SMART LIGHTING USING INTERNET OF THINGS

Project submitted to

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

In partial fulfillment of the requirements for the degree

of **MASTER OF COMPUTER APPLICATIONS,**

2018 BY

PABITRA KUMAR SADHUKHAN

Examination Roll: MCA186015

Registration No: 133677 of 2015-2016

Under the guidance of

Dr. Debesh Kumar Das

Professor, Department of Computer Science Engineering

Jadavpur University

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

TO WHOM IT MAY CONCERN

*I hereby recommend that the project entitled "**Smart Lighting using Internet of Things**" prepared under my supervision and guidance at Jadavpur University, Kolkata by PABITRA KUMAR SADHUKHAN (Reg. No. 133677 of 2015 – 16, Class Roll No. 001510503015 of 2015-16), may be accepted in partial fulfillment for the degree of Master of Computer Applications in the Faculty of Engineering and Technology, Jadavpur University, during the academic year 2017 – 2018. I wish him every success in life.*

……..
Prof. (Dr.) Ujjwal Maulik
Head of the Department
Department of Computer Science and Engineering
Jadavpur University, Kolkata – 700032.

……
Prof. (Dr.) Debesh Kumar Das
Project Supervisor,
Department of Computer Science and Engineering
Jadavpur University, Kolkata – 700032.

……..
Prof. (Dr.) Chiranjib Bhattacharjee
Dean, Faculty council of Engg. & Tech.
Jadavpur University, Kolkata – 700032.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC PROJECT

I hereby declare that this project contains literature survey and original research work by the undersigned candidate, as part of his MASTER OF COMPUTER APPLICATIONS studies. All information in this document have been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material results that are not original to this work.

NAME: PABITRA KUMAR SADHUKHAN

ROLL NUMBER: 001510503015

PROJECT TITLE: "SMART LIGHTING USING INTERNET OF THINGS."

SIGNATURE WITH DATE:

JADAVPUR UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE OF APPROVAL

The forgoing project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR
EVALUATION OF PROJECT:

1 .....
(Signature of Examiner)

2 .....
(Signature of Examiner)

ACKNOWLEDGEMENT

I express my honest and sincere thanks and humble gratitude to my respected teacher and guide *Prof. (Dr.) Debesh Kumar Das*, Professor of the Department of Computer Science & Engineering, Jadavpur University, for his exclusive guidance and entire support in completing and producing this project successfully. I am very much indebted to him for the constant encouragement, and continuous inspiration that he has given to me. The above words are only a token of my deep respect towards him for all he has done to take my project to the present shape.

I would like to thank *Mr. Manash Chaudhuri, CEO of Corelynx Solutions Pvt. Ltd., Mr. Ashish Majumder, Our Project Leader, Mr. Joydev Payra, Our Team Leader, and Mr. Shouvik Ghosh, Head of Linux System Administration, Corelynx Solutions Pvt. Ltd.* for their valuable support and suggestions to the activities of the project.

Finally, I convey my real sense of gratitude and thankfulness to my friend as well as project partner Mr. Arunoday Samanta for being an endless source of optimism and positive thoughts; and last but not the least, my father & mother for their unconditional support, without which I would hardly be capable of producing this huge work.

.....

(Signature with Date)

PABITRA KUMAR SADHUKHAN

Examination Roll: MCA186015

Registration No: 133677 of 2015 – 2016

Abstract

We're entering a new era of computing technology that many are calling the Internet of Things (IoT). Machine to machine, machine to infrastructure, machine to environment, the Internet of Everything, the Internet of Intelligent Things, intelligent systems—call it what you want, but it's happening, and its potential is huge.

We see the IoT as billions of smart, connected "things" (a sort of "universal global neural network" in the cloud) that will encompass every aspect of our lives, and its foundation is the intelligence that embedded processing provides. The IoT is comprised of smart machines interacting and communicating with other machines, objects, environments and infrastructures. As a result, huge volumes of data are being generated, and that data is being processed into useful actions that can "command and control" things to make our lives much easier and safer—and to reduce our impact on the environment. The creativity of this new era is boundless, with amazing potential to improve our lives. The following thesis is an extensive reference to the possibilities, utility, applications and the evolution of the Internet of Things.

Smart led street lighting system aims for designing and executing the advanced development in IOT for energy saving of street light, the best solution for electrical power wastage is automation of street light, the manual operation of
the lighting system is completely eliminate. A method for modifying street light illumination by using sensor at minimum electrical energy consumption ,when object presence is detected, street lights glow at their brightest mode, else they stay in the dim mode during night time Internet of things (IOT) is used to visualize the real time updates of street processing and notifying the changes occur. This shall reduce heat emissions, power consumption, maintenance and replacement costs .

