energy directly from PV cells or from the battery charged by these PV cells. The decision making will mainly take into consideration the cost of each solution. Choosing the battery to be the source of energy will include extra cost which is the storage cost.

The Smart Home can be also integrated with Smart Grid via smart meter. This smart meter not only measures the total power consumption of the home as one entity but also measures the power consumption at each load. The ability to control lighting, appliances will become more important as Smart Grid initiatives are rolled out.

This Smart Home will be connected to the cloud. The cloud service will reduce IT cost, achieve scalability, gives access to automatic updates, and reduce energy consumption.

A security system will be integrated into this smart home.

Appendix

Appendix A: Source Codes

Appendix A-1: Arduino Source Code

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(9, 11, 26, 27, 29, 33);

const int analogIn = 15;

int mVperAmp = 185; // use 100 for 20A Module and 66 for 30A Module

int RawValue = 0;

int ACSoffset = 2500;

double Voltage = 0;

double Amps = 0;

double wattOut=0;


int pirpin = 12;              // choose the input pin (for PIR sensor)

int pirState = LOW;           // we start, assuming no motion detected

int val = 0;


//always high

int CH_PD_8266 = 53;


int photocellPin = 1;

int photocellReading;

int LEDbrightness;


int lamp1 = 6;

int lamp2 = 5;
```

```
int socket = 7;

int inPin = 8;
int inPin2 = 4;
int inPin3 = 3;

int all = 1;

int state = HIGH;
int reading;
int previous = LOW;

int state2 = HIGH;
int reading2;
int previous2 = LOW;

int state3 = HIGH;
int reading3;
int previous3 = LOW;

int state4 = HIGH;
int reading4;
int previous4 = LOW;
void setup() {
  Serial.begin(115200);
  Serial3.begin(115200);
  pinMode(lamp1, OUTPUT);
  pinMode(lamp2, OUTPUT);
```

```
  pinMode(socket, OUTPUT);

  pinMode(inPin, INPUT);
  pinMode(inPin2, INPUT);
  pinMode(inPin3, INPUT);
  pinMode(pirpin, INPUT);

  digitalWrite(lamp1, LOW);
  digitalWrite(lamp2, LOW);
  digitalWrite(socket, LOW);

  pinMode(CH_PD_8266, OUTPUT);
  digitalWrite(CH_PD_8266, HIGH);

  Serial3.write("AT+RST\r\n", 2000); // reset module
  Serial3.write("AT+CWMODE=2\r\n", 1000); // configure as access point
  Serial3.write("AT+CIFSR\r\n", 1000); // get ip address
  Serial3.write("AT+CIPMUX=1\r\n", 1000); // configure for multiple connections
  Serial3.write("AT+CIPSERVER=1,80\r\n", 1000); // turn on server on port 80

  lcd.begin(16, 2);
  // Print a message to the LCD.
}
void loop() {
  lcd.setCursor(0, 1);
  lcd.print("  Smart Home ");
  val = analogRead(pirpin);  // read input value
  if (val >= 940) {
```

```
  lcd.setCursor(1, 0);

  lcd.print("Motion Detected     ");
  //delay(100);
 } else {

  // lcd.setCursor(0, 1);

  //lcd.print("Door Closed    ");
  //delay(100);

 }
 RawValue = analogRead(analogIn);
 Voltage = (RawValue / 1024.0) * 5000; // Gets you mV
 Amps = ((Voltage - ACSoffset) / mVperAmp);

 double watt= (RawValue*Amps)/1000*-1;
 wattOut = wattOut +watt;

 //lcd.clear();
 //lcd.print("watt = ");
// lcd.print(watt);
// Serial.println(watt);
//Serial.print("Raw Value = " ); // shows pre-scaled value
//Serial.println(watt);
 lcd.setCursor(9, 1);
 lcd.print(wattOut);
```

```
  //Serial.print("\t mV = "); // shows the voltage measured
  //Serial.print(Voltage, 3); // the '3' after voltage allows you to display 3 digits after decimal point
  //Serial.print("\t Amps = "); // shows the voltage measured
  //Serial.println(Amps, 3); // the '3' after voltage allows you to display 3 digits after decimal point
  //delay(100);
  photocellReading = analogRead(photocellPin);
  if ( photocellReading >= 1000 ) {

    lcd.setCursor(0, 1);

    lcd.print("Night Mode ");
    delay(100);
  }
  if ( photocellReading <= 1000 ) {
    lcd.setCursor(0, 1);
    lcd.print("Day Mode ");
    delay(100);
  }
  // Serial.println(photocellReading);
  // LED gets brighter the darker it is at the sensor
  // that means we have to -invert- the reading from 0-1023 back to 1023-0
  photocellReading = 1023 - photocellReading;
  //now we have to map 0-1023 to 0-255 since thats the range analogWrite uses
  LEDbrightness = map(photocellReading, 0, 1023, 0, 255);
  //analogWrite(LEDpin, LEDbrightness);

  // print the number of seconds since reset:
```

```
  while (Serial.available() > 0) {
    String a = Serial3.readString();
    // Serial3.println(a);
    Serial.println(a);

  }

  reading = digitalRead(inPin);
  reading2 = digitalRead(inPin2);
  reading3 = digitalRead(inPin3);

//--------lamp 1----------
  if (reading == HIGH && previous == LOW ) {
    if (state == HIGH)
    {
      state = LOW;
      lcd.clear();
      // lcd.print("Lamp 1 OFF  ");
      all = 0;

    }
    else {
      state = HIGH;
      lcd.clear();
      lcd.print("Lamp 1 ON     ");
      all = 1;
```

```
    }
   }

   digitalWrite(lamp1, state);
   previous = reading;
 //--------lamp 2----------
   if (reading2 == HIGH && previous2 == LOW ) {
    if (state2 == HIGH)
    {
     state2 = LOW;
     lcd.clear();

