

DESIGN AND IMPLEMENTATION OF ENERGY MONITORING USING Wi-Fi NETWORK

A Major Project work submitted in partial fulfillment of the requirement for the
award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS & COMMUNICATION ENGINEERING

By

**P SUMANTH
15211A04E9**

Under the esteemed guidance of

**Dr. T. VASUDEVA REDDY
Associate Professor**



**B.V.Raju Institute of Technology
UGC- AUTONOMOUS
Department of Electronics and Communication Engineering
Vishnupur, Narsapur, Medak.(Dt)
(Affiliated to JNTU, Hyderabad)
2018-2019**

B.V.Raju Institute of Technology
Vishnupur, Narsapur, Medak.(Dt) Pin:502313
(Affiliated to JNTU, Hyderabad)
Ph: 08458-222000, 222001 Fax: 08458-222002

Department of Electronics & Communication Engineering



CERTIFICATE

This is to certify that the Major Project work entitled Design and Implementation of Energy Monitoring Using Wi-Fi Network is being submitted by Mr. P SUMANTH in partial fulfillment of the requirement for the award of the degree of **B.Tech. in Electronics & Communication Engineering**, by Jawaharlal Nehru Technological University Hyderabad is a record of bonafide work carried out by him under my guidance and supervision from 10th December 2018 to 12th April 2019.

The results presented in this project have been verified and are found to be satisfactory.

Dr.T.Vasudeva Reddy
Associate Professor,M.tech,Ph.D
INTERNAL GUIDE

Dr.I.A.Pasha
M.E, PhD, Post Doc. MISTE, MIEEE.
Professor & HOD,Dept. of ECE

EXTERNAL EXAMINER



B V Raju Institute of Technology

(UGC-Autonomous, NBA and NAAC Accredited)

Vishnupur, Narsapur, Dist: Medak.



TEXAS INSTRUMENTS LABORATORY

Electronics and Communications Engineering

CERTIFICATE

This is to certify that Mr. /Ms. _____ bearing
Regd.No. _____, has successfully completed his/her training on TI modules and
implemented a project titled “_____” in Texas
Instruments laboratory, BVRIT under TI India University Program from _____ to
_____.

Coordinator

HOD

ACKNOWLEDGEMENT

I take opportunity to express my indebt gratitude to the persons who contributed for my work, for being my inspiration and guide which led to the successful completion of the project.

I am very grateful towards my College Management and my beloved Principal **Dr. Y. Krishna Reddy, M.tech, Ph.D** for provisioning us the necessary infrastructure and facilities that ensured smooth and satisfactory execution of the project.

I would like to express my profound gratitude to our Head of Department **Dr. I. A. Pasha, M.E, Ph.D, Post Doc., MISTE**. Dept. Of ECE, for his encouragement inspiration and close monitoring and guidance he gave the execution of the project.

I am highly indebted to my guide **Dr. T. Vasudeva Reddy, M.Tech, Ph.D**, Associate Professor, Department of ECE, BVRIT-N, for his excellent guidance and constant encouragement throughout, for the successful completion of the project.

I sincerely express my thanks to the Project Review Committee Members, for their valuable suggestions in each and every review conducted during the course of our project. In this regard I express my thankful regards to Mr. J. Yeshwanth Reddy, Assistant Professor, Department of ECE, BVRIT-N for his valuable suggestions.

By

P Sumanth

15211A04E9

ABSTRACT

The world is nowadays running with smart solutions everywhere in the form of embedded systems. Whatever may be the industry and problem in the environment embedded systems provide optimized solution. The development in the technology both in hardware and software made problems to solve easily. The power management and monitoring is one of the biggest issues dealing nowadays. Various algorithms have been implemented from embedded domain with the help of microcontrollers, sensors and software development tools. Different hardware components have been manufactured for different solutions. Texas Instruments is one of the leading semiconductor companies which provide large varieties of microcontrollers and processors. TI MSP430 G2553 is the one of the ultra low power microcontroller from Texas Instruments. The MSP430 and nodeMCU Wi-Fi along with 1-channel relay and 60Watt bulb are the hardware components used in this project. 1-Channel Relay is used to operate the bulb in this project. Energia Integrated Development Environment (IDE) is one of the software development tools for programming MSP430. It has friendly programming environment for developing firmware to microcontrollers.

The algorithm for monitoring is converted into Embedded C program and debugged into the microcontroller through Energia IDE and NodeMCU through Arduino IDE. This project sends the data to Google Firebase to know when did the electrical appliances turned ON/OFF and calculate the amount of power consumed in a mobile application at every instant and display it in excel sheet. NodeMCU is the Wi-Fi module used to transmit the data from MSP430 through UART communication protocol. The data is sent to the firebase as long as the power is supplied to the program and NodeMCU is connected to the Local Area Network. This project uses less power, and hardware used in this project are easily available at low cost. This project is very reliable and can be used at any environment. This project entirely depends on the algorithm used in it. It was developed to make available for small scale industries and even at home, schools, offices. This project not only saves the power used by the electrical appliances but also it saves power used by hardware components because the entire project is built on low power. MSP430 uses 3.3V and even the components are supplied from MSP430 with the help of level shifters, therefore no need of any external battery. The detailed explanation of this project is described in the report.

CONTENTS

CERTIFICATE	I
ACKNOWLEDGEMENTS	IV
ABSTRACT	V
CONTENTS	VI
LIST OF FIGURES & TABLES	VII
1. INTRODUCTION	1
1.1 Motivation	1
1.2 Objective	1
1.3 Scope	2
2. LITERATURE SURVEY	3
3. ANALYSIS & DESIGN	7
3.1 Circuit Design	7
3.2 Design Flow	25
3.3 Circuit Analysis	26
3.4 Algorithm	27
4. IMPLEMENTATION	32
4.1 Hardware Connections	32
4.2 Software Implementation	35
4.3 Mobile Application Development	38
5. RESULTS	41
5.1 Working Output	41
5.2 Firebase Output	43
5.3 Mobile Application & Excel Output	44
6. CONCLUSION AND FUTURE WORK	46
6.1 Conclusion	46
6.2 Future Work	46
7. REFERENCES	47
8. APPENDIX	48
8.1 MSP430 Demo Program	48
8.2 NodeMCU Program	49
8.3 Main Program	40
8.4 Ultrasonic Sensor Reference distance measurement code	55

LIST OF FIGURES & TABLES

List of Figures:	Page no
1.1 Picture of office	2
1.2 Picture of classroom	2
2.1 Block Diagram of Smart Home Energy Monitoring & Management System	3
2.2 Block Diagram of Data Acquisition	5
2.3 Block Diagram of Data display	5
3.1 Transmitter Block Diagram	7
3.2 Receiver Block Diagram	7
3.3 MSP430 G2553	8
3.4 Block Diagram of MSP430	9
3.5 Memory Map	10
3.6 1-Channel Relay Module	11
3.7 Level Shifter	12
3.8 NodeMCU Wi-Fi Module	12
3.9 Energia IDE	13
3.10 Arduino IDE	21
3.11 APP Inventor	22
3.12 Firebase DB	24
3.13 Design Flow	25
3.14 Block Diagram of Wireless Power Saving Using TI MSP	26
4.1 Demo Hardware Connections	32
4.2 MSP430 Hardware Connections	34
4.3 Circuit Hardware Connections	34
4.4 Energia Software Implementation	36
4.5 Firebase Rules	36
4.6 Designer Window	38
4.7 Blocks Window	39
4.8 Final Project Blocks	40
5.1 Image Inside the Room Without Human	41
5.2 Image Inside the Room With Human Entrance	41
5.3 Image Inside the Room With Human at One Corner	42
5.4 Image Inside the Room With Human Reading Book	42
5.5 Firebase Output	43
5.6 Firebase Storage Output	43
5.7 Mobile Application Output	44
5.8 Excel Sheet Output	45

List of Tables:

4.1	GPIO Connections	33
-----	------------------	----

1. INTRODUCTION:

1.1 Motivation:

Power monitoring is the most commonly faced problem nowadays. Due to the increasing inventions in the field of electrical and electronics the power monitoring of those devices has been difficult. Even though the power consumption of new inventions is less, the production in the utilization of number of components results in more power consumption and harder to monitor. Different technologies and concepts are implemented to monitor the power utilized by each and every electrical component, one of such solution is discussed here for one kind of power monitoring.

Generally humans are more habituated by not taking a look at the power meters and power consumed by the day to day electrical components. Due to the busy schedule we are not at all concerned about utilization of power. This has been a major advantage for power meter readers to increase the electricity bill as per their wish, and also the wrong meter readings. To control theses kind of actions many proposals and implementation had been approaching from the past few years. Different Solutions from embedded domain came into the actions and a few were implemented. Only a few solutions were implemented because a lot of parameters have to be considered such as accuracy, performance, reliability, economic etc. This problem inspired me to provide an optimized solution such that each and everyone should able to implement it.

1.2 Objective:

The main objective of this project is to monitor the power consumed by each electrical appliances in any commercial spaces like offices, schools, colleges and houses more accurately with full reliability without any external hardware and low cost and then display it an excel sheet with timestamp, so that the end user can easily notice. In terms of accuracy the present solution offered by direct power meter readings which may sometimes provide wrong meter readings output but the problem is with the high cost for implementation. Low cost is also the major function to be noted because our aim also to reduce the power used by this project because it should also be implemented by small scale industries. The below figures 1.1 and 1.2 represents the office and classroom.



Fig: 1.1 Picture of Office



Fig: 1.2 Picture of Classroom

1.3 Scope:

This solution for power monitoring of electrical appliances that operates on human detection which was implemented for power saving works more accurately with low cost. The circuit and technical concepts involved in this project is simple to understand and also the hardware parts are often available and easy to implement in any environment. The solution offered by this project has a lot additional benefits along with the primary task. The calculation of power is done in mobile application which uses less space and user friendly. The final power utilized is displayed in excel sheet with timestamp.

The hardware consists of Wi-Fi module so that the data will be sent continuously without any delay. The data sent to the Firebase can be used for multiple purposes, it can be used for data analytics, storing the data at cloud and also for cross checking with the electricity bill received at every month. The data can also be viewed in the mobile app by just connecting the link of the server to the Mobile app which is created by any freely available sources such as android studio, MIT app inventor etc. This project can be extended more for future use and can develop for machine learning and data analytics.

2 LITERATURE SURVEY

2.1 Smart Home Energy Monitoring and Management System

Smart home energy monitoring and management system is the project developed by Abhishek and published to Instructables website. The idea of this project is to develop a system such that it will be capable to keep a track of each and every appliance in the home and the user will be able to acquire all appliance energy consumption parameters. Along with this, the energy consumption parameters of each individual appliance will be sent to gateway where an intelligent algorithm will be running to manage all the appliances as per user requirements. The user can monitor the energy parameters of each individual load using an android smart phone which will also work as a data setter to set various user programmable parameters like high/low cut-off voltage, etc.

The objective of this project is the use of smart meter system. Smart Meters are electronic measurement devices used by utilities to communicate information for billing customers and operating their electric systems. For over fifteen years electronic meters, have been used effectively by utilities in delivering accurate billing data for at least a portion of their customer base. A Smart Meter System is required which can analyze multiple appliances in a household getting readings such as voltage, current, active power, apparent power, reactive power, power factor and frequency. With the help of a wired / wireless connection, the device can connect to a central gateway and the gathered information can be uploaded and processed by the gateway management system. The data can then be displayed on the platform's graphical android-based user interface. The platform allows users to access the data from any android enabled device. To reduce cost the system requires energy metering nodes that can communicate with the gateway wirelessly or in wired way in such a way that only one Wi-Fi access point is needed for a household containing many monitored appliances. Furthermore it is required that the current information regarding the appliances can also be viewed on a local display with a menu interface. The remote energy metering node will be considered to be successful if the following criteria are met.



Fig: 2.1 Block diagram of Smart Home Energy Monitoring & Management system

The figure 2.1 is the block diagram of Smart Home Energy Monitoring and Management System. The system consists of three main parts. The Gateway, the energy nodes, and an android application interface. Figure 1 below shows how these parts all interact with each other.

Energy Metering Node: The energy node unit has the task of taking power measurements when requested and sending them then to the gateway. It waits for a request from the gateway via UART (a wired connection). When a request is received it reads the measurements from the EMIC and then sends this information back to the gateway.

Gateway Unit: The gateway is responsible for collecting data and then sending it to the android application interface. It also has a console access that can show all the relevant information along with a small interface. The requests information from the energy node via wired communication. The energy node then sends the information back to the gateway. The gateway then forwards this information on to the android application interface. This process happens in regular intervals which can be set in the menu interface. Because the gateway and the energy nodes are separate a single system can comprise of many energy nodes. The advantage of this is that the cost to monitor another additional appliance is low because only the energy node needs to be purchased.

Android application interface: An android application is designed which is responsible for reading the gateway for each energy node connected and is also responsible for setting various gateway configurations and parameters like various thresholds, etc.

2.2 Home Electric Energy Monitoring System Design and Prototyping

The project Home Electric Energy Monitoring System Design and Prototyping is developed by J.G. Josue, J.M. Pina, and M.V. Neves and published in ResearchGate Website in 2011. The developed system consists of two electronic devices, they are data acquisition device and data display device. The data acquisition system measures power and energy consumed by loads and the data display device displays measured data onto a small LCD screen and sends result to computer.

The main features of this monitoring system are: wireless communication between acquisition and display devices, monitoring capability at the appliance and switchboard circuit's level, average hourly energy use and electricity cost information display, and data recording on the computer. Wireless communication between devices ensures greater flexibility and system's ease of use. The system's ability to monitor both appliance level and switchboard circuit's level informs the consumer about the balance of each appliance or circuit load. The knowledge of average hourly energy use and electricity cost provides an important information that motivates changes to consumer's behavior. A computer connection is available to record measured data and to use it in many computer applications, such as, daily charting energy use data.

The data acquisition device diagram block is represented in figure 2.2. This device consists of five major blocks: power integrated circuit (IC), microcontroller, wireless transceiver, signal conditioning and power supply. The data acquisition device measures line voltage and current signals through appropriate sensors. These analog signals are then conditioned and used by a power IC, which measure RMS voltage, RMS current, power factor and active power. This information is transmitted to a microcontroller that computes the energy consumed by a load and communicates with a wireless transceiver. The transceiver is responsible for sending the measured data to the data display device and for receiving commands from the user. The microcontroller can also communicate with status LEDs and with a reset button. The device power supply provides 5V DC from 230V AC line.