Contents

Contents

Chapter 1

Introduction

The Internet of Things may be a hot topic in the industry but it's not a new concept. In the early 2000's, Kevin Ashton was laying the groundwork for what would become the Internet of Things (IoT) at MIT's AutoID lab. Ashton was one of the pioneers who conceived this notion as he searched for ways that Proctor & Gamble could improve its business by linking RFID information to the Internet. The concept was simple but powerful. If all objects in daily life were equipped with identifiers and wireless connectivity, these objects could be communicate with each other and be managed by computers.

The term Internet of Things (often abbreviated IoT*)* was coined by industry researchers but has emerged into mainstream public view only more recently. IoT is a network of physical devices, including things like smartphones, vehicles, home appliances, and more, that connect to and exchange data with computers.

Some claim the Internet of Things will completely transform how computer networks are used for the next 10 or 100 years, while others believe IoT is simply hype that won't much impact the daily lives of most people.

1.1 Internet of Things

The **Internet of Things** (**IoT**) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these objects to connect and exchange data. Each thing is uniquely identifiable through its embedded computing system but is able to inter-operate within the existing Internet infrastructure.

The figure of online capable devices increased 31% from 2016 to 8.4 billion in 2017. Experts estimate that the IoT will consist of about 30 billion objects by 2020. It is also estimated that the global market value of IoT will reach $7.1 trillion by 2020.

The IoT allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit in addition to reduced human intervention. When IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical system, which also encompasses technologies such as smart grides, virtual power plants, smart homes, intelligent transportation and smart cities.

"Things", in the IoT sense, can refer to a wide variety of devices such as heart monitoring implants, biochip transponders on farm animals, cameras streaming live feeds of wild animals in coastal waters, automobiles with built-in sensors, DNA analysis devices for environmental / food /pathogen monitoring, or field operation devices that assist firefighters in search and rescue operations. Legal scholars suggest regarding "things" as an "inextricable mixture of hardware, software, data and service".

These devices collect useful data with the help of various existing technologies and then autonomously flow the data between other devices.

There seems to be a general consensus that term "the Internet of things" was coined by Kevin Ashton of Procter & Gamble, later MIT's Auto-ID Center, in 1999.The first written and referable source that mentions the Internet of Things seems to be the White Paper published by the MIT Auto-ID Center in November 2001 (but made public only in February 2002), which cites an earlier paper from October 2000.

The first research article mentioning the Internet of Things appears to be, which was preceded by an article published in Finnish in January 2002. The implementation described there was developed by Kary Främling and his team at Helsinki University of Technology in Finland. Contrary to the rather RFID and Supply Chain Management view of the Internet of Things, the vision of the Internet of Things presented there was closer to the modern one, i.e. an information system infrastructure for implementing smart, connected objects.

1.2 Arduino :

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on our computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – we can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

1.3 Sensors:

A sensor is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena.

Looking at the definition of IoT, as well as some of the benefits and real-world use cases, it's critical to examine how sensors play a role in this equation. While software, machine-to-machine learning and other technologies work together to analyze data from physical objects – the sensors are key to gathering the information. If software is the brains of the IoT, sensors are the nervous system collecting continuous streams of data to be processed. Industrial systems rely on sensors for reliable, consistent and accurate data in all aspects of automation. One could even argue the IoT is nothing without sensors to measure parameters such as strain, temperature, position, and pressure.

1.4 MQTT Server:

MQTT(Message Queuing Telemetry Transport) is an ISO Standard publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The publish subscribe messaging pattern requires a message broker.

MQTT enables resource-cos strained IoT devices to send, or publish, information about a given topic to a server that functions as an MQTT message broker. The broker then pushesthe information out to those clients that have previously subscribed to the client's topic. To a human, a topic looks like a hierarchical file path. Clients can subscribe to a specific level of a topic's hierarchy or use a wild card character to subscribe to multiple levels.

1.5 Apache Kafka :

Apache Kafka is an open source streaming-processing software platform developed by the Apache Software foundation, written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Its storage layer is essentially a "massively scalable pub/sub message queue architected as a distributed transaction log," making it highly valuable for enterprise infrastructures to process streaming data.

Additionally, Kafka connects to external systems (for data import/export) via Kafka Connect and provides Kafka Streams, a Java stream processing library.

1.6 Web Server using Nodejs:

A Web Server is a software application which handles HTTP requests sent by the HTTP client, like web browsers, and returns web pages in response to the clients. Web servers usually deliver html documents along with images, style sheets, and scripts.