     //lcd.print("Lamp 2 OFF  ");
     all = 0;

    }
    else {
     state2 = HIGH;
     lcd.clear();
     lcd.print("Lamp 2 ON  ");
     all = 1;
    }
   }
   digitalWrite(lamp2, state2);
 //--------lamp 3----------
   previous2 = reading2;

   if (reading3 == HIGH && previous3 == LOW ) {
```

```
    if (state3 == HIGH)
    {
      lcd.clear();

      // lcd.print("Socket OFF ");
      state3 = LOW;
      all = 0;
    }
    else {
      lcd.clear();

      lcd.print("Socket ON ");
      state3 = HIGH;
      all = 1;
    }

  }
  digitalWrite(socket, state3);
  previous3 = reading3;
  if (all == 0) {
    lcd.clear();
    // lcd.print(" ");
  }
}
void serialEvent3() {
  while (Serial3.available() > 0 ) {
    String s = Serial.readString();
    Serial3.println(s);
```

```
if (Serial3.find("parm=")) {
  String a = Serial3.readString();
  String parm1 = a.substring(0, 3);

  if (parm1 == "on1") {
    digitalWrite(lamp1, HIGH);
    lcd.clear();
    lcd.print("Lamp 1 ON");
    previous = LOW;
  }
  if (parm1 == "of1") {
    digitalWrite(lamp1, LOW);
    lcd.clear();

    lcd.print("Lamp 1 OFF");
    previous = LOW;
  }
  if (parm1 == "on2") {
    digitalWrite(lamp2, HIGH);
    lcd.clear();

    lcd.print("Lamp 2 ON");
    previous2 = LOW;

  }
  if (parm1 == "of2") {
    digitalWrite(lamp2, LOW);
    lcd.clear();
```

```
  lcd.print("Lamp 2 OFF");
  previous2 = LOW;

}
if (parm1 == "son") {
  digitalWrite(socket, HIGH);
  lcd.clear();

  lcd.print("Socket ON");
  previous3 = LOW;

}
if (parm1 == "sof") {
  digitalWrite(socket, LOW);
  lcd.clear();

  lcd.print("Socket OFF");
  previous3 = LOW;

}
if (parm1 == "mdd") {
  digitalWrite(lamp1, HIGH);
  delay(50);
  previous = LOW;
  digitalWrite(lamp2, HIGH);
  delay(50);
  previous2 = LOW;
```

```
      digitalWrite(socket, HIGH);
      previous3 = LOW;

      lcd.clear();
      lcd.print("DAY Mode ");
     }

    if (parm1 == "mdn") {
      digitalWrite(lamp1, LOW);
      delay(50);
      previous = LOW;
      digitalWrite(lamp2, LOW);
      delay(50);
      previous2 = LOW;
      digitalWrite(socket, LOW);
      previous3 = LOW;
      lcd.clear();
      lcd.print("NIGHT Mode ");
     }
   }
  }
}
```

Android Source Code

Java Folder:

Splash Screen Java Code:

```
public class SplashScreen extends AppCompatActivity {
    // Splash screen timer
    private static int SPLASH_TIME_OUT = 3000;
    public static final String MyPREFERENCES = "mode" ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);


        SharedPreferences sharedpreferences = getSharedPreferences(MyPREFERENCES,
Context.MODE_PRIVATE);


        SharedPreferences.Editor editor = sharedpreferences.edit();
        editor.putString("mode", "Night Mode");
        editor.commit();

        new Handler().postDelayed(new Runnable() {

            /*
             * Showing splash screen with a timer. This will be useful when you
             * want to show case your app logo / company
             */

            @Override
            public void run() {
                // This method will be executed once the timer is over
                // Start your app main activity
                Intent i = new Intent(SplashScreen.this, MainActivity.class);
                startActivity(i);

                // close this activity
                finish();
            }
        }, SPLASH_TIME_OUT);
    }
}
```

Main Activity Java Code:

```
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {
    Dialog dialog;
    TextView currentMode;
    public static final String MyPREFERENCES = "mode" ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        currentMode = (TextView) findViewById(R.id.currentMode);
    // Create custom dialog object
         dialog = new Dialog(MainActivity.this);


        // Strict mode for connection
        StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        // Restore preferences
        SharedPreferences settings = getSharedPreferences(MyPREFERENCES, 0);
        String mode = settings.getString("mode", "");
        currentMode.setText(mode);

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
```

```
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        if (id == R.id.nav_modes) {
            // Handle the camera action

            startActivity(new Intent(this.getApplicationContext(),Modes.class));
        } else if (id == R.id.nav_energy) {
            startActivity(new
Intent(this.getApplicationContext(),EnergyConsumption.class));
        } else if (id == R.id.nav_switch_mode) {


            // Include dialog.xml file
            dialog.setContentView(R.layout.custom_dialog);
            dialog.show();
            Button dayModeButton = (Button) dialog.findViewById(R.id.btn_day);
            Button nightModeButton = (Button) dialog.findViewById(R.id.btn_night);
```

```
            dayModeButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {

                    SharedPreferences sharedpreferences =
getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);


                    SharedPreferences.Editor editor = sharedpreferences.edit();
                    editor.putString("mode", "Day Mode");
                    editor.commit();
                    // Restore preferences
                    SharedPreferences settings = getSharedPreferences(MyPREFERENCES,
0);
                    String mode = settings.getString("mode", "Da");
                    currentMode.setText(mode);

                    changeMode("http://192.168.4.1/?parm=mdd");


                }
            });
            nightModeButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    SharedPreferences sharedpreferences =
getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);