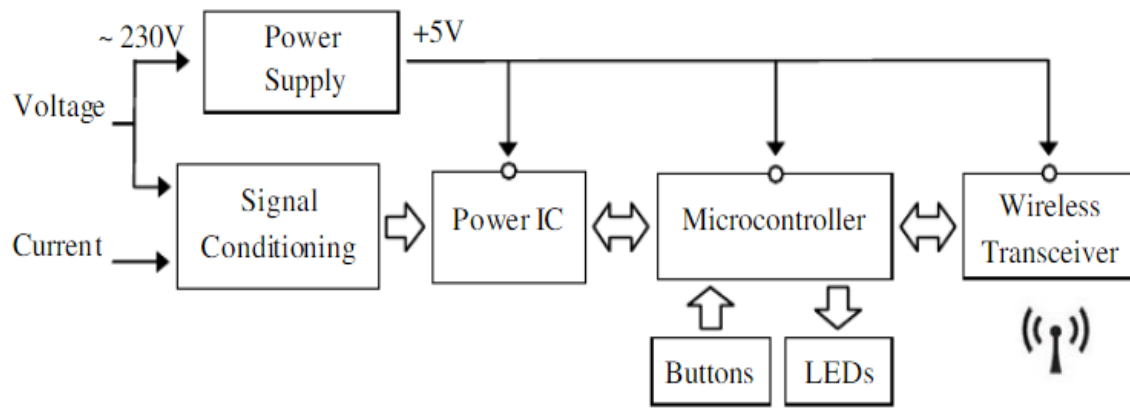


Fig: 2.2 Block Diagram of Data Acquisition

Figure 2.3 presents data display device diagram block. This device consists of four major blocks: microcontroller, wireless transceiver, LCD and power supply.

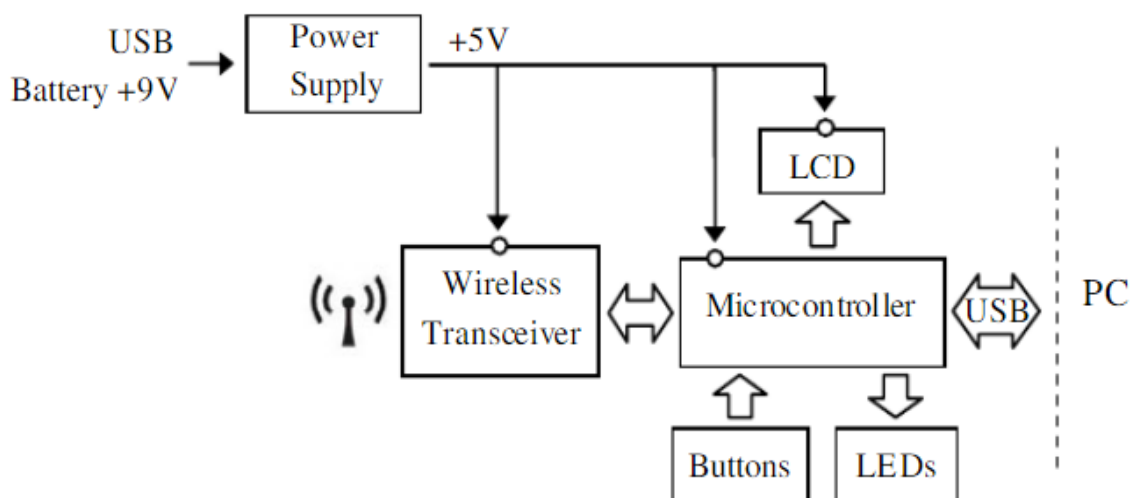


Fig: 2.3 Block Diagram of Data Display

Data display device receives measured data from the data acquisition device through a wireless transceiver. The transceiver sends the received information to a microcontroller. This microcontroller is responsible for several operations, including LCD data driving, communication with a computer via USB, LEDs and buttons control and electricity cost calculation. The device buttons allow the user to select and perform many functions, such as data measurement initialization, power IC calibration and electricity tariff definition. This device is supplied by a 9V DC battery or 5V DC USB interface.

3. ANALYSIS AND DESIGN

3.1 Circuit Design

The previous mentioned projects have some drawbacks. These drawbacks have to be solved and should be available to middle class people and also for small scale industries. The challenge is to design a circuit and implement it with minimal cost, less hardware and easy available. Different algorithms have been researched to limit the hardware components and relay for operating bulbs and nodeMCU Wi-Fi module for transmitting data to server. The micro controller used for this project is Texas Instruments MSP430 G2553. The detailed explanation of components used and working principles are explained in detail. The below figure is the Block diagram of the project. The below figures 3.1 and 3.2 represents the block diagram of transmitter and receiver.

BLOCK DIAGRAM:

Transmitter:

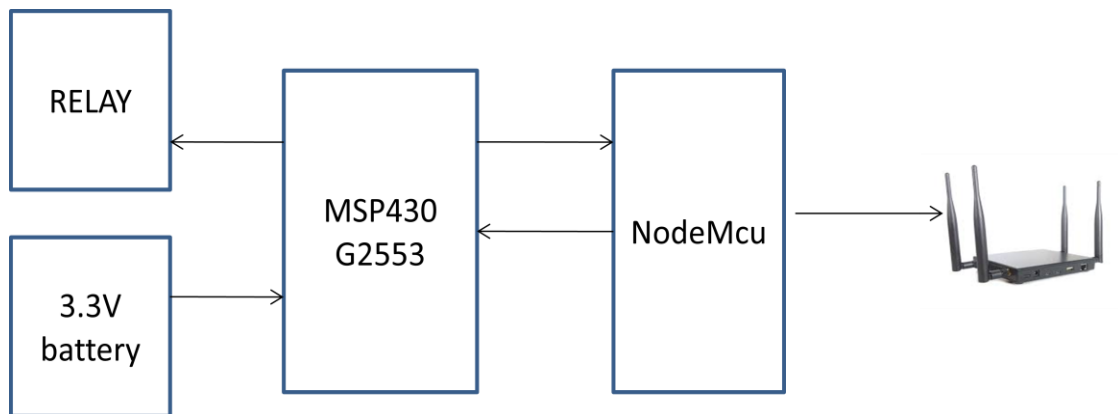


Fig: 3.1 Transmitter Block Diagram

Receiver:

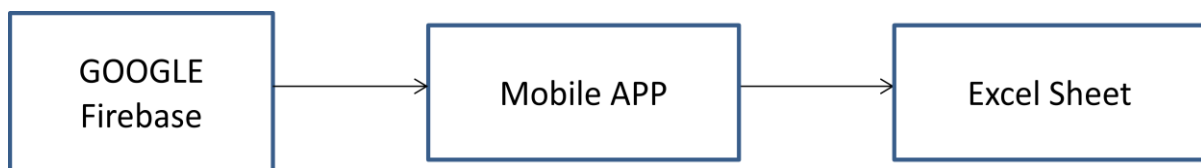


Fig: 3.2 Receiver Block Diagram

3.1.1 Hardware components:

MSP430 G2553:

It is an ultra low power micro controller from Texas instruments. It is a 16 bit micro controller which runs at a clock speed of 16MHz. The current drawn in idle mode can be less than 1 μ A. It can be throttled back for lower power consumption. The MSP430 also uses six different low-power modes, which can disable unneeded clocks and CPU. Additionally, the MSP430 is capable of wake-up times below 1 microsecond, allowing the microcontroller to stay in sleep mode longer, minimizing its average current consumption. The device comes in a variety of configurations featuring the usual peripherals: internal oscillator, timer including PWM, watchdog, USART, SPI, I2C, 10/12/14/16/24-bit ADCs and brownout reset circuitry. Some less usual peripheral options include comparators (that can be used with the timers to do simple ADC), on-chip op-amps for signal conditioning, 12-bit Digital to Analog Converter, LCD driver, hardware multiplier, universal serial bus, and dynamic memory allocation for ADC results. Apart from some older EPROM(MSP430E3xx) and high volume mask ROM (MSP430Cxxx) versions, all of the devices are in system programmable via JTAG (full four-wire or Spy Bi Wire) or a built in bootstrap loader (BSL) using UART such as RS232, or USB on devices with USB support.

There are, however, limitations that preclude its use in more complex embedded systems. The MSP430 does not have an external memory bus, so it is limited to on-chip memory (up to 4 KB flash memory and 128 KB RAM) which may be too small for applications that require large buffers or data tables. Also, although it has a DMA controller, it is very difficult to use it to move data off the chip due to a lack of a DMA output strobe. The below fig 3.3 is the MSP430 G2553 launch pad. The voltage required to run the micro controller is 3.3V.

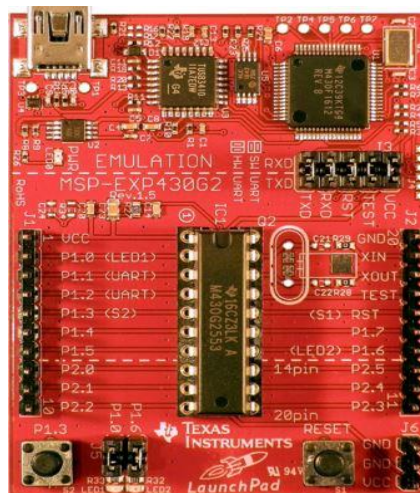


Fig: 3.3 MSP430 G2553

Key features of the MSP430x2xx family include:

- Ultralow-power architecture extends battery life
 - 0.1 μ A RAM retention
 - 0.8 μ A real-time clock mode
 - 250 μ A/MIPS active
- High-performance analog ideal for precision measurement
 - Comparator-gated timers for measuring resistive elements
- 16-bit RISC CPU enables new applications at a fraction of the code size.
 - Large register file eliminates working file bottleneck
 - Compact core design reduces power consumption and cost
 - Optimized for modern high-level programming.
 - Only 27 core instructions and seven addressing modes
 - Extensive vectored-interrupt capability
- In-system programmable Flash permits flexible code changes, field upgrades and data logging.

MSP430G2553

Functional Block Diagram, MSP430G2x53

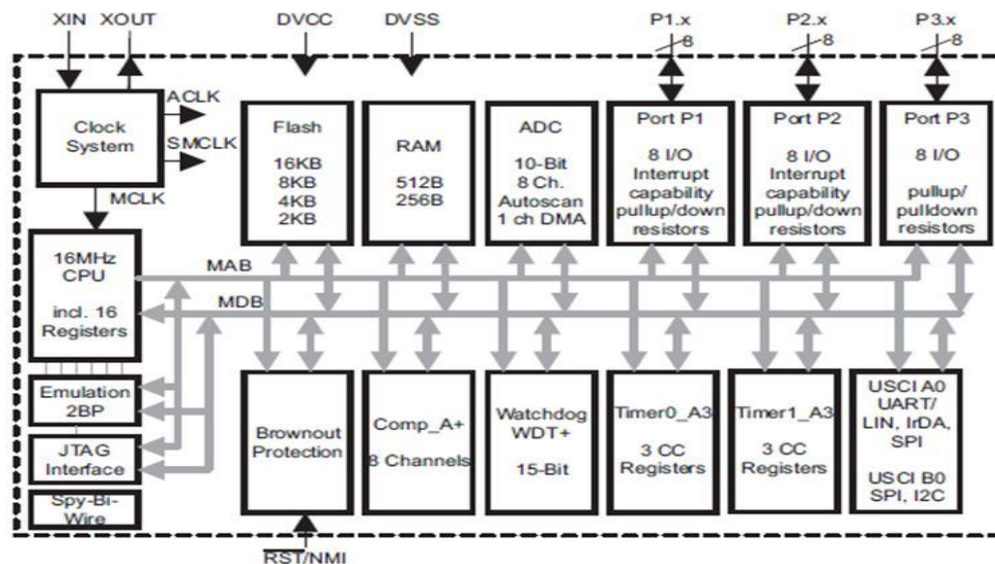


Fig: 3.4 Block Diagram of MSP430

The block diagram of MSP430 is shown in figure 3.4. The architecture followed by MSP430 is Von Neumann which means it has single bus system for both code and data. It has both 8 bit and 16 bit registers. There are about 16 registers among which 12 are general purpose registers and 4 are special purpose registers. They are

1. Program Counter
2. Stack Pointer
3. Status Register
4. Constant generator

These 4 Special functions registers are 16 bit registers. They are seven addressing modes They are

1. Register Mode
2. Indexed Mode
3. Symbolic Mode
4. Absolute Mode
5. Indirect Register Mode
6. Indirect Auto Increment Mode
7. Immediate Mode

The sample assembly language instruction is given as

`MOV #45h,TONI`

Description: Move the immediate constant 45h, which is contained in the word following the instruction, to destination address TONI. When fetching the source, the program counter points to the word following the instruction and moves the contents to the destination.

The below figure 3.5 shows the memory map of MSP430 G2553.

Address	Type of memory
0xFFFF	interrupt and reset
0xFFC0	vector table
0xFFBF	flash code memory
0xF800	(lower boundary varies)
0xF7FF	
0x1100	
0x10FF	flash
0x1000	information memory
0x0FFF	<i>bootstrap loader</i>
0x0C00	(not in F20xx)
0x0BFF	
0x0280	
0x027F	RAM
0x0200	(upper boundary varies)
0x01FF	peripheral registers
0x0100	with word access
0x00FF	peripheral registers
0x0100	with byte access
0x000F	special function registers
0x0000	(byte access)

Fig: 3.5 Memory Map

1 - CHANNEL RELAY MODULE:

A relay is an electrically operated device. It has a control system and (also called input circuit or input contactor) and controlled system (also called output circuit or output contactor). It is frequently used in automatic control circuit. To put it simply, it is an automatic switch to controlling a high-current circuit with a low-current signal.

The advantages of a relay lie in its lower inertia of the moving, stability, long-term reliability and small volume. It is widely adopted in devices of power protection, automation technology, sport, remote control, reconnaissance and communication, as well as in devices of electro mechanics and power electronics. Generally speaking, a relay contains an induction part which can reflect input variable like current, voltage, power, resistance, frequency, temperature, pressure, speed and light etc. It also contains an actuator module (output) which can energize or de-energize the connection of controlled circuit. There is an intermediary part between input part and output part that is used to coupling and isolate input current, as well as actuate the output. When the rated value of input (voltage, current and temperature etc.) is above the critical value, the controlled output circuit of relay will be energized or de-energized. The figure 3.6 represents 1 channel relay module.



Fig: 3.6 1- Channel Relay Module

LEVEL SHIFTER:

Since MSP430 provides only 3.3V. The voltage is not sufficient for driving component relay modules. In order to drive those components we need 5V, so to provide 5V we need a new battery of 5V or else a level shifter. A level shifter will shift the 3.3V to 5V with the simple circuit of capacitor. Figure 3.7 showing the level shifter having low voltage and high voltage pins at either ends. It is available with very low cost and the following figure shows the level shifter.

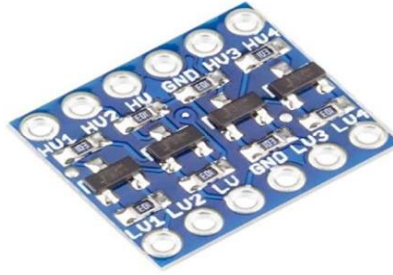


Fig: 3.7 Level Shifter

NODEMCU WI-FI MODULE:

NodeMCU is an open source IoT platform. It has ESP8266 Wi-Fi System on Chip from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. Serial communication like UART is sufficient for transmitting the data from micro controller.

The NodeMCU is programmed with arduino IDE therefore we can directly transmit data through serial communication. RX and TX are sufficient for data transmission and receiving. Since this project does not need any data to be received we can neglect the concept of receiving data. NodeMCU is very easily available Wi-Fi module and can be obtained at low price. Fig 3.8 shows the picture of nodeMCU Wi-Fi module.