Most of the web servers support server-side scripts, using scripting languages or redirecting the task to an application server which retrieves data from a database and performs complex logic and then sends a result to the HTTP client through the Web server.

Apache web server is one of the most commonly used web servers. It is an open source project.

1.7 Database – MongoDB:

A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.

MongoDB is a free and open-source cross-platform document-oriented databse program. Classified as a NoSql database program, MongoDB uses jSon-like documents with schemas. MongoDB is developed by MongoDB Inc.

1.8 Visualization:

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, you can take the concept a step further by using technology to drill down into charts and graphs for more detail, interactively changing what data you see and how it’s processed.

1.9 Conclusion:

This chapter has presented the overview of the hardware, software and Database, which are needed in the project.

Chapter 2

Hardware Requirement

2.1 Mega 2560 Controller Board:

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560(datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs),16 analog inputs, 4 UARTs(hardware serial ports), a 16 MHzcrystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Technical Specification:

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

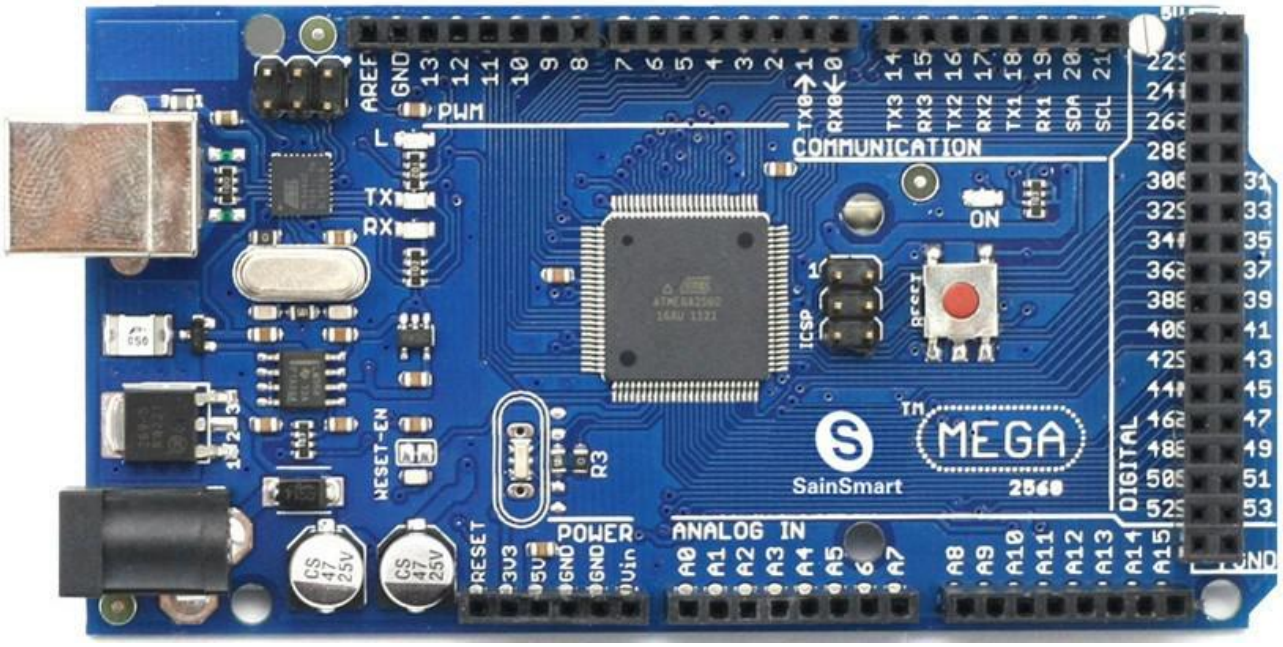


Fig 2.1 Mega 2560 Controller

2.2 830 Tie-Points Breadboard:

A **breadboard** is a construction base for protyping of electronics. Originally it was literally a bread board, a polished piece of wood used for slicing bread. In the 1970s the **solderless breadboard** (**plugboard**, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these.Because the solderless breadboard does not require soldering, it is reusable

830 tie point solderless "plug-in" breadboards provide a quick way to build and test circuits for experimentation or when learning electronics.

Solderless BreadBoard Specifications:

Body Material:	White ABS Plastic with Black Printed Legend
Body Material:	White ABS Plastic with Color Printed Legend
Body Material:	Transparent ABS Plastic with Color
Printed LegendHole Pitch/Style:	0.1" (2.54 mm), Square

Wire Holes ABS Heat Distortion Temperature: 84°C. (183°F.)