                    SharedPreferences.Editor editor = sharedpreferences.edit();
                    editor.putString("mode", "Night Mode");
                    editor.commit();
                    // Restore preferences
                    SharedPreferences settings = getSharedPreferences(MyPREFERENCES,
0);
                    String mode = settings.getString("mode", "Da");
                    currentMode.setText(mode);

                    changeMode("http://192.168.4.1/?parm=mdn");

                }
            });

        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
    public void changeMode(String urll) {



        AsyncHttpClient client = new AsyncHttpClient();
        client.setTimeout(2000);
        client.setResponseTimeout(10);
        client.get("urll", new AsyncHttpResponseHandler() {

            @Override
            public void onStart() {
                // called before request is started
```

```
            }

            @Override
            public void onSuccess(int statusCode, Header[] headers, byte[]
responseBody) {

            }

            @Override
            public void onFailure(int statusCode, Header[] headers, byte[]
responseBody, Throwable error) {

            }


            @Override
            public void onRetry(int retryNo) {
                // called when request is retried
            }

            @Override
            public void onFinish() {
                super.onFinish();
                dialog.dismiss();
            }
        });


        try {
            HttpGet httpGet = new HttpGet(urll);
            HttpParams httpParameters = new BasicHttpParams();
            // Set the timeout in milliseconds until a connection is established.
            // The default value is zero, that means the timeout is not used.
            int timeoutConnection = 3000;
            HttpConnectionParams.setConnectionTimeout(httpParameters,
timeoutConnection);
            // Set the default socket timeout (SO_TIMEOUT)
            // in milliseconds which is the timeout for waiting for data.
            int timeoutSocket = 10;
            HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);

            DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);
            HttpResponse response = httpClient.execute(httpGet);
        } catch (ConnectTimeoutException e) {
            //Here Connection TimeOut excepion

        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

Modes Java File:

```
public class Modes extends AppCompatActivity {
    public static final String MyPREFERENCES = "mode" ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_modes);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        // Restore preferences
        SharedPreferences settings = getSharedPreferences(MyPREFERENCES, 0);
        String mode = settings.getString("mode", "Da");
        TabLayout tabLayout = (TabLayout) findViewById(R.id.tab_layout);
        tabLayout.addTab(tabLayout.newTab().setText(mode));

        tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);

        final ViewPager viewPager = (ViewPager) findViewById(R.id.pager);
        final PagerAdapter adapter = new PagerAdapter
                (getSupportFragmentManager(), tabLayout.getTabCount());
        viewPager.setAdapter(adapter);
        viewPager.addOnPageChangeListener(new
TabLayout.TabLayoutOnPageChangeListener(tabLayout));
        tabLayout.setOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                viewPager.setCurrentItem(tab.getPosition());
            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {

            }

            @Override
            public void onTabReselected(TabLayout.Tab tab) {

            }
        });
    }

}
```

Power Consumption Java Code:

```
public class EnergyConsumption extends AppCompatActivity {

    EditText pricePerDollar;
    TextView energyConsumed,costPerDay,costPerMonth;
    Button calculate;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_energy_consumption);

        pricePerDollar = (EditText)findViewById(R.id.kwh_per_hour_cost);
        energyConsumed = (TextView) findViewById(R.id.energy_consumed);
        costPerDay = (TextView) findViewById(R.id.cost_per_day);
        costPerMonth = (TextView) findViewById(R.id.cost_per_month);
        calculate = (Button) findViewById(R.id.calculate);



        calculate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                double price =Double.parseDouble( pricePerDollar
```

```
.getText().toString());
                double costPday= Math.round ((Double.parseDouble(
energyConsumed.getText().toString())/1000)*24)*price;
                costPerDay.setText(costPday +" $");
                costPerMonth.setText(Math.round(costPday*30 )+" $");
            }
        });


    }
}
```

Manual Configuration Java Code:

```
public class DayMode extends Fragment {


    Switch lamp1switch, lamp2switch,switch_;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View rootview = inflater.inflate(R.layout.day_mode, container, false);

        lamp1switch = (Switch) rootview.findViewById(R.id.lamp1switch);
        lamp2switch = (Switch) rootview.findViewById(R.id.lamp2switch);
        switch_ = (Switch) rootview.findViewById(R.id.switch_);


        StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);



        lamp1switch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
                // do something, the isChecked will be
                // true if the switch is in the On position
                if(isChecked) {
                    turnLamp("http://192.168.4.1/?parm=on1");

                }
                else {
                    turnLamp("http://192.168.4.1/?parm=of1");
                }

            }
        });

        lamp2switch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
                // do something, the isChecked will be
                // true if the switch is in the On position
                if(isChecked) {
                    turnLamp("http://192.168.4.1/?parm=on2");
                }
                else {
```

```
                    turnLamp("http://192.168.4.1/?parm=of2");
                }
            }
        });
        switch_.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
                // do something, the isChecked will be
                // true if the switch is in the On position
                if(isChecked) {
                    turnLamp("http://192.168.4.1/?parm=son");

                }
                else {
                    turnLamp("http://192.168.4.1/?parm=sof");
                }

            }
        });

        return rootview;
    }

    public void turnLamp(String urll) {



        AsyncHttpClient client = new AsyncHttpClient();
        client.setTimeout(2000);
        client.setResponseTimeout(10);
        client.get("urll", new AsyncHttpResponseHandler() {

            @Override
            public void onStart() {
                // called before request is started
            }

            @Override
            public void onSuccess(int statusCode, Header[] headers, byte[]
responseBody) {

            }

            @Override
            public void onFailure(int statusCode, Header[] headers, byte[]
responseBody, Throwable error) {

            }


            @Override
            public void onRetry(int retryNo) {
                // called when request is retried
            }
        });


        try {
            HttpGet httpGet = new HttpGet(urll);
            HttpParams httpParameters = new BasicHttpParams();
            // Set the timeout in milliseconds until a connection is established.
```

```
            // The default value is zero, that means the timeout is not used.
            int timeoutConnection = 3000;
            HttpConnectionParams.setConnectionTimeout(httpParameters,
timeoutConnection);
            // Set the default socket timeout (SO_TIMEOUT)
            // in milliseconds which is the timeout for waiting for data.
            int timeoutSocket = 10;
            HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);

            DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);
            HttpResponse response = httpClient.execute(httpGet);
        } catch (ConnectTimeoutException e) {
            //Here Connection TimeOut excepion

        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

Page Adapter Fragment Java Code:

```
public class PagerAdapter extends FragmentStatePagerAdapter {
    int mNumOfTabs;

    public PagerAdapter(FragmentManager fm, int NumOfTabs) {
        super(fm);
        this.mNumOfTabs = NumOfTabs;
    }

    @Override
    public Fragment getItem(int position) {

        switch (position) {
            case 0:
                DayMode tab1 = new DayMode();
                return tab1;

            default:
                return null;
        }
    }

    @Override
    public int getCount() {
        return mNumOfTabs;
    }
}
```

XML Layout Folder:

Splash Activity Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_splash_screen"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    tools:context="com.devteam.arduinoproject.SplashScreen">
    <ImageView
        android:id="@+id/imgLogo"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_centerInParent="true"
        android:scaleType="fitCenter"
        android:src="@mipmap/home" />
</RelativeLayout>
```

Main Activity Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@mipmap/ee"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

App bar Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.devteam.arduinoproject.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
```

```
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

</android.support.design.widget.CoordinatorLayout>
```

Activity Modes Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_modes"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.devteam.arduinoproject.Modes">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="?attr/colorPrimary"
        android:elevation="6dp"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/toolbar"
        android:background="?attr/colorPrimary"
        android:elevation="6dp"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>

    <android.support.v4.view.ViewPager
        android:id="@+id/pager"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:layout_below="@id/tab_layout"/>

</RelativeLayout>
```

Main Fragment Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/content_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
```

```
    tools:context="com.devteam.arduinoproject.MainActivity"
    tools:showIn="@layout/app_bar_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_above="@+id/currentMode"
        android:textColor="@android:color/black"
        android:text="Current Mode" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:id="@+id/currentMode"
        android:textSize="20sp"
        android:textStyle="bold"
        android:textColor="@android:color/holo_red_dark"
        android:text="Day Mode" />
</RelativeLayout>
```

Switch Modes Dialog Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#3E80B4"
    android:orientation="vertical"
    android:padding="10dp">

    <TextView
        android:id="@+id/txt_dia"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="10dp"
        android:text="Choose Mode"
        android:textAlignment="center"
        android:textColor="@android:color/white"
        android:textSize="20dp"
        android:textStyle="bold" />


    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:background="#3E80B4"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btn_day"
            android:layout_width="150dp"
            android:layout_height="40dp"
            android:background="@android:color/white"
            android:clickable="true"
            android:text="Day Mode"
            android:textColor="#5DBCD2"
            android:textStyle="bold" />
```

```
        <Button
            android:id="@+id/btn_night"
            android:layout_width="150dp"
            android:layout_height="40dp"
            android:layout_marginLeft="5dp"
            android:background="@android:color/white"
            android:clickable="true"
            android:text="Night Mode"
            android:textColor="#5DBCD2"
            android:textStyle="bold" />
    </LinearLayout>

</LinearLayout>
```

Navigation Header Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="@dimen/nav_header_height"
    android:background="@drawable/side_nav_bar"
    android:gravity="bottom"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">


    <TextView
        android:text="Smart Home"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/textView2" />
</LinearLayout>
```

Manual Configuration Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

    >

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:layout_height="wrap_content">

        <Switch
            android:id="@+id/lamp1switch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:switchMinWidth="100dp"
```

```
            android:textSize="20sp"
            android:textStyle="bold"
            android:text="Lamp 1           " />

        <Switch
            android:id="@+id/lamp2switch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignEnd="@+id/lamp1switch"
            android:layout_below="@+id/lamp1switch"
            android:layout_marginTop="20dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:switchMinWidth="100dp"
            android:text="Lamp 2           " />

        <Switch

            android:id="@+id/switch_"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:layout_alignEnd="@+id/lamp2switch"
            android:textSize="20sp"
            android:textStyle="bold"

            android:layout_below="@+id/lamp2switch"
            android:switchMinWidth="100dp"
            android:text="Switch            " />
    </RelativeLayout>
</RelativeLayout>
```

Energy Consumption Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_energy_consumption"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context="com.devteam.arduinoproject.EnergyConsumption">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:orientation="horizontal">


        <TextView
            android:id="@+id/textView"
            android:layout_width="200dp"
            android:layout_height="50dp"

            android:text="Energy Consumption"
            android:textColor="#000000"
            android:gravity="center_vertical"
            android:textSize="20sp"
            android:textStyle="bold" />
```

```
    <LinearLayout

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <EditText
            android:id="@+id/energy_consumed"
            android:layout_width="70dp"
            android:layout_height="50dp"
            android:layout_marginBottom="10dp"
            android:gravity="center"
            android:text="100"
            android:textSize="20sp" />

    </LinearLayout>
    <TextView
        android:id="@+id/dasasd"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_marginBottom="10dp"
        android:gravity="center"
        android:layout_marginLeft="10dp"
        android:text="Watt"
        android:textSize="20sp" />

</LinearLayout>


<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="left"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/sdasd"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="Rating"

        android:textColor="#000000"
        android:gravity="center_vertical"
        android:textSize="20sp"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/asd"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_marginBottom="10dp"
        android:textAlignment="center"
        android:gravity="center"
        android:text=" 1 KW /hr ="
        android:textSize="20sp" />