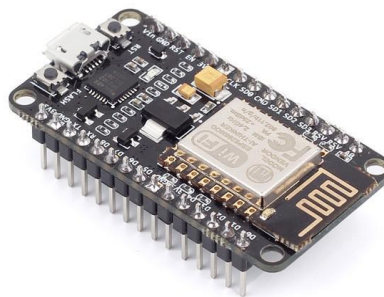


Fig: 3.8 NodeMCU

3.1.2 Software Required:

ENERGIA IDE:

Energia is an open source & community-driven integrated development environment (IDE) & software framework. Based on the Wiring framework, Energia provides an intuitive coding environment as well as a robust framework of easy-to-use functional APIs & libraries for programming a microcontroller. Energia supports many TI processors, primarily those available in the LaunchPad development ecosystem.

Energia is simple and easy-to-use code editor and compiler with built-in Serial Monitor/terminal. It features a robust framework of intuitive functional APIs for controlling microcontroller peripherals (i.e. digitalRead, digitalWrite, Serial.print, etc). It also Support for various TI embedded devices (MSP430, TM4C, CC3200, C2000, etc). Open source & hosted in GitHub. Higher level libraries are also available (i.e. Wi-Fi, Ethernet, displays, sensors & more). Seamlessly migrate your Energia project into Code Composer Studio v6, allowing developers to take advantage of the LaunchPad kit's on-board debugger. Figure 3.9 shows the Energia sketch with void setup() and voidloop() function.



Fig: 3.9 Energia IDE

Here we have given what is a sketch and explained a few basic functionalities and terms of Energia such as how to declare variables, print statements, comments, delay and how to execute loops.

SKETCH:

A *sketch* is the name that Energia uses for a program. It's the unit of code that is uploaded to and run on a LaunchPad board.

COMMENTS:

The first few lines of the Blink sketch are a *comment*:

```
/*
 * The basic Energia Example.
 * Turns on an LED on for one second, then off for one second, repeatedly.
 * Pin 6 has an LED connected on MSP430 boards, has a name 'RED_LED' in the code.
 */
```

Everything between the `/*` and `*/` is ignored by the LaunchPad when it runs the sketch (the `*` at the start of each line is only there to make the comment look pretty, and isn't required). It's there for people reading the code: to explain what the program does, how it works, or why it's written the way it is. It's a good practice to comment your sketches, and to keep the comments up-to-date when you modify the code. This helps other people to learn from or modify your code.

There's another style for short, single-line comments. These start with `//` and continue to the end of the line. For example, in the line:

```
int ledPin = 14;           // LED connected to digital pin 14
```

The message "LED connected to digital pin 14" is a comment.

VARIABLES:

A *variable* is a place for storing a piece of data. It has a name, a type, and a value. For example, the line from the Blink sketch above declares a variable with the name `ledPin`, the type `int`, and an initial value of 14. It's being used to indicate which LaunchPad pin the LED is connected to. Every time the name `ledPin` appears in the code, its value will be retrieved. In this case, the person writing the program could have chosen not to bother creating the `ledPin` variable and instead have simply written 14 everywhere they needed to specify a pin number. The

advantage of using a variable is that it's easier to move the LED to a different pin: you only need to edit the one line that assigns the initial value to the variable.

Often, however, the value of a variable will change while the sketch runs. For example, you could store the value read from an input into a variable. There's more information in the Variables tutorial.

FUNCTIONS:

A *function* (otherwise known as a *procedure* or *sub-routine*) is a named piece of code that can be used from elsewhere in a sketch. For example, here's the definition of the `setup()` function from the Blink example:

```
void setup()
{
    pinMode(ledPin, OUTPUT); // sets the digital pin as output
}
```

The first line provides information about the function, like its name, "setup". The text before and after the name specify its return type and parameters: these will be explained later. The code between the { and } is called the *body* of the function: what the function does. You can *call* a function that's already been defined (either in your sketch or as part of the Energia language). For example, the line `pinMode(ledPin, OUTPUT);` calls the `pinMode()` function, passing it two *parameters*: `ledPin` and `OUTPUT`. These parameters are used by the `pinMode()` function to decide which pin and mode to set. Some basic functions are

1. `pinMode()`
2. `digitalWrite()`
3. `delay()`

The `pinMode()` function configures a pin as either an input or an output. To use it, you pass it the number of the pin to configure and the constant `INPUT` or `OUTPUT`. When configured as an input, a pin can detect the state of a sensor like a pushbutton; this is discussed in a later tutorial. As an output, it can drive an actuator like an LED.

The `digitalWrite()` functions outputs a value on a pin. For example, the line:

```
digitalWrite(ledPin, HIGH);
```

Set the `ledPin` (pin 14) to `HIGH`, or ~3 volts. Writing a `LOW` to pin connects it to ground, or 0 volts.

The `delay()` causes the LaunchPad to wait for the specified number of milliseconds before continuing on to the next line. There are 1000 milliseconds in a second, so the line:

```
delay(1000);
```

It creates a delay of one second. `setup()` and `loop()`. These are two special functions that are a part of every Energia sketch: `setup()` and `loop()`. The `setup()` is called once, when the sketch starts. It's a good place to do setup tasks like setting pin modes or initializing libraries. The `loop()` function is called over and over and is the heart of all sketches. You need to include both functions in your sketch, even if you don't need them for anything.

CORE FUNCTIONS:

Simple programs that demonstrate basic Energia commands use Core functions. These are included with the Energia environment; to open them, click the Open button on the toolbar and look in the examples folder. For some examples, additional hardware is required. These can be acquired individually or in popular electronics starter kits.

Basics:

- BareMinimum: The bare minimum of code needed to start an Energia sketch.
- Blink: Turn an LED on and off.
- DigitalReadSerial: Read a switch, print the state out to the Energia Serial Monitor.
- AnalogReadSerial: Read a potentiometer, print it's state out to the Energia Serial Monitor.
- Fade: Demonstrates the use of analog output to fade an LED.
- ReadAnalogVoltage: Reads an analog input and prints the voltage to the serial monitor.

Digital:

- Blink Without Delay: Blinking an LED without using the `delay()` function.
- Button: Use a pushbutton to control an LED.
- Debounce: Read a pushbutton, filtering noise.
- Button State Change: Counting the number of button pushes.
- Input Pullup Serial: Demonstrates the use of `INPUT_PULLUP` with `pinMode()`.
- Tone: Play a melody with a Piezo speaker.
- Pitch follower: Play a pitch on a piezo speaker depending on an analog input.
- Simple Keyboard: A three-key musical keyboard using force sensors and a piezo speaker.
- Tone4: Play tones on multiple speakers sequentially using the `tone()` command.

Analog:

- AnalogInOutSerial: Read an analog input pin, map the result, and then use that data to dim or brighten an LED.
- Analog Input: Use a potentiometer to control the blinking of an LED.
- AnalogWrite: Fade 7 LEDs on and off, one by one, using an MSP430G2 LaunchPad board.
- Calibration: Define a maximum and minimum for expected analog sensor values.
- Fading: Use an analog output (PWM pin) to fade an LED.
- Smoothing: Smooth multiple readings of an analog input.

Communication:

- ReadASCIIString: Parse a comma-separated string of ints to fade an LED.
- ASCII Table: Demonstrates Energia's advanced serial output functions.
- Dimmer: Move the mouse to change the brightness of an LED.
- Graph: Send data to the computer and graph it in Processing.
- Physical Pixel: Turn a LED on and off by sending data to your LaunchPad from Processing.
- Virtual Color Mixer: Send multiple variables from LaunchPad to your computer and read them in Processing.
- Serial Call Response: Send multiple variables using a call-and-response (handshaking) method.
- Serial Call Response ASCII: Send multiple variables using a call-and-response (handshaking) method, and ASCII-encode the values before sending.
- SerialEvent: Demonstrates the use of SerialEvent().
- Serial input (Switch (case) Statement): How to take different actions based on characters received by the serial port.

Control Structures:

- If Statement (Conditional): How to use an if statement to change output conditions based on changing input conditions.
- For Loop: Controlling multiple LEDs with a for loop.
- Array: A variation on the For Loop example that demonstrates how to use an array.
- While Loop: How to use a while loop to calibrate a sensor while a button is being read.
- Switch Case: How to choose between a discrete number of values. Equivalent to multiple If statements. This example shows how to divide a sensor's range into a set of four bands and to take four different actions depending on which band the result is in.
- Switch Case 2: A second switch-case example, showing how to take different actions based in characters received in the serial port.

Strings:

- StringAdditionOperator: Add strings together in a variety of ways.
- StringAppendOperator: Append data to strings.
- StringCaseChanges: Change the case of a string.
- StringCharacters: Get/set the value of a specific character in a string.
- StringComparisonOperators: Compare strings alphabetically.
- StringConstructors: How to initialize string objects.
- StringIndexOf: Look for the first/last instance of a character in a string.
- StringLength & StringLengthTrim: Get and trim the length of a string.
- StringReplace: Replace individual characters in a string.
- StringStartsWithEndsWith: Check which characters/substrings a given string starts or ends with.
- StringSubstring: Look for "phrases" within a given string.

Sensors, Motors, & Displays:

- Temperature: use on board MCU core temp sensor.
- Tilt Sensor: use a basic tilt sensor.
- Servo: move a servo to control mechanical objects.
- Basic Motor: turn a basic motor.
- Segment Display: display basic number and letter values.
- 2x16 Character Display: output strings to a character display.

MultiTasking:

- ButtonEvent: Read a button in one task and have another task wait for the button to be pressed.
- EventLibrary: Send an event in one task and have another task wait for the event.
- Monitor: Displays CPU utilization, task memory usage, etc. Requires VT100 terminal.
- MultiAnalogInput: Reads analog inputs in different tasks at different rates.
- MultiBlink: Blink 3 LEDs at different rates.
- MultiTaskSerial: Shows 2 threads sending a string to the Serial monitor at different rates.

Connectivity:

- WiFi: WiFi library examples.
- MQTT: Use the MQTT lightweight protocol to enable IoT & M2M applications.
- StandardFirmata: Use firmata protocol to dynamically communicate with the microcontroller.
- Temboo: Access hundreds of web APIs through Temboo using Energia.
- AT&T M2X: Post Energia data to AT&T M2X cloud service.

- BLE Mini: Use Red Bear Lab BLE Mini to control you LaunchPad.
- Freeboard.io: Create a cloud dashboard with your Energia data using freeboard.io.
- Contiki: Access Contiki OS for IoT using Energia.

BoosterPacks:

- EducationalBP MKII: examples involving buzzer, LCD, LEDs, accelerometer, push buttons, and more
- CC3100: introduction to SimpleLink WiFi CC3100 BoosterPack

Other Tutorials:

- Sidekick for TI LaunchPad: Use the Seeedstudio Sidekick Basic Kit for TI LaunchPad with Energia.
- SIK for LaunchPad: Use the Sparkfun Inventor's Kit with Energia.
- Grove Starter Kit for LaunchPad: Use Grove modules to access sensors and components for prototyping.
- O-Scope Operation: Learning how to use an Tektronix Oscilloscope with MSP430 LaunchPad.
- LabVIEW Home: Use Energia in National Instruments LabVIEW.
- Processing: Create GUIs and visual representations of Energia data using Processing IDE.
- Energia.nu/learn: Full workshops on Energia material.
- IoP Machine: Learn how to create an internet connected popcorn machine.
- Energy Trace: Learn how to measure energy consumption in your Energia system.

Here is a simple program that describes some of the core functions and energia sketch.

```
int buttonPin = 3;

// setup initializes serial and the button pin
void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

// loop checks the button pin each time,
// and will send serial if it is pressed

void loop()
{
  if (digitalRead(buttonPin) == HIGH)
```

```

    Serial.write('H');
else
    Serial.write('L');

    delay(1000);
}

```

Here button is connected to the pin 3. Whenever the button is pressed it prints 'H' and when it is not pressed it prints 'L' in the serial monitor and waits for a second as we mentioned delay and then continues.

ARDUINO IDE:

The Arduino IDE is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiringproject, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

NodeMCU is programmed through Arduino IDE by using the libraries of Google firebase, required as per the project. We can get a lot of libraries in Arduino IDE from Github since it is a open source. Anyone can easily start programming using Arduino IDE since lot of videos and tutorials are available in internet. The below figure 3.10 represents the Arduino sketch having void setup() and void loop() function.

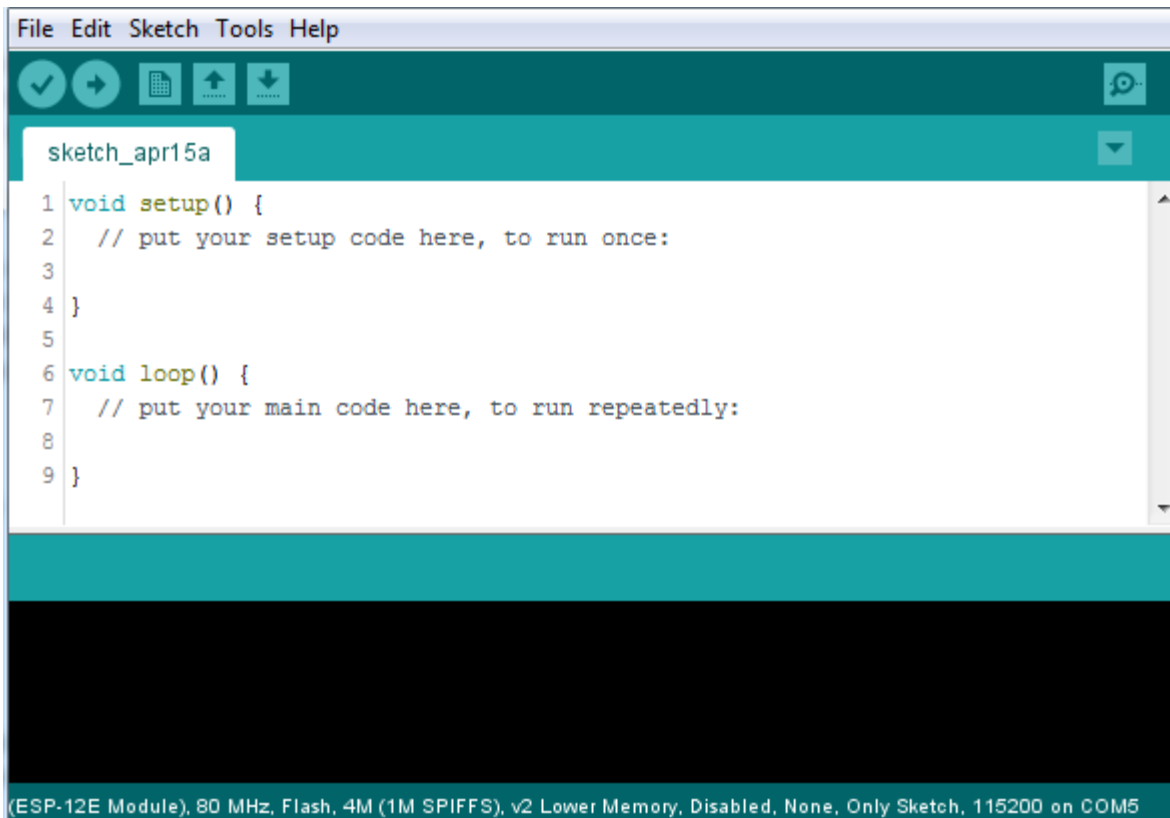


Fig: 3.10 Arduino IDE

MIT APP INVENTOR 2:

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).

It allows newcomers to computer programming to create software applications for the Android operating system. It uses a graphical interface, very similar to Scratch and the StarLogo TNG user interface, which allows users to drag and drop visual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments.