Spring Clip Contact:	Phosphor Bronze with Plated Nickel Finish
Contact Life:	50,000 insertions
Rating:	36 Volts, 2 Amps
Insertion Wire Size:	21 to 26 AWG 0.016 to 0.028 inches diameter (0.4 to 0.7mm diameter)
Backing:	Peelable adhesive tape for attaching to a surface. Metal back plate provided with 830 tie point breadboard.
Metal Back Plate Thickness:	0.031 inches (0.8mm)

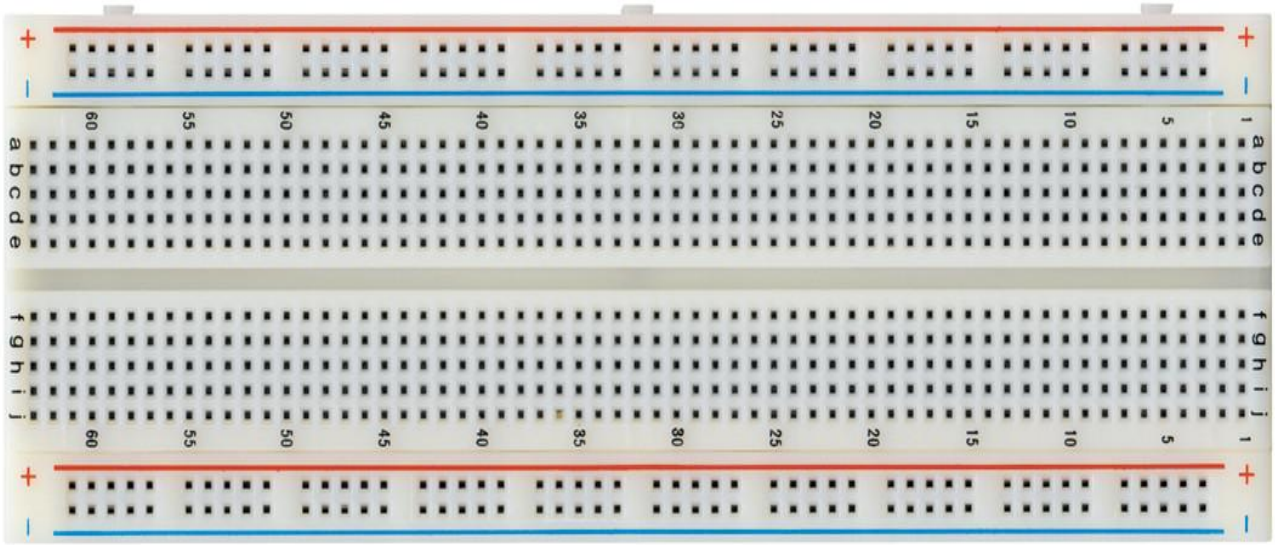


Fig 2.2 830 Tie Point Breadboard

2.3 Photoresistor (Photocell):

The photocell used is of a type called a light dependent resistor, sometimes called an LDR. As the name suggests, these components act just like a resistor, except that the resistance changes in response to how much light is falling on them. This one has a resistance of about 50 kΩ in near darkness and 500 Ω in bright light. To convert this varying value of resistance into something we can measure on an MEGA2560 R3 board's analog input, it needs to be converted into a voltage. The simplest way to do that is to combine it with a fixed resistor.

Technical Specification:

Size:	Round, 5mm (0.2") diameter. (Other photocells can get up to 12mm/0.4" diameter!)
Resistance range:	200KΩ (dark) to 10KΩ (10 lux brightness)
Sensitivity range:	CdS cells respond to light between 400nm (violet) and 600nm (orange) wavelengths, peaking at about 520nm (green).
Power supply:	pretty much anything up to 100V, uses less than 1mA of current on average (depends on power supply voltage)