    <EditText
        android:id="@+id/kwh_per_hour_cost"
        android:layout_width="70dp"
        android:layout_height="50dp"
        android:ems="10"
        android:textSize="20sp"
        android:gravity="center"
```

```
            android:layout_marginBottom="10dp"
            android:text="0"
            android:inputType="numberDecimal" />

        <TextView
            android:id="@+id/asdsd"
            android:layout_width="wrap_content"
            android:layout_height="50dp"
            android:layout_marginBottom="10dp"
            android:gravity="center"
            android:text="US $"
            android:textSize="20sp" />

    </LinearLayout>


    <Button
        android:text="Calculate"
        android:layout_gravity="center_horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/calculate" />
    <LinearLayout

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/costeee"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:text="Energy Cost/day"
            android:textColor="#000000"
            android:textSize="20sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/cost_per_day"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="70dp"
            android:text="120 $"
            android:textColor="#ff3333"
            android:textSize="20sp" />
    </LinearLayout>


    <LinearLayout

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_gravity="right"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/fgfshrt"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:text="Energy Cost/Month"
            android:textColor="#000000"
```

```
            android:textSize="20sp"

            android:textStyle="bold" />

        <TextView
            android:id="@+id/cost_per_month"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="70dp"
            android:layout_gravity="right"
            android:text="120 $"
            android:textColor="#ff3333"
            android:textSize="20sp" />
    </LinearLayout>
</LinearLayout>
```

ASP.NET Project Source Codes:

HTML5 Source Code:

```
<!DOCTYPE html5>
<html>
<head>
<meta charset="UTF-8">
<title>Smart Home</title>

<link href="css/style.css" rel="stylesheet" type="text/css">

</head>
<body>

<div id="wrapper">

  <div id="header">

    <div class="top_banner">
      <h1>Smart Home</h1>
      <p>Stay Connected</p>
    </div>

  </div>

  <div id="page_content">

    <div class="navigation">
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>

      </ul>
    </div>

    <div class="left_side_bar">

      <div class="col_1">
        <h1>Configuration</h1>

          <ul>
            <li><a  href="http://192.168.4.1/?parm=on1">Turn ON Lamp 1</a></li>
```

```
            <li><a  href="http://192.168.4.1/?parm=of1">Turn OFF Lamp 1</a></li>
                  <li><a  href="http://192.168.4.1/?parm=on2">Turn ON Lamp 2</a></li>
            <li><a  href ="http://192.168.4.1/?parm=of2">Turn OFF Lamp 2</a></li>
                  <li><a  href="http://192.168.4.1/?parm=son">Turn ON Switch</a></li>
            <li><a  href="http://192.168.4.1/?parm=sof">Turn OFF Switch</a></li>
            </ul>


        </div>
      </div>

      <div class="col_1">
        <h1>Modes</h1>
         <ul>
              <li><a  href="http://192.168.4.1/?parm=mdd">Day Mode</a></li>

            <li><a   href="http://192.168.4.1/?parm=mdn">Night Mode</a></li>
        </ul>

        </div>
      </div>



    <div class="right_section">
      <div class="common_content">
        <h2>Description</h2>
        <hr>
        <p>Select the action you want to take from the Configuration menu.
                You can either turn on or turn off any appliance you have connection to
in your home.</p>
               <p>Select the mode of operation of your home from the Modes menu.</p>
        <br>




        </div>

    </div>

    <div class="clear"></div>


    <div id="footer">Mariam Al Sadek, Hasan Beqai<br>

    <a><img src="images/footer.gif" class="copyright" ></a></div>


  </div>

</body></html>
```

CSS3 Source Code:

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote,
pre, a, abbr, acronym, address, big, cite, code, del, dfn, em, font, img, ins, kbd, q, s,
samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li,
fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr, th, td, hr,
button {
	margin:0;
	padding:0;
	border:0;
	outline:0;
	font-size:100%;
	vertical-align:baseline;
	background:none;
}

ol, ul {
	list-style-type:none;
}

h1, h2, h3, h4, h5, h6, li {
	line-height:100%;
}

blockquote, q {
	quotes:none;
}

q:before, q:after {
	content:"";
}

input, textarea, select {
	font:12px Arial, Helvetica, sans-serif;
	vertical-align:middle;
	padding:0;
	margin:0;
}

textarea {
	overflow:auto;
	resize:none;
}

body {
	margin:0px;
	padding:0px;
	font-size:13px;
	line-height:21px;
	color:#113e5d;
	background-color:#9ba0a3;
	font-family:Georgia, "Times New Roman", Times, serif;
}

h1, h2, h3, h4, h5, h6 {
	font-weight:normal;
}

h1 {
	font-size:33px;
```

```
	margin-bottom:10px;
}

h2 {
	font-size:25px;
	margin-bottom:10px;
}

h3 {
	font-size:18px;
	margin-bottom:10px;
}

h4 {
	font-size:16px;
	margin-bottom:10px;
}

h5 {
	font-size:14px;
	margin-bottom:10px;
}

h6 {
	font-size:12px;
	margin-bottom:10px;
}


	*****************************/
	a {color: #1B5219;text-decoration: underline;}
	a:hover {color: #1B5219;text-decoration: none;}

	*****************************/
	sup, sub {vertical-align: baseline;position: relative;top: -0.4em;}
	sub { top: 0.4em; }

	ul, ol {padding:0;margin:0;}
	ul li {list-style-type:disc;}
	ol li {list-style-type:decimal;}
	ul li,
	ol li {font-size: 12px; line-height: 16px;margin: 0 0 10px 15px;}

	table,th,td {border:1px solid #AAAAAA;}
	th {padding:3px; font-weight:bold;color:#344b3b;}
	tr {margin:0 0 5px 0;padding:3px;}
	td {padding:5px;margin:0 1px;}

	hr {border-bottom: 1px solid #C1C1CB;margin:5px 0;padding:0;height:2px;width:100%;
margin-bottom:15px;}

.clear {
	clear:both;
}

.left {
	float:left;
}
```