App Inventor and the projects on which it is based are informed by constructionist learning theories, which emphasizes that programming can be a vehicle for engaging powerful ideas through active learning. App Inventor also supports the use of cloud data via an experimental FirebaseDB component. We have developed the application for calculating the power by taking the data from FirebaseDB. Figure 3.11 represents the MIT APP inventor developing window in blocks.

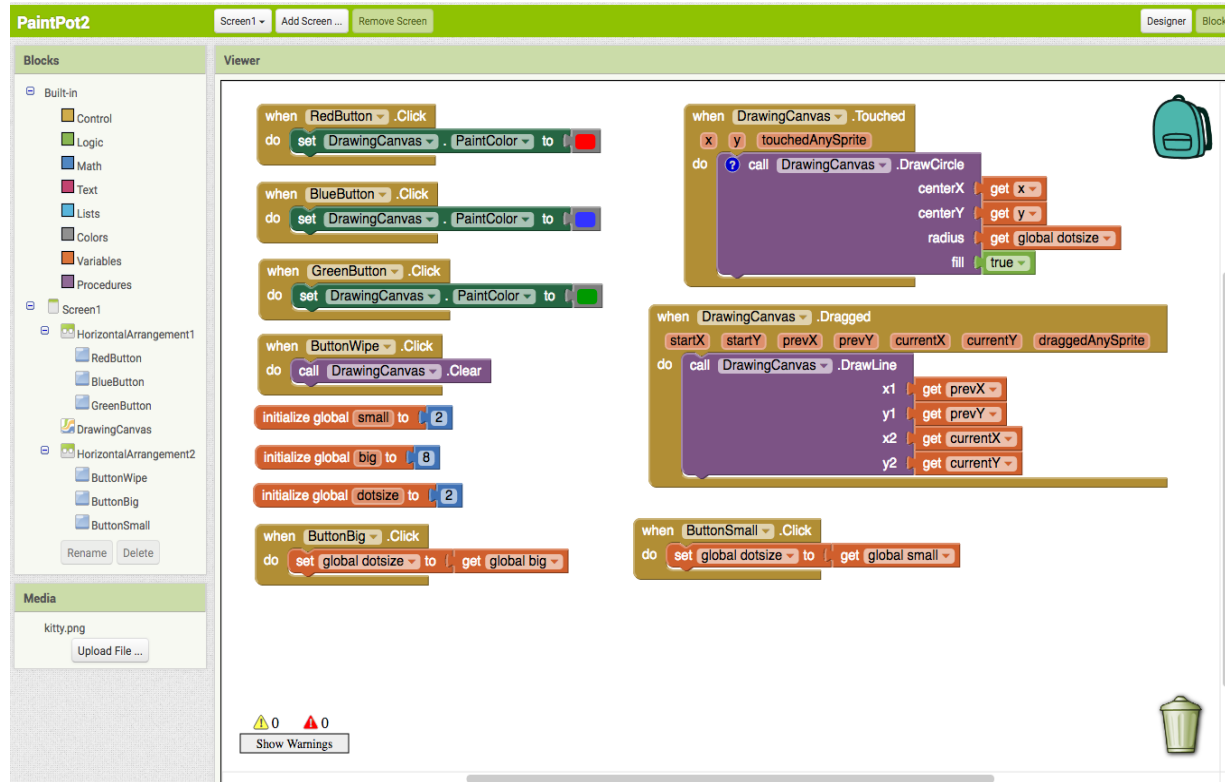


Fig: 3.11 MIT APP Inventor

GOOGLE FIREBASE:

Firebase is a mobile and web application development platform developed by Firebase in 2011, then acquired by Google in 2014. Firebase provides the tools to develop high-quality apps, grow the user base. Firebase provides large number of services they are

Firestore Analytics:

Firebase Analytics is a cost-free app measurement solution that provides insight into app usage and user engagement.

Firebase Cloud Messaging:

Formerly known as Google Cloud Messaging, Firebase Cloud Messaging is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which as of 2016 can be used at no cost.

Firebase Auth:

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google. Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

Realtime database:

Firebase provides a realtime database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with Android, iOS, JavaScript, Java, Objective-C, Swift and Node.js applications. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the realtime database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the Realtime Database was released for beta use.

Firebase Storage:

Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage.

Firebase Hosting:

Firebase Hosting is a static and dynamic web hosting service that launched on May 13, 2014. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layer encryption (SSL). Firebase partners with Fastly, a CDN, to provide the CDN backing Firebase Hosting. The company states that Firebase Hosting grew out of customer requests; developers were using Firebase for its real-time database but needed a place to host their content.

ML Kit:

ML Kit is a mobile machine learning system for developers launched on May 8, 2018 in beta during the Google I/O 2018. ML Kit API's feature a variety of features including text recognition, detecting faces, scanning barcodes, labelling images and recognising landmarks. It is currently available for iOS or Android developers. You may also import your own TensorFlow Lite models, if the given API's aren't enough. The API's can be used on-device or on cloud.

Crashlytics:

Crash Reporting creates detailed reports of the errors in the app. Errors are grouped into clusters of similar stack traces and triaged by the severity of impact on app users. In addition to automatic reports, the developer can log custom events to help capture the steps leading up to a crash. Before acquiring Crashlytics, Firebase was using its own Firebase Crash Reporting.

Performance:

Firebase Performance provides insights into an app's performance and the latencies the app's users experience.

Firebase Test Lab for Android and iOS:

Firebase Test Lab for Android and iOS provides cloud-based infrastructure for testing Android and iOS apps. With one operation, developers can initiate testing of their apps across a wide variety of devices and device configurations. Test results including logs, videos, and screenshots are made available in the project in the Firebase console. Even if a developer hasn't written any test code for their app, Test Lab can exercise the app automatically, looking for crashes. Test Lab for iOS is currently in beta.



Fig: 3.12 Firebase DB

3.2 Design Flow

Here is the design flow of the project presented in below figure 3.13 with explanation.

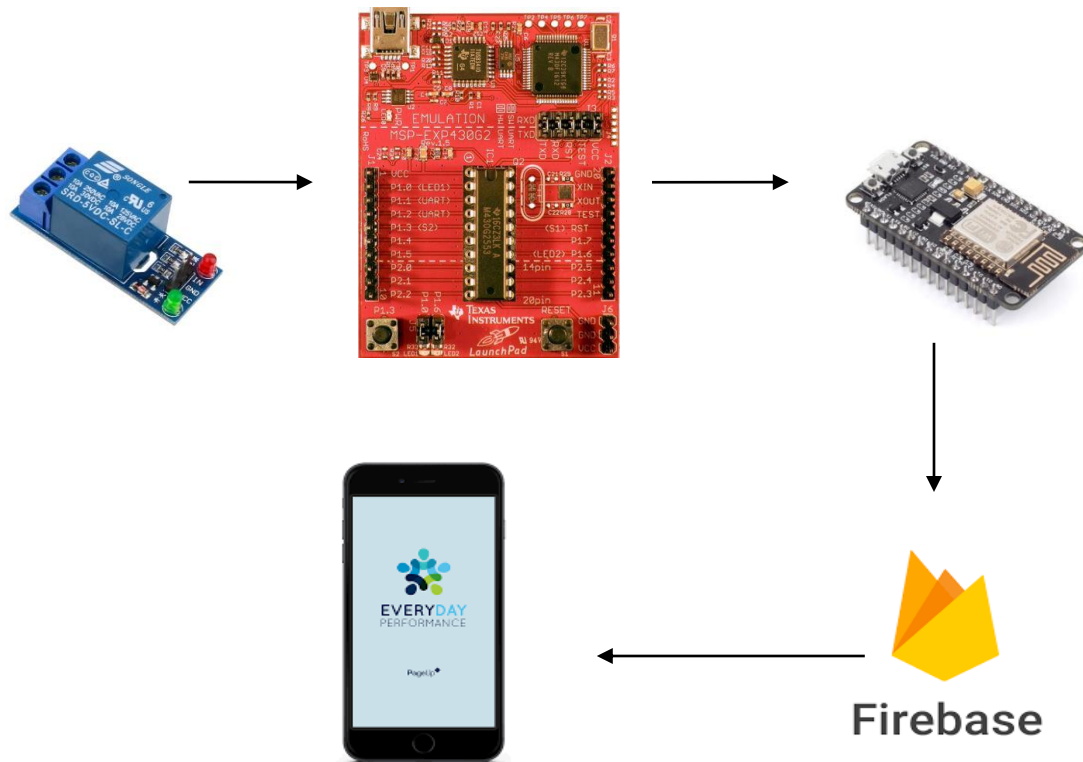


Fig: 3.13 Design Flow

When ever the relay is ON then we send a variable pr1 which is assigned as value 5 from MSP430 to nodeMCU through UART. The data received from UART is send to Firebase DB using Local Area network (Wi-Fi) and whenever the firebase is updated, mobile application retrieve data from firebase and checks the condition whether the variable sent is 4 or 5, If value is 5, the timer is enabled and starts counting in seconds and displays the time in app. Again when relay is OFF we send another variable pr2 with value 4 from MSP430 to nodeMCU. The nodeMCU again sends the data to firebase DB. The mobile application gets data from firebase and checks the condition whether the value is 5 or 4. If the value is 4 then it stops the timer and calculates the power consumed for that period of that respective device. Here we are using 60W bulb so the time period is divided by 3600 since time is measured in seconds and power is measrued in Kwh and multiplied with factor 60. The timer is set to 0 and the power in watts is displayed in the mobile app. The process is followed from refernce [4]. The final stage is the calculated power at that instant period is send to excel sheet. The excel sheet gets appended whenever the power is calculated along with the timestamp. The detailed explanation of calculating power, app development, firebase working is given 4.1 software implementation.

3.3 Circuit Analysis

Since this project is extension to my mini project “Wireless Power Saving Using TI MSP”, here is the brief explanation of that. The Fig 3.15 is block diagram of Wireless Power Saving Using TI MSP.

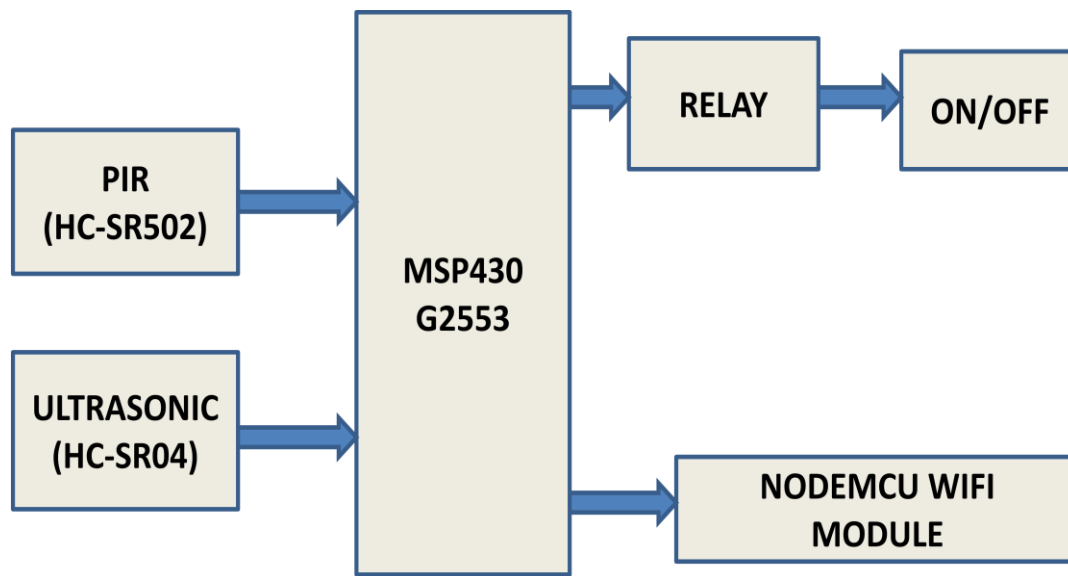


Fig: 3.14 Block diagram of wireless power saving using TI MSP

We took the reference [1] for designig the circuit. We can see that the ultrasonic sensor and the PIR sensors are used as input to the microcontroller the relay is used as output to the microcontroller, NodeMCU is used an output for transmitting the data to a firebase DB this is the minimum connection and from the firebase DB to mobile APP for calculating power. The core of this project is the algorithm to be implemented because the entire set up have already done in previous projects. The algorithm plays key role. Simple concept and different approaches to solve this problem laid down this algorithm. The microcontroller performs logical operations on the input data from ultrasonic sensors and PIR sensors and based on the conditions of logic we perform different actions by sending the output data to relay and NodeMCU. The entire algorithm runs of input from PIR, Ultrasonic sensors and Data sending to NodeMCU.

In this prototype we are using two PIR sensors and one ultrasonic sensor to show the accurate output. We can also use single PIR sensor but using more sensor can give a clear picture. One ultrasonic sensor is mandatory for this project. The number of relays is based on the number of bulbs to be controlled in the room. We can use multiple number of relays. In this prototype I am using two relays and two 5W bulbs. The circuit is very simple and easy to design and implement with hardware.

3.4 Algorithm

The core of the project is the algorithm to be implemented. Since we are using PIR sensors in this project we need to take care more about that because it detects the motion that is static but not the presence, so we need to implement an algorithm that should be effective and stable. Here we are declaring some variables initializing to zero to perform the operations on the bulbs. Whenever PIR sensor detects the motion the variable is set to one. Based on the variable we perform the operations on the relay such that the bulb glows only when a person is inside the room. Since the PIR sensor has a range of 270° it can detect the person outside the room to make sure the person is inside the room we need the confirmation that the person is inside the room. This can be done with the help of ultrasonic sensors, ultrasonic sensor are placed at the entrance of the room.

A certain distance is calculated at the entrance of the room set it as a reference distance. When a Human enters into the room the reference distance gets disturb and the new distance is sent to the microcontroller. If the reference distance is disturb then it indicates that a person is entering into the room and now with the help of PIR sensor and ultrasonic sensor we are able to control the lights in a room. This prototype uses two PIR sensors located at two corners of the room. Here we need to consider all the cases for the accurate working of the project we have done the several assumption and practical cases and finally brought down the algorithm into a c program. Based on the number of PIR sensors placed in the 2 to the power of number of PIR sensors case should be considered. For 2 sensors we need to consider at 4 cases. They are shown below. PIR1 and PIR2 are sensor readings in digital that is it represents 0 or 1. Zero indicates no motion is detected and 1 indicates motion is detected.