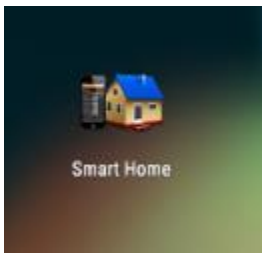
```
.right {
	float:right;
}

.copyright {
	border: 0px;
	height: 1px;
	width: 1px;
}

#wrapper {
	width:1000px;
	margin:0 auto;
	padding:20px 0 10px 0;
}
#header {
	padding:0px 0px 5px;
}

#header .top_banner{
	background:url(../images/header.jpg) no-repeat left top;
	height:175px;
	padding:75px 95px 0 295px;
	border:#496066 solid 1px;
}

#header h1{
	font-size:32px;
	color:#000000;
	margin:0px;
	font-family:Georgia, "Times New Roman", Times, serif;
	font-weight:bold;
	padding:0px 0px 10px;
}

#header h1 a {
	color:#000000;
	text-decoration: none;
}

#header h1 a:hover {
	text-decoration: none;
}

#header p{
	font-size:16px;
	margin-left:1px;
	color:#000000;
	font-family:Georgia, "Times New Roman", Times, serif;
}

.navigation{
	margin:0px 0px 25px;
	border-bottom:#adb1b2 solid 1px;
	padding:15px 5px;
	background: #e4e6e5;
}
```

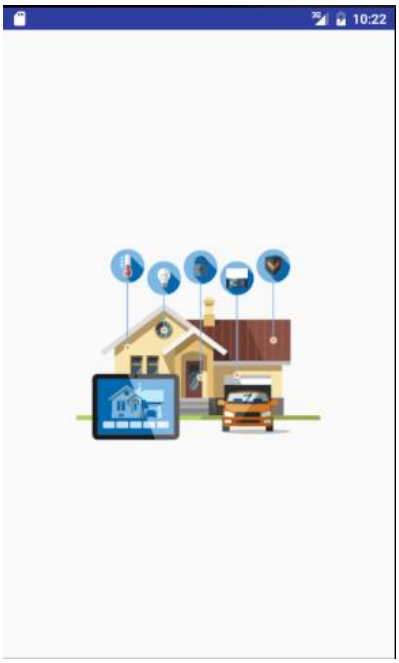
```
.navigation ul{
	margin:0px;
	padding:0px;
}

.navigation ul li{
	margin:0px;
	padding:0px 32px;
	display:inline;
	font-size:15px;
	list-style-type:none;
	border-right:#868B8F dotted 2px;
}

.navigation ul li a{
	color:#03314b;
	text-decoration:none;
}

.navigation ul li a:hover{
	text-decoration:underline;
}

.navigation ul li:last-child {
	border-right:none;
}
#page_content{
	background:#EBEEEE;
	padding:10px;
}

#page_content .right_section{
	width:735px;
	float:right;
}

#page_content .right_section .top_content{
	padding:15px 0 0px 0;
}

#page_content .right_section .top_content .column_one{
	width:300px;
	float:left;
	margin-bottom:20px;
}

#page_content .right_section .top_content .column_two{
	width:300px;
	float:right;
	margin-bottom:20px;
}

.common_content{
	padding:0 0 20px 0;
}

.border_left{
```

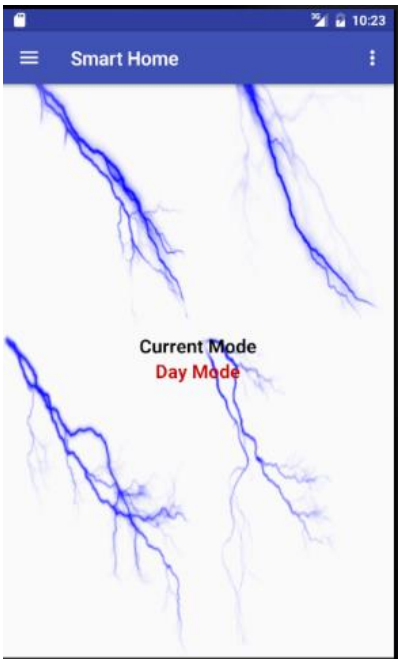
```
	border-left:#6e7678 dotted 1px;
	padding-left:20px;
}

.border_none{
	border:none!important;
	margin:0px!important;
}

.left_side_bar{
	width:210px;
	margin-bottom:10px;
	float:left;
}

.left_side_bar h1{
	font-size:18px;
	padding:0 0 0 35px;
	margin-bottom:15px;
	color:#1b0400;
	background:url(../images/pencil.png) no-repeat 10px center;
}

.left_side_bar .col_1{
	padding:0px 0 20px 0;
}

.left_side_bar .col_1 .box{
	padding:10px 15px 5px;
	border:#A1A9AB solid 2px;
}

.left_side_bar .col_1 .box ul{
	margin:0px;
	padding:0px;
}

.left_side_bar .col_1 .box ul li{
	margin:0px 0px 8px;
	padding:0px 0px 0px 20px;
	font-size:13px;
	list-style-type:none;
	background:url(../images/li.png) no-repeat left center;
}

.left_side_bar .col_1 .box ul li a{
	color:#364b38;
	text-decoration:underline;
}

.left_side_bar .col_1 .box ul li a:hover{
	color:#364b38;
	text-decoration:none;
}

#footer{
	clear:both;
	margin-top:18px;
```

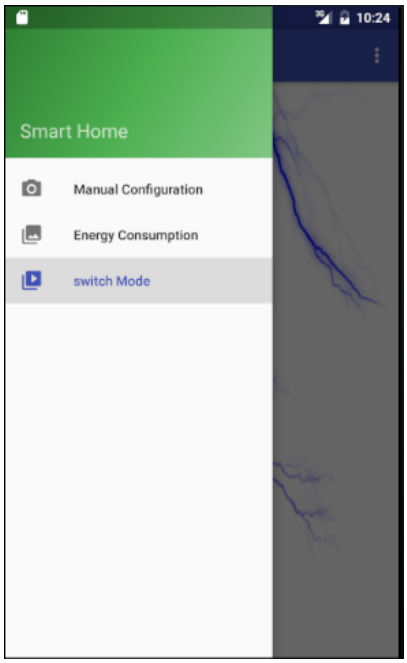
```
        padding:15px 0 12px 0;
        height:25px;
        text-align:center;
        font-family:Verdana, Geneva, sans-serif;
        font-size:12px;
        line-height:19px;
        border-top:#b9bdbe solid 1px;
}

#footer a{
        color: #1B5219;
        text-decoration:none;
}

#footer a:hover{
        color: #1B5219;
        text-decoration:underline;
}
```

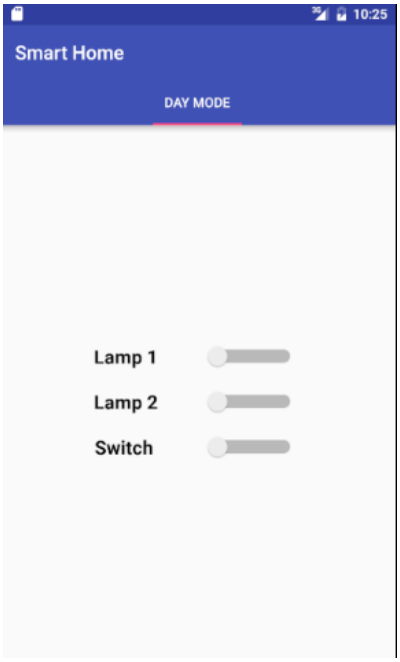
Android Application:

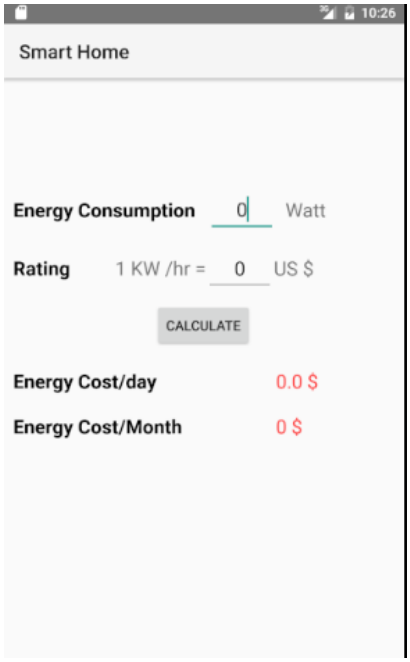


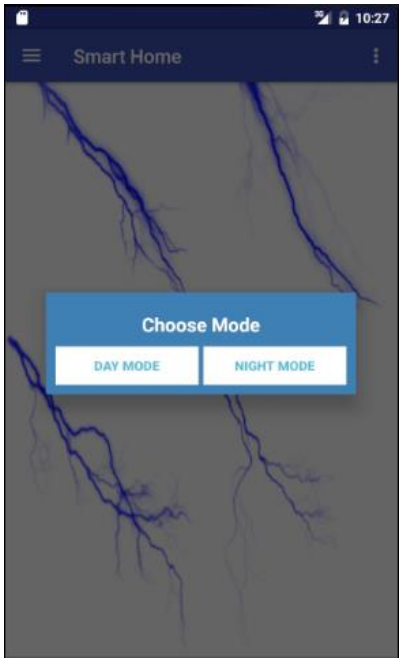




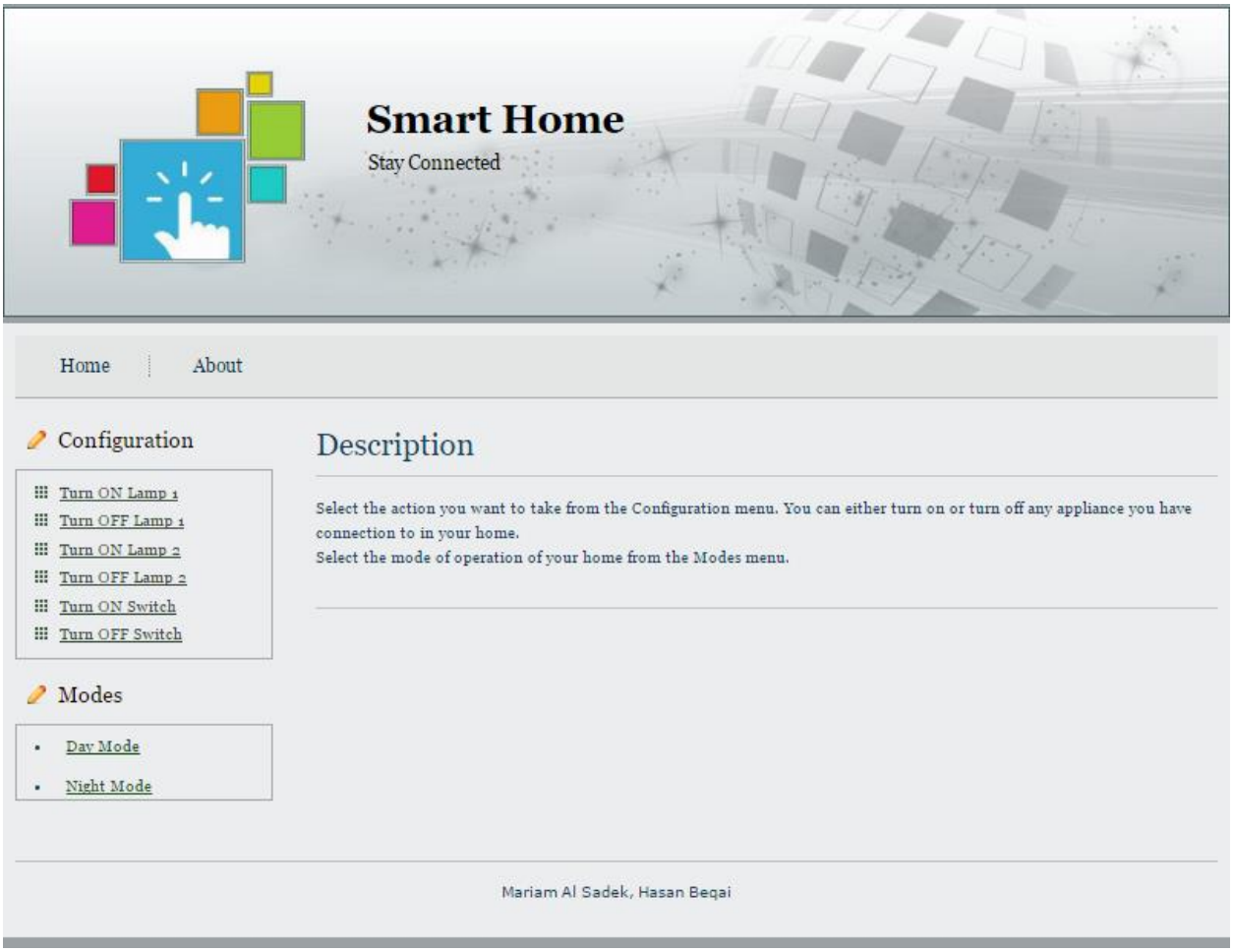




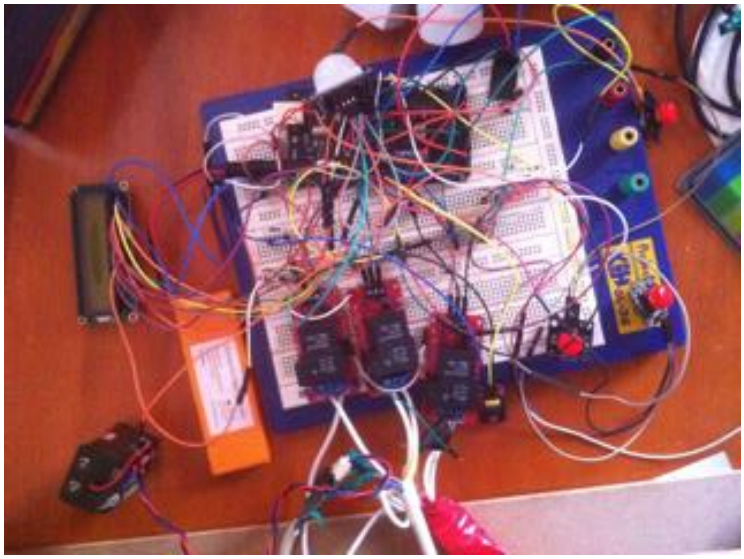




Web Application



Circuit Connections



References:

(2016). Retrieved from http://www.aptinex.com/services/industrial-automation

(2016). Retrieved from https://www.juniperresearch.com

(2016). Retrieved from https://www.electronichouse.com/smart-home/home-automation-protocols-what-technology-is-right-for-you

(2016). Retrieved 7 December 2016, from http://www.learn.sparkfun.com

Arduino - Home. (2016). *Arduino.cc*. Retrieved from https://www.arduino.cc

CNC LAB. (2016). *Cnclablb.com*. Retrieved from http://www.cnclablb.com

DFRobot - Quality Arduino Robot IOT DIY Electronic Kit. (2016). *Dfrobot.com*. Retrieved from https://www.dfrobot.com/

DIY Hacking - Sparking the Indian Maker Movement!. (2016). *DIY Hacking*. Retrieved from https://diyhacking.com

IEEEXplore Digital Library. (2016). *Ieeexplore.ieee.org*. Retrieved from http://ieeexplore.ieee.org/

Instructables - DIY How To Make Instructions. (2016). *Instructables.com*. Retrieved from http://www.instructables.com/

Manohar, S. (2016). *Index of /docs/papers/July2015*. *Ijcsmc.com*. Retrieved from http://www.ijcsmc.com/docs/papers/July2015

Mitchell, B. (2016). *Before IoT was a thing, X10 was there... but does anyone still use it?*. *Lifewire*. Retrieved from https://www.lifewire.com/x10-home-automation-817751

Protocols - buildyoursmarthome.co. (2016). *Buildyoursmarthome.co*. Retrieved from http://buildyoursmarthome.co/home-automation/protocols

Serial Data Transmission and KNX Protocol. (2016). Retrieved from http://www.knx.org/fileadmin/template/documents/downloads_support_menu/KNX_tutor_seminar_page/tutor_documentation/05_Serial%20Data%20Transmission_E0808f.pdf

The definition of automation. (2016). *Dictionary.com*. Retrieved from http://www.dictionary.com/browse/automation

Vangie Beal, V. (2016). *What is Internet Protocol (IP)? Webopedia Definition*. *Webopedia.com*. Retrieved from http://www.webopedia.com/TERM/I/IP.html