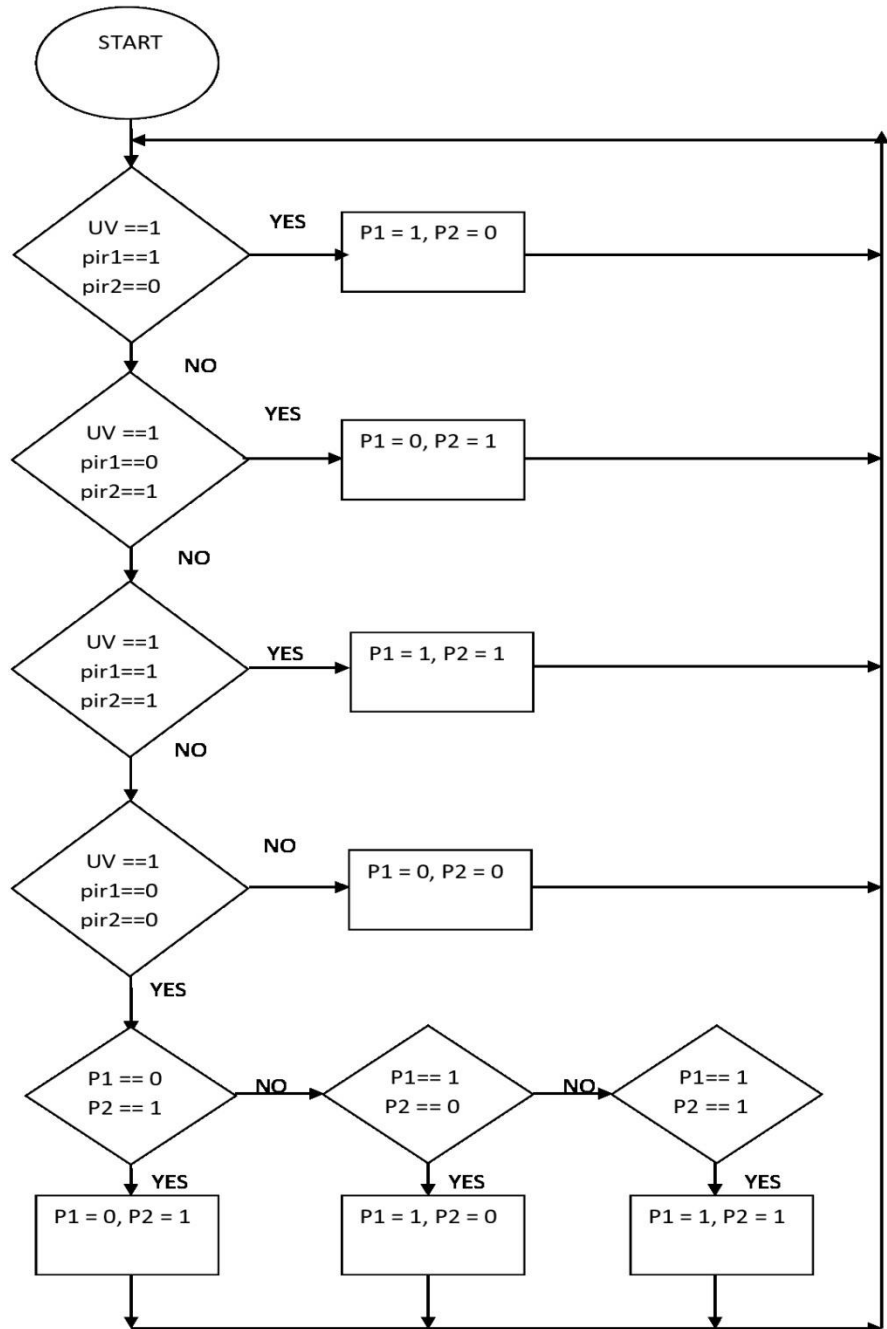
1. PIR1 = 0 and PIR2 = 0
2. PIR1 = 1 and PIR2 = 0
3. PIR1 = 0 and PIR2 = 1
4. PIR1 = 1 and PIR2 = 1

The above mentioned are the primary cases to be considered and only valid if the ultrasonic sensor is disturbed with the reference distance. We also declare another variable for ultrasonic sensor (UV) and we set to UV value to 1 when the reference distance is disturbed. The above mentioned cases are modified a little as shown in below.

1. UV = 1 & PIR1 = 1 & PIR2 = 0
2. UV = 1 & PIR1 = 0 & PIR2 = 1
3. UV = 1 & PIR1 = 1 & PIR2 = 1
4. UV = 1 & PIR1 = 0 & PIR2 = 0
5. UV = 0 & PIR1 = 1 & PIR2 = 0

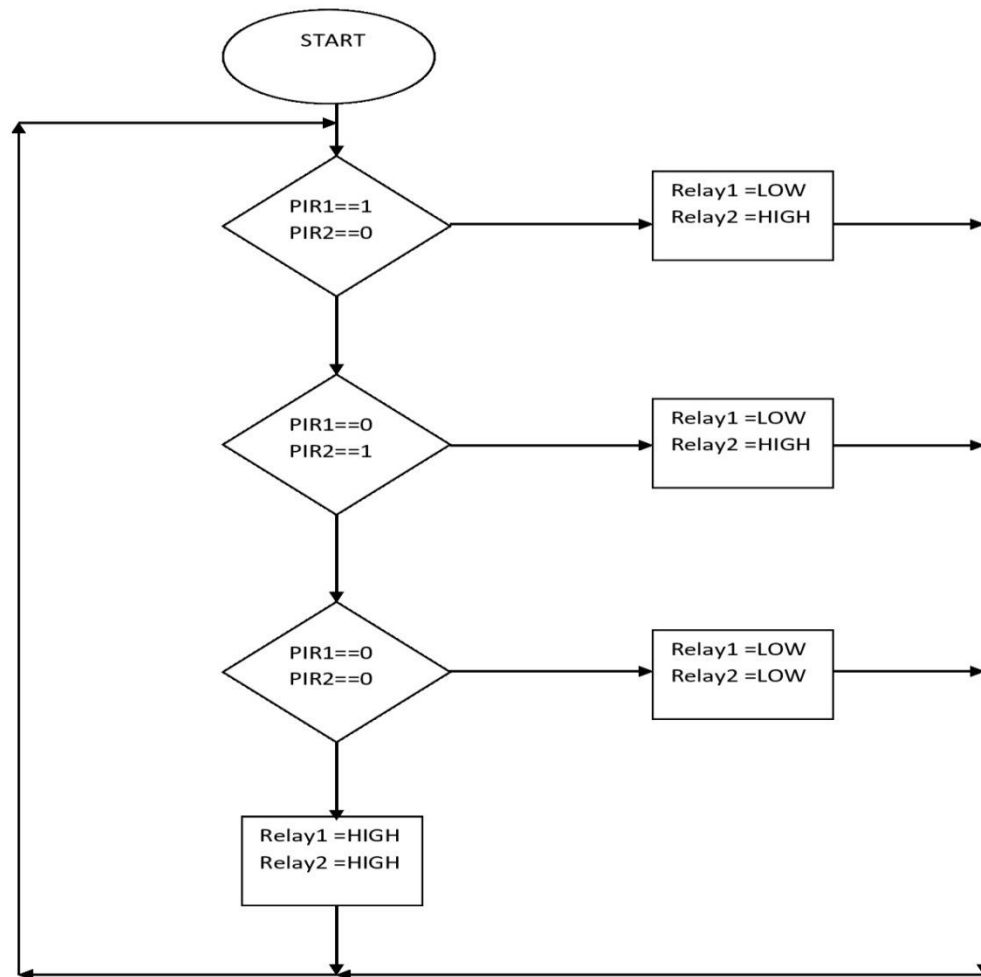
The following flowcharts describe the algorithm and give the clear picture of working principle.

FLOW CHART:



First we initialize the variables to zero that is $u=0$, $pir1=0$, $pir2=0$, $pr1=4$, $pr2=5$. Based on the different cases and conditions we modify the variables to one. $Pr1$ and $Pr2$ are variables to send the data from nodeMCU to mobile APP through firebase DB. We perform the operations on relay based on the different combinations of initialized variables to control the lights and sending data to firebase. The following Flowchart provides the information on working of relays. Since Relays are negative edge triggered we set values to zero.

FLOW CHART:



The Flowcharts gives the complete information and working principle of the project. Then finally the data ON/OFF of the Bulbs is sent to the server with the serial communication. Due to the inbuilt libraries present in the Energia IDE, the programming part is simple to execute and following chapter will give the hardware connections and output of the project.

There are also few parameters have to be considered, the reference distance is to be calculated before the program to be implemented. The reference distance can be calculated directly by setting the trigger pin and echo pin. The distance can be calculated by a standard formula given by

$$\text{distance} = \text{duration} * 0.034 / 2;$$

The duration is the time interval between the Pulse triggered and received by Echo. The flowchart below gives the information of setting the ultrasonic sensor variable to one.

Algorithm for ultrasonic sensor distance:

- Set Trig pin to LOW
- Set Delay of 2 microseconds
- Set Trig pin to HIGH
- Set Delay of 10 microseconds
- Calculate the duration from echo pin
- Calculate the distance from above formula

The above two Flowcharts and the algorithm for ultrasonic sensor distance measurement along with the initializing the variables and configuring the serial communication complete the entire project algorithm to be implemented. We are sending the data i.e variables through serial communication to nodeMCU.

Here we are calculating the power of one electric 60W Bulb1. The data sent to nodeMCU is the following. So whenever the relay for bulb1 is ON, then variable pr1 is sent to nodeMCU using serial communication and when relay for bulb1 is OFF, then variable pr2 is sent to nodeMCU. We program the nodeMCU to connect to Google firebase DB through Wi-Fi network and also send the data of variables. We have created a mobile application using MIT APP Inventor 2 for calculating the power consumed by Bulb1 and displaying and sending to excel sheets.

Algorithm for calculating power in APP and displaying:

- Set the timer ON when on fixed variable (k=4) is received through firebase DB from NodeMCU.
- Set the timer OFF when another fixed variable (k=5) is received through firebase DB from NodeMCU.

- Get the ONOFF duration from the timer in seconds.
- Divide the duration with 3600 and multiple with power rating of Bulb1. In this case it is 60W.
- The calculated power is displayed in the mobile APP and also append to the excel sheets.

4. IMPLEMENTATION

4.1 Hardware Connections:

The below figure 4.1 is the hardware connections of project “Energy Monitoring Using Wi-Fi Network”. It consists of Relay, MSP430, NodeMCU, level shifter and connecting wires.

The nodeMCU and MSP430 G2553 micro controller are connected through connecting wires based on serial communication (UART). Universal Asynchronous Receiver transmitter is serial communication protocol used to connect two peripherals. It uses only two hardware pins Tx and Rx. NodeMCU is connected to Wi-Fi network for sending data to mobile application. The Tx pin of MSP430 is connected to Rx pin of nodeMCU and the Rx pin of MSP430 is connected to Tx pin of nodeMCU, The Vcc and Gnd pins are connected to level shifter and the relay’s Vcc and Gnd pins are connected to level shifter other end.

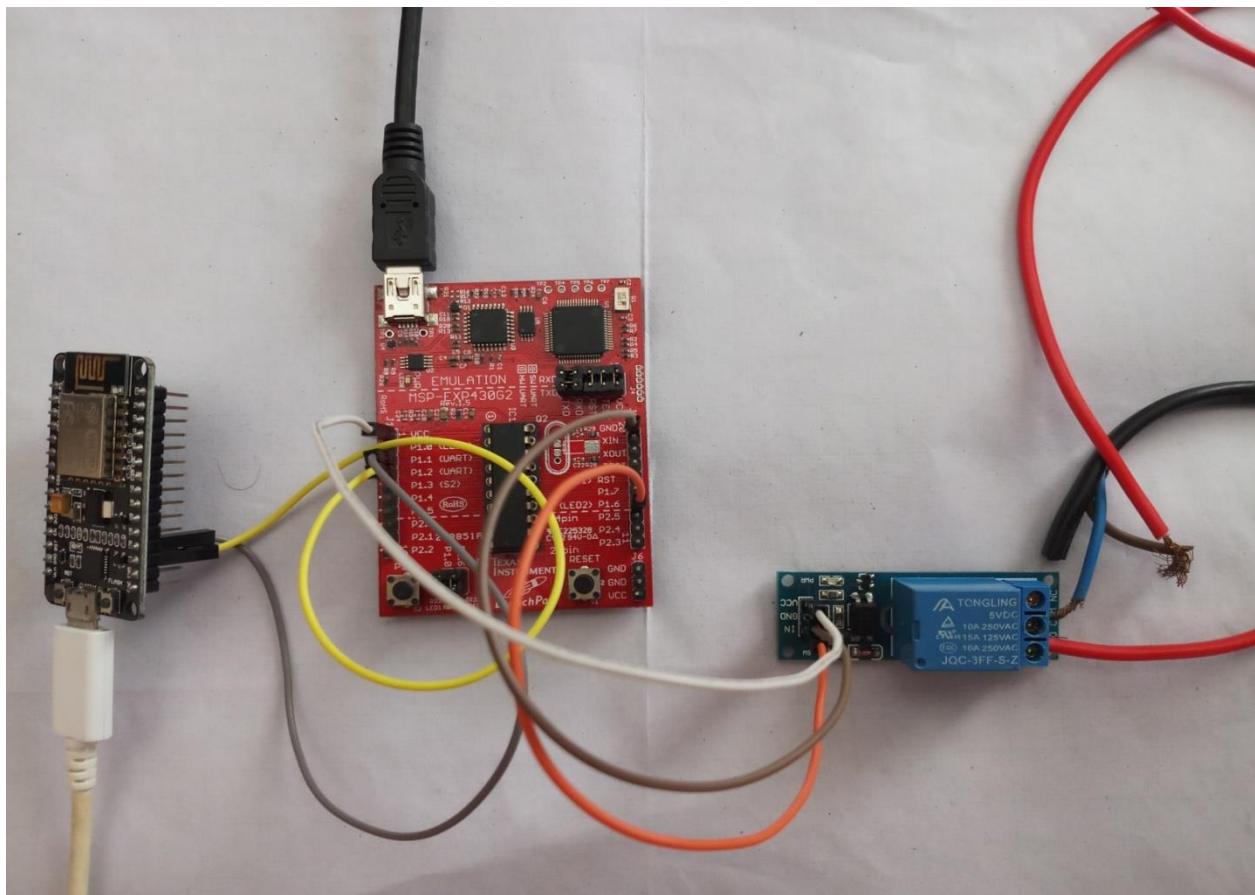


Fig: 4.1 Demo Hardware connection

Now here are final Hardware connections of both the projects “Design and Implementation of Energy Monitoring Using Wi-Fi Network” and “Wireless Power Saving Using TI MSP”. The components we are using are digital components. Therefore the connections among the sensors,

microcontroller, relays, NodeMCU are connected with General Purpose Input Output (GPIO) pins of micro controller. The power supply to the MSP430 LaunchPad is provided by a battery of 5V externally with the USB cable. We have 20 GPIO pins for MSP430 LaunchPad. The selection of GPIO pins should be taken care since pin 2 and 3 of MSP430 LaunchPad are used for serial communication we cannot use these pins for general purpose. Therefore NodeMCU is connected with that serial pins. TX pin of MSP430 is connected to RX pin of NodeMCU and TX pin of NodeMCU is connected to RX of MSP430 with the help of connecting wires. Pins 8, 9 are used for ultrasonic sensors. Trigger pin is connected to pin 8 and echo pin is connected to pin 9. The ultrasonic sensor uses 5V. MSP430 does not provide 5V it provides 3.3V. Therefore Level shifter is connected to VCC and Ground pins of MSP430. The 5V power supply and ground can be obtained from Level shifter to other components.

Pins 5 and 6 are used for PIR sensors. The Output pin of PIR1 sensor is connected to pin 5 of MSP430 and output pin of PIR2 sensor is connected to pin 6 of MSP430. The Relays are connected to pins 14 and 15. The following table gives the information GPIO connections.

TABLE

GPIO	COMPONENT
1	VCC of level shifter
2	RX pin of NodeMCU
3	TX pin of NodeMCU
4	Not Connected
5	Output pin of PIR1 sensor
6	Output pin of PIR2 sensor
7	Not Connected
8	Trigger pin of ultrasonic sensor
9	Echo pin of ultrasonic sensor
10	Not Connected
11	Not Connected
12	Not Connected
13	Not Connected
14	Input pin of Relay 1
15	Input pin of Relay 2
16	Not Connected
17	Not Connected
18	Not Connected
19	Not Connected
20	Ground pin of Level Shifter

Table 4.1 GPIO Connections

The below Figs show the connections of hardware. Fig 4.2 is the MSP430 connections components connected to GPIO pins. Fig 4.3 shows the entire circuit connections.

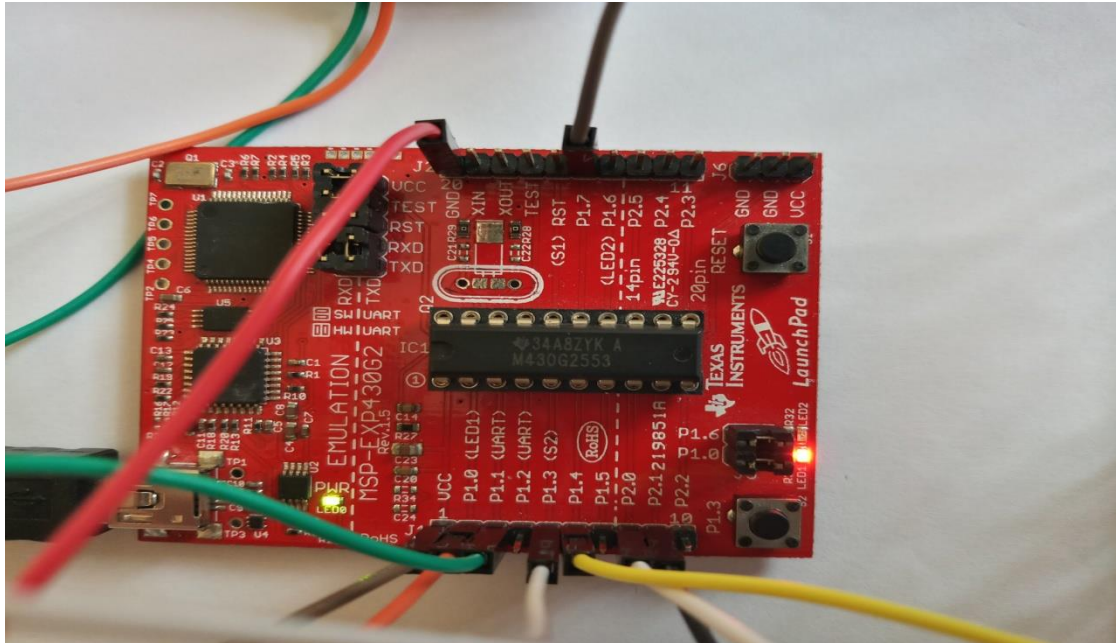


Fig 4.2 MSP430 Hardware Connections

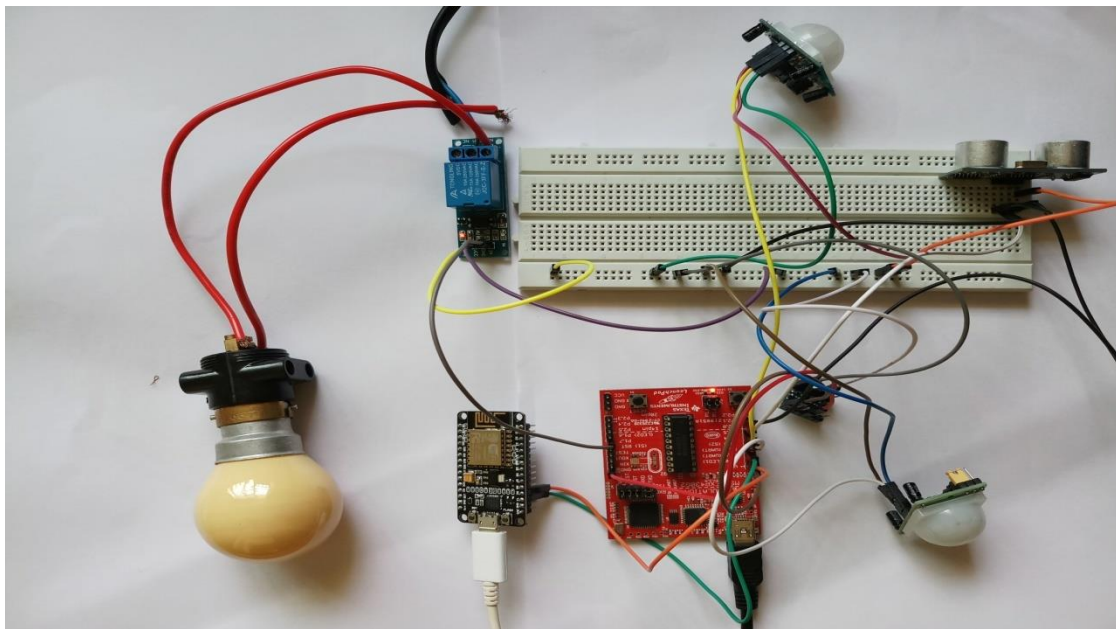


Fig 4.3 Circuit Hardware Connections

The above figure 4.3 is final hardware connections of project.

4.2 Software Implementation:

First the variables have to be declared and initialized to some value. The hardware components which are connected to GPIO pins are initialized. It depends upon the algorithm as shown.

```
int relay1 = 14           // Connected relay to 14th pin
int pr1 = 4;              // This variable used to start the timer
int pr2 = 5;              // This variable used to stop the timer
```

The variables pr1 and pr2 are send as data through serial communication (UART), pr1 indicates the timer in mobile app to start and pr2 indicates the timer to stop. Digital pin 14 is connected to relay1.

The serial communication is initialized in void setup() function as shown below.

```
void setup()
{
    Serial.begin(9600);      // Starts the serial communication
}
```

9600 denoted in the above line is the baud Rate for UART communication which means 9600 bits are transmitted per second.

The algorithm discussed in the chapter Circuit Analysis and Design is implemented as a c program. Two different programs have to be implemented for the project. They are

1. A program to ON/OFF the relay and send the data from MSP430 to nodeMCU through UART communication. This program is debugged into MSP430 using Energia IDE.
2. A program to receive the data from MSP430 and transmit to Firebase DB. The firebase is configured here and send data continuously to firebase DB.

The first part is the initializing the variables and GPIO pin as inputs and outputs. The second part is the send the data continuously as a periodic event. So it is programmed in void loop(). The code edited in the void loop will run continuously until high priority interrupt is occurred.

Due to the inbuilt libraries the programming is become easy with Energia IDE. The flow of signal is completely Digital. There is no Analog part in the programming. Due to the high clock frequency the system runs the program with very fast without any delay. The code to be run continuously is kept inside the Void loop () function and code which has to be executed once is kept in the void setup () function as shown in the below fig. The entire program is in the appendix. The void loop function contains the entire main algorithm such that it repeats continuously whenever the power supply is connected to the microcontroller. The execution of each statement depends on clock frequency and the entire function of statements and delay inserted. Figure 4.4 displays the void setup() and void loop() functions. These two functions are basic function in programming through IDE.

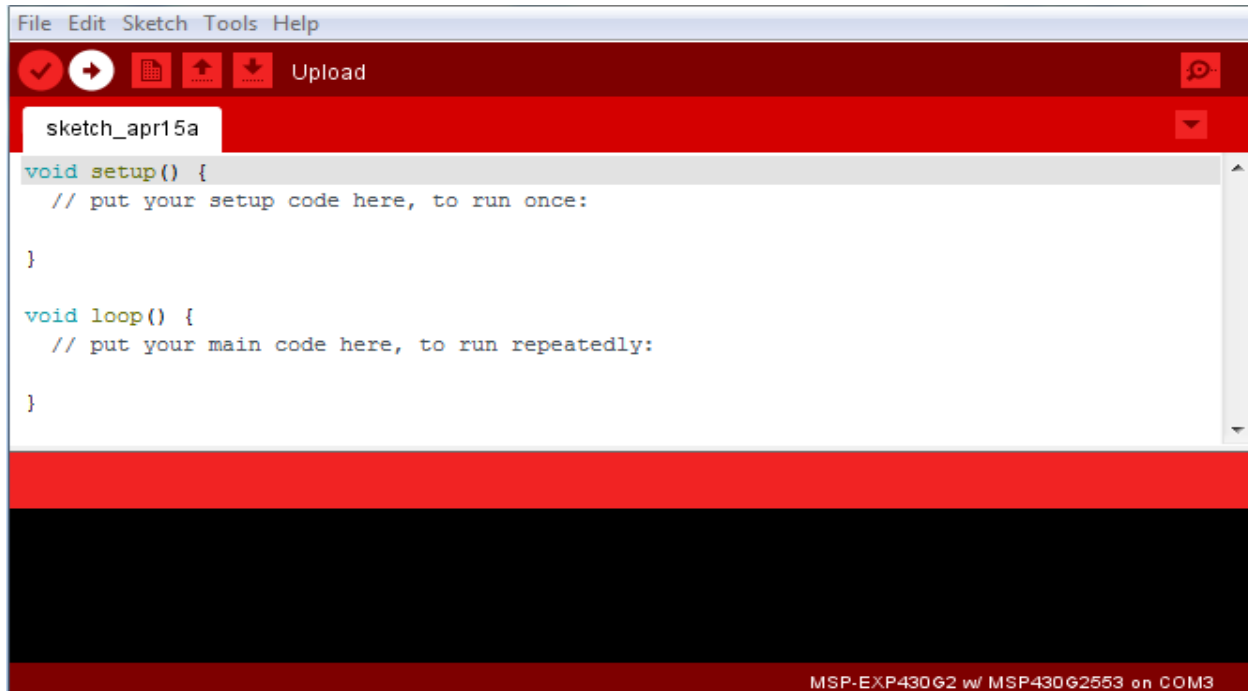


Fig: 4.4 Energia software Implementation

The Firebase DB is configured from nodeMCU using Arduino IDE. Initially we need to create an account in Google firebase data base. We can directly access the website and create new account by entering our Gmail and regarding details, then we need to create project enable the read and write rules to true for writing and retrieving the data to firebase as shown in figure 4.5.

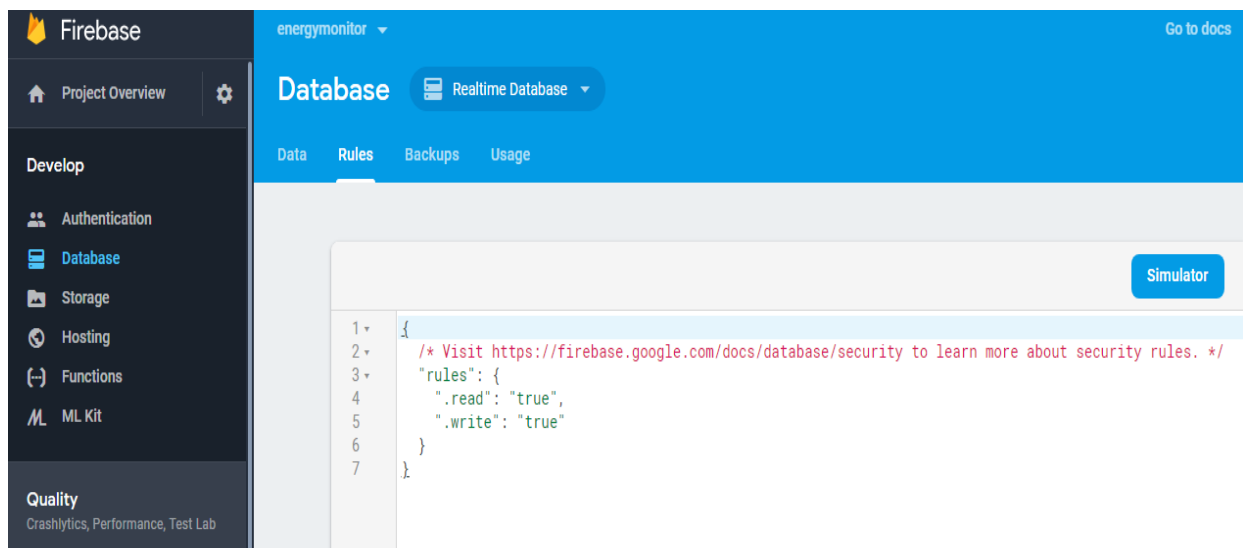


Fig: 4.5 Firebase Rules

The address of Firebase account is to be configured in Arduino IDE along with authentication ID for using the firebase, then we can upload and retrieve the data from firebase. We have to download the library for firebase from arduino IDE.

Here are the steps to download the firebase library.

- Open Arduino IDE and open sketch and click on include library
- Click manage library and search the firebase and install the latest version of firebase package.
- Click on file and open example to see whether firebase library is added or not.

Open the firebase example demo project and edit the program we our requirements like receive serial communication data and configuring the LAN and firebase account.

The entire three flow charts discussed in the previous chapter will be implemented in the format of energia sketch in the void loop() function.

4.3 Mobile Application Development:

As we mentioned before, the calculation of power consumed is performed in the mobile application, it is the crucial progress in the project. We had gone through different app developing platforms to built best application and finally developed with MIT APP inventor 2. We have to create an account to use the MIT APP inventor 2, which is free of cost. It is easiest way to develop an application for beginners. We just need some logical analysis like having experience in any programming language. Lot of online sessions are provided for app development. Many features are integrated in the tool, We can customize the frontend and backend of application as per our requirement. We will have two ends of applications. They are

1. **Designer :** It's where user can set the application display interface activities. We need to drag the components displayed left side of screen to the centre and can modify its name, dimensions, positions etc as per our requirement.
2. **Blocks:** It is used for backend development. We perform the operations on the components we dragged for our application in the form of connecting blocks logically.

The components we used in one screen can only do operation for that screen. Few components may not be visible in the user interface but they run in the background such as firebase, clock etc. The below figure 4.6 is the image of Designer window.

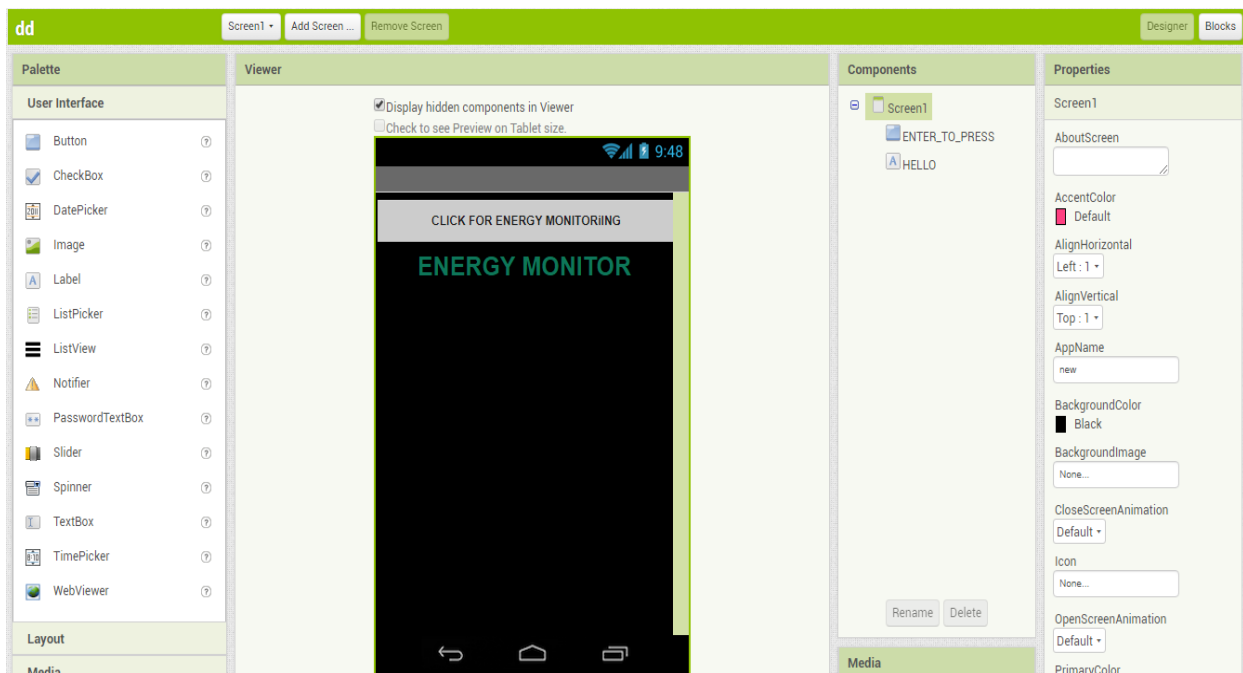


Fig: 4.6 Designer Window

We have used two screens one for logging and other for displaying the duration time and units of power consumed. We are using the components like firebase, timer in the second screen, so we

drag those components in screen 2. The usage of these components is done in the Block window. The figure 4.7 shows image of the blocks window.

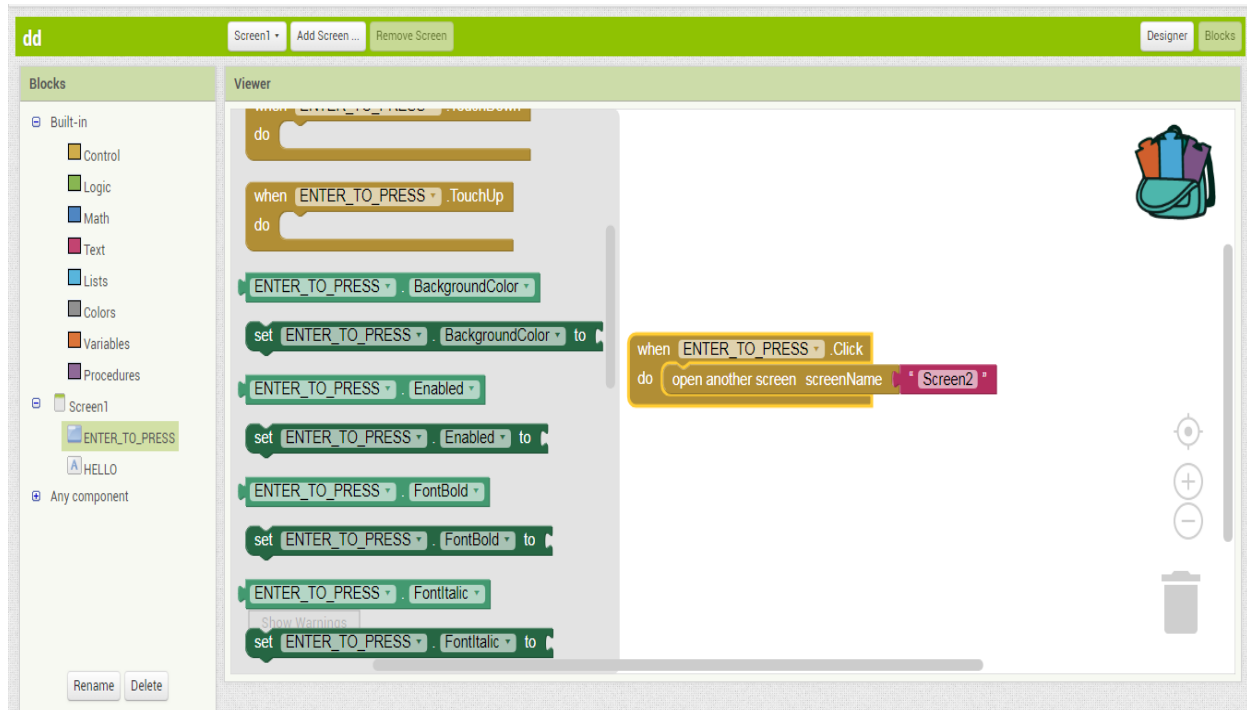


Fig 4.7 Block Window

We have to arrange the blocks of the components required in a meaningful way, i.e logically correct. It looks like programming C language in terms of blocks. All mathematical calculations can be taken place. Since we are taking data from Firebase DB we need a firebase component and timer component to calculate the time period from the start and stop variable from firebase DB. We also used the two labels for displaying time and power in units. We used three user interface buttons start, stop, pause to operate manually the time. The time is counted in seconds so in order calculate the power and displaying in the excel sheet we need to do some mathematical conversions, calculations, excel sheet operations. They are

1. Converting time from seconds to hours
2. Multiplying with power rating of device
3. Display in the mobile
4. Add to the excel sheet

To display the data in excel sheet we have to create new excel sheet with the elements in columns. The columns should be updated from the mobile application. The entire operation are done only when get the data from firebase DB, so the app should be updated continuously. To make this condition we need to select the right firebase block. The back end block development is shown in the figure 4.8.

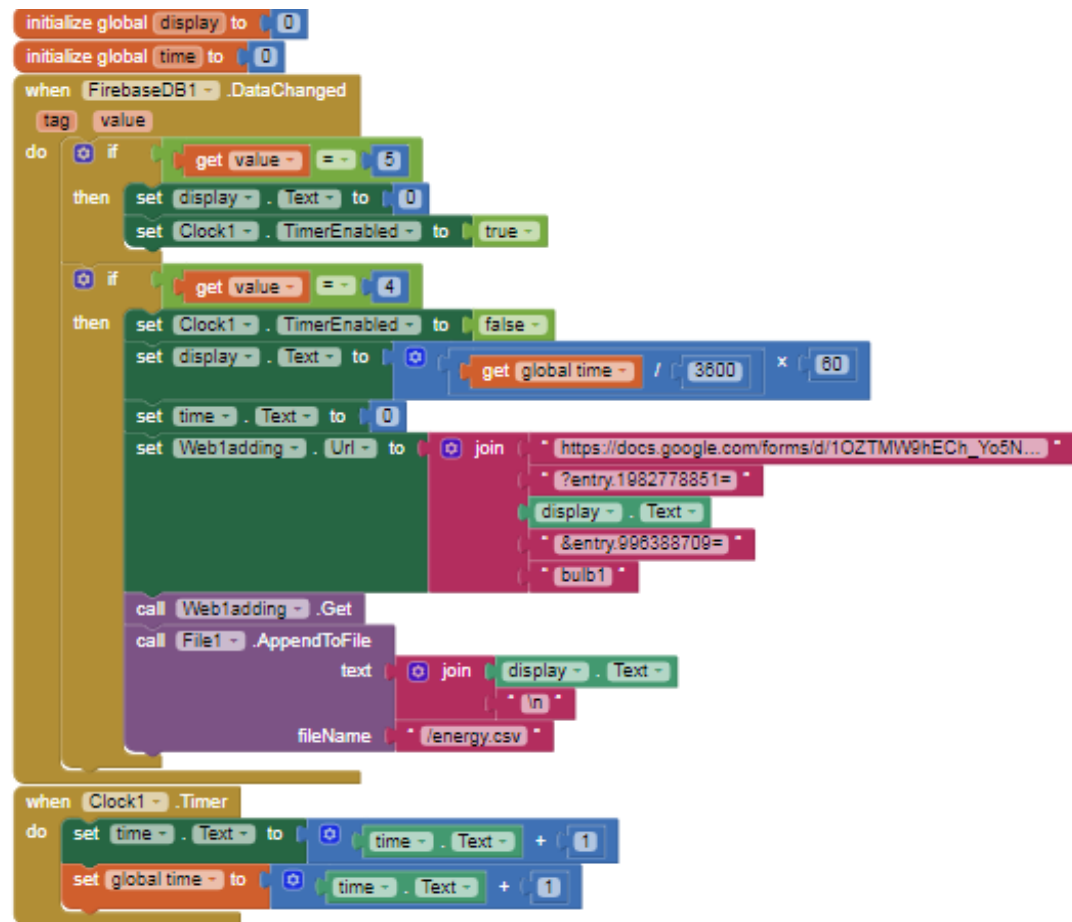


Fig: 4.8 Final Project Blocks

The working of above logical blocks is whenever the firebase gets new data it checks the value and enables or disables the timer. If the timer is disabled the time calculated till then is divided by 3600 and multiplied with 60 and displays in the app and also appends to excel sheets with the component name.

5. RESULTS:

5.1 Working Output:

This project gives the effective output. The project is executed practically and finally we are able to get the required result with low cost and full reliable. The following images give the output of the working program. Different images have been captured and explained. The below image is the first image when there is no one inside the room.



Fig: 5.1 Image Inside the Room Without Human

The second image shown in the Fig 5.2 is when a person is entering into the room. The first PIR sensor detect the motion and sends the signal to micro controller, due to that relay1 is set to low and therefore bulb1 is ON.



Fig: 5.2 Image Inside the Room With Human Entrance

The third image shown below in Fig 5.3 is the working of Bulb2 when the human is entered at another corner of the room. Second bulb is turned ON by setting the second Relay to zero because now PIR2 detects the motion and as per the algorithm mentioned it works. The first bulb is also in the ON state because the PIR1 sensor also covers the distance of PIR2 sensor.



Fig: 5.3 Image Inside the Room With Human at One Corner

The final image shown in the Fig 5.4 is when human is not in the motion for example like reading book. The project works excellent in this case without any delay. The PIR sensors Delays are set to minimum and output is sent to micro controller without any delay.



Fig: 5.4 Image Inside the Room With Human Reading Book

5.2 Firebase Output:

The data that is sent from MSP 430 through serial communication is received by NodeMCU and then transmitted to firebase DB continuously without any delay. The data in firebase can be seen in database console under the project name in real time. The figure 5.5 shows the data received from nodeMCU of value 4 or 5.

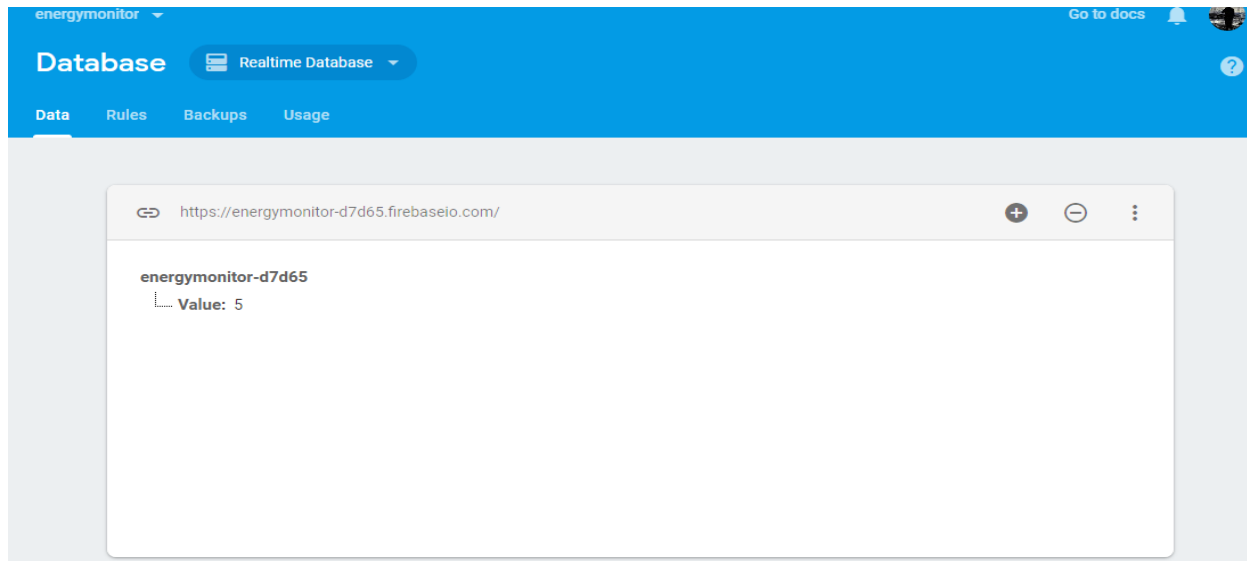


Fig: 5.5 Firebase Output

We can also get the amount of data storage consumed by these values in the graph as shown in fig: 5.6

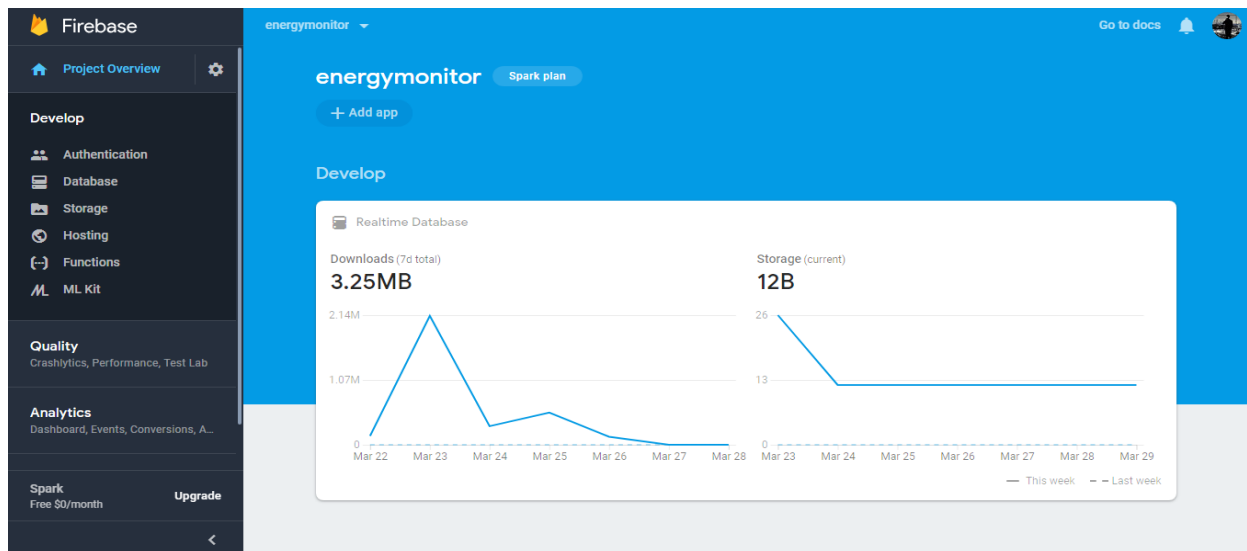


Fig: 5.6 Firebase Output storage

5.3 Mobile APP Output :

The duration of running electrical device and power consumed is displayed in screen2 as shown below. The power is displayed in terms of Watts. The figure 5.7 shown gives the output of mobile application. Both the screens of application are shown.

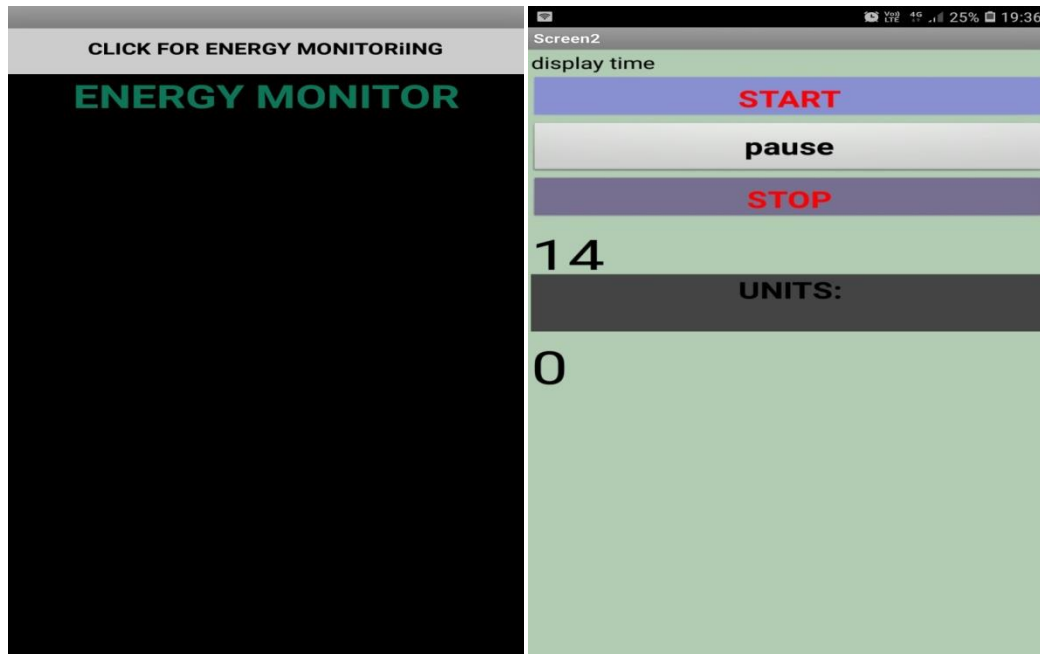
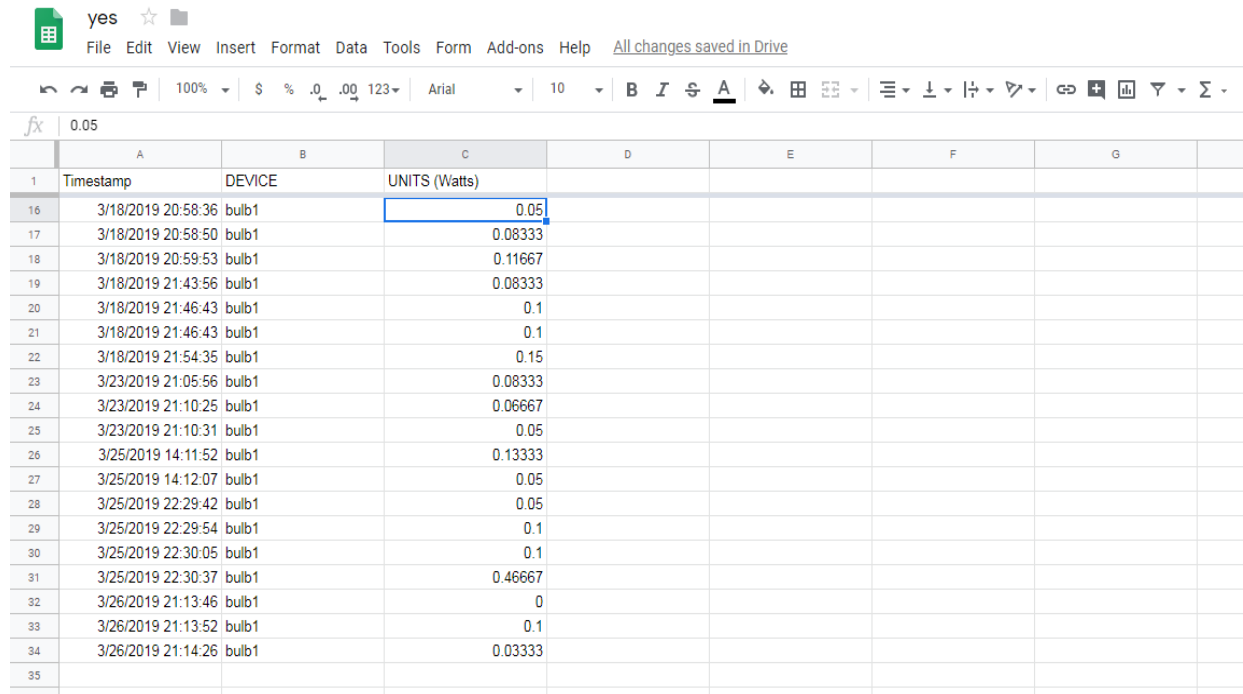


Fig: 5.7 Mobile Application Output

The above figure has two screens. Left picture is the screen 1 which has only functionality to open screen 2 and the right picture is screen 2 which displays the time and power consumed by device in watts. The screen 2 is displaying the time of 14 seconds it means that the bulb1 is being ON for 14 seconds and power is zero because it still did not turned off. When the bulb1 is turned off then units is displayed and timer is set to zero.

Excel sheet output

This is the final stage of the project. The calculated power in the mobile app is sent to excel sheet which is added in app. The data is continuously appended whenever it get updates firebase DB. The units consumed is displayed along with the device name and time stamp, so user can monitor without any difficulties.



	A	B	C	D	E	F	G
1	Timestamp	DEVICE	UNITS (Watts)				
16	3/18/2019 20:58:36	bulb1	0.05				
17	3/18/2019 20:58:50	bulb1	0.08333				
18	3/18/2019 20:59:53	bulb1	0.11667				
19	3/18/2019 21:43:56	bulb1	0.08333				
20	3/18/2019 21:46:43	bulb1	0.1				
21	3/18/2019 21:46:43	bulb1	0.1				
22	3/18/2019 21:54:35	bulb1	0.15				
23	3/23/2019 21:05:56	bulb1	0.08333				
24	3/23/2019 21:10:25	bulb1	0.06667				
25	3/23/2019 21:10:31	bulb1	0.05				
26	3/25/2019 14:11:52	bulb1	0.13333				
27	3/25/2019 14:12:07	bulb1	0.05				
28	3/25/2019 22:29:42	bulb1	0.05				
29	3/25/2019 22:29:54	bulb1	0.1				
30	3/25/2019 22:30:05	bulb1	0.1				
31	3/25/2019 22:30:37	bulb1	0.46667				
32	3/26/2019 21:13:46	bulb1	0				
33	3/26/2019 21:13:52	bulb1	0.1				
34	3/26/2019 21:14:26	bulb1	0.03333				
35							

Fig: 5.8 Excel Sheet output

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion:

Finally power consumed by the individual electric device is calculated at every instant. The end user can just perform the excel summing operation by selecting the period. This project provides the accurate and reliable results with low cost and easily understood concept. The Technology used in this project is also easily available. This project can be turned into product and made available into the market so that small scale industries can implement it. Even though this project is reliable there is a drawback.

The program gets complex when more number of devices are operated in single network, because different elements have to be considered. So we can divide into different groups so that it could be easy for developing. In final the project gets complex when more PIR sensors are considered. If three sensors are used then eight plus one cases have to be considered, for n PIR sensors 2 to the power of n cases plus one cases have to be considered. So for more PIR sensors the program gets complex and sometimes the controller cannot handle it. To overcome this solution we have to limit the number of PIR sensors to the controller or else the group the operation on relays.

6.2 Future work:

We would like to extend this project and make this as a product. The extension to the project could be the calculating the power automatically without making the end user to not even touch and also the calculating the power on different time basis like daily calculation, monthly calculation and yearly. This note can be verified and cross check with the monthly power bill etc.

The another extension can also be done that is since there is a restriction in using the number PIR sensors an algorithm can be designed such that the Code can be implemented automatically by just knowing PIR sensors used. This algorithm may be tough and little difficult to design, but if we are able to design it then this could be an ultimate solution for power saving. Many more different extensions can be done and different algorithms can be implemented by taking this project as a reference. The reduction of hardware components may also be possible if motion detecting sensor is available.

7. REFERENCES

- [1] <https://ieeexplore.ieee.org/document/8081933>
- [2] <https://www.electronicshub.org/automatic-room-lights-using-arduino-pir-sensor/>
- [3] <https://www.instructables.com/id/Smart-home-energy-management-system-SHEMMS/>
- [4] https://www.researchgate.net/publication/220832504_Home_Electric_Energy_Monitoring_System_Design_and_Prototyping
- [5] https://www.researchgate.net/publication/322074074_Mobile_Web_Energy_Monitoring_System_Using_DFRduino_Uno

8.APPENDICES

8.1MSP430 Demo Program:

```
#define LED RED_LED

int pr1 = 5 ;           //Global variable for starting timer in app
int pr2 = 4 ;           //Global variable for stoping timer in app

void setup()
{
    // initialize the digital pin as an output.
    pinMode(LED, OUTPUT);
    pinMode(14,OUTPUT);
    Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop()
{
    digitalWrite(LED, HIGH);
    Serial.print(i);
    Serial.print("\n");
    Serial.write(pr1);
    digitalWrite(14,LOW);
    delay(10000);
    digitalWrite(LED, LOW);
    Serial.print(i);
    Serial.print("\n");
    Serial.write(pr2);
```



```

    digitalWrite(14, LOW);

    delay(5000);

}

```

8.2 NodeMcu Program:

```

#include <ESP8266WiFi.h>

#include <FirebaseArduino.h>

// Set these to run example.

#define FIREBASE_HOST "energymonitor-d7d65.firebaseio.com"

#define FIREBASE_AUTH "3NdJs6oniCvwNnD9w8OtPOUGAIdYKiofhA4SNYZ6"

#define WIFI_SSID "Hotspot"

#define WIFI_PASSWORD "00000000"

void setup()
{
    Serial.begin(9600);

    // connect to wifi.

    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    Serial.print("connecting");

    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");

        delay(500);
    }

    Serial.println();

    Serial.print("connected: ");

    Serial.println(WiFi.localIP());
}

```

```

    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

void loop()
{
    int i = Serial.read(); //reading serial data and storing in a variable
    Firebase.setInt("Value", i); //sending data to firebase

    // handle error
    if (Firebase.failed())
    {
        Serial.print("setting /number failed:");
        Serial.println(Firebase.error());
        return;
    }
}

```

8.3 Main Program:

```

int uv=0; // Setting up constants
int pir1=0; // Initializing PIR1 variable to zero
int pir2=0; // Initializing PIR2 variable to zero
int pir1value; // Setting up sensor inputs
int pir2value; // Setting up sensor inputs
const int reldist = 61; // Put a reference distance here

const int trigPin = 8 ; //Put pin numbers(digital)
const int echoPin = 9; //Put pin numbers(digital)

```

```
const int relay1 = 14;           //Put pin numbers(digital)
const int relay2 = 15;           //Put pin numbers(digital)
const int pirsensor1 = 5;        //Put pin numbers(digital)
const int pirsensor2 = 6;        //Put pin numbers(digital)

// defines variables
int pr1 = 5;
int pr2 = 4;
long duration;
int distance;

void setup()
{
  pinMode(trigPin, OUTPUT);      // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);       // Sets the echoPin as an Input
  pinMode(pirsensor1, INPUT);    // Sets the pir1 as an Input
  pinMode(pirsensor2, INPUT);    // Sets the pir2 as an Input
  Serial.begin(9600);            // Starts the serial communication
}

void loop()
{

  pir1value=digitalRead(pirsensor1); //Reading values from pir sensors
  pir2value=digitalRead(pirsensor2); //Reading values from pir sensors

  digitalWrite(trigPin, LOW);     // Clears the trigPin
```

```
delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in
microseconds

duration = pulseIn(echoPin, HIGH);

//Calculating the distance

distance= duration*0.034/2;

// Prints the distance on the Serial Monitor

//logic

if(distance != reldist)

    ~u;

else

    u = 0;

if(pir1value || pir2value)

    u=1;
```

```

if(u && pir1)
    digitalWrite(relay1,LOW);
if(u && pir2)
    digitalWrite(relay2,LOW);
else
{
    digitalWrite(relay1,HIGH);
    digitalWrite(relay2,HIGH);
}

if((u==1) && (pir1value == 1) && (pir2value == 0))
{
    pir1=1;
    pir2=0;
}
else if((u==1) && (pir1value == 0) && (pir2value == 1))
{
    pir1=0;
    pir2=1;
}
else if((u==1) && (pir1value == 0) && (pir2value == 0))
{
    if((pir1 == 1) && (pir2 == 0)){
        pir1=1;
        pir2=0;
    }

    else if((pir2==1) && (pir1==0))

```

```

{
    pir1=0;
    pir2=1;
}
Else
{
    pir1=1;
    pir2=1;
}
}
else
{
    pir1=0;
    pir2=0;
}

if((pir1 == 1) && (pir2 == 0))
{
    digitalWrite(relay1,LOW);           //turning Bulb1 ON
    digitalWrite(relay2,HIGH);         //turning Bulb2 OFF
    Serial.write(pr1);
}
else if((pir2 == 1) && (pir1 == 0))
{
    digitalWrite(relay1,HIGH);         //turning Bulb1 OFF
    digitalWrite(relay2,LOW);          //turning Bulb2 ON
    Serial.write(pr2);
}

```

```

}
else if((pir1 == 0) && (pir2 == 0))
{
    digitalWrite(relay1,HIGH);          //turning Bulb1 OFF
    digitalWrite(relay2,HIGH);          //turning Bulb2 OFF
    Serial.write(pr2);
}
else
{
    digitalWrite(relay1,LOW);           //turning Bulb1 ON
    digitalWrite(relay2,LOW);           //turning Bulb2 ON
    Serial.write(pr1);
}
}

```

8.4 Ultrasonic Sensor Reference distance measurement code:

```

const int trigPin = 5;                //Put pin numbers(digital)
const int echoPin = 6;                //Put pin numbers(digital)

long duration;
int distance;

void setup()
{
    pinMode(trigPin, OUTPUT);          // Sets the trigPin as an Output
    pinMode(echoPin, INPUT);           // put your setup code here, to run
once:

```

```
Serial.begin(9600);  
  
}  
  
void loop()  
{  
    digitalWrite(trigPin, LOW);      // Clears the trigPin  
    delayMicroseconds(2);  
  
    // Sets the trigPin on HIGH state for 10 micro seconds  
  
    digitalWrite(trigPin, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(trigPin, LOW);  
  
    // Reads the echoPin, returns the sound wave travel time in  
    microseconds  
  
    duration = pulseIn(echoPin, HIGH);  
  
    // Calculating the distance  
  
    distance= duration*0.034/2;  
  
    Serial.print(distance); // Prints the distance on the Serial Monitor  
}
```