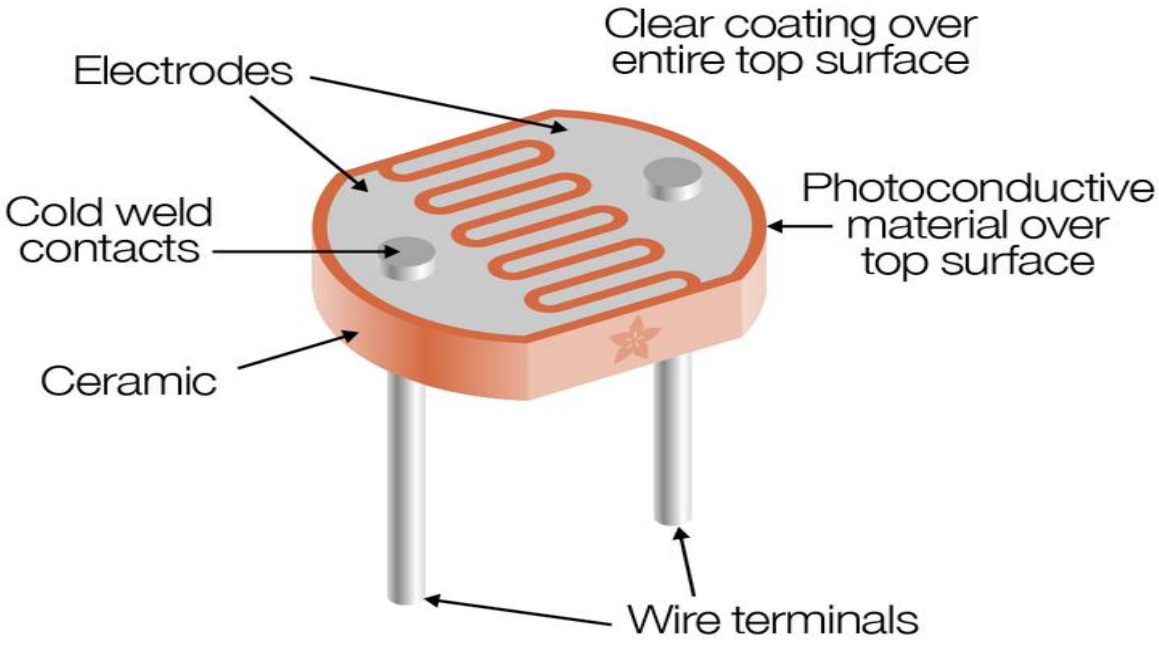


Fig 2.3 Photocell

2.4 SIM800L Module and SIM:

Mini GSM / GPRS breakout board is based on SIM800L module, supports quad-band GSM/GPRS network, available for GPRS and SMS message data remote transmission.The board features compact size and low current consumption. With power saving technique, the current consumption is as low as 1mA in sleep mode.It communicates with microcontroller via UART port, supports command including 3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands. It's operating voltage: 3.7 ~ 4.2V , peak current: 1A .

Technical Specification:

Quad-band 850/900/1800/1900MHz

Connect onto any global GSM network with any 2G SIM (in the USA, T- Mobile is suggested).

Make and receive voice calls using a headset or an external 8 speaker and electret microphone.

Send and receive SMS messages.

Send and receive GPRS data (TCP/IP, HTTP, etc.) .

Scan and receive FM radio broadcasts.

Lead out buzzer and vibrational motor control port.

AT command interface with "auto baud" detection.

SIM800L With Arduino:

SIM800 is one of the most commonly used GSM module among hobbyists and Arduino community. Even though AT command reference is available with a quick Google search, it is not very easy for a beginner to properly understand and use Arduino with SIM800. Therefore, this post summarizes how a beginner could interact with SIM800 using Arduino and in few future posts we'll be going ahead with several other real life use cases discussing how SIM800 can be used with Arduino effectively.

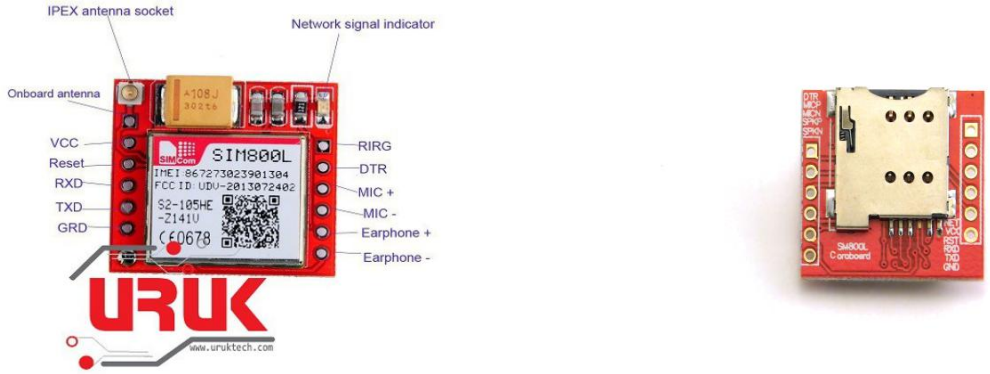


Fig 2.4 SIM800L

2.5 Light-Emitting Diode(LED):

A **light-emitting diode** (**LED**) is a two-lead semiconductor light-source. It is a p-n junction diode that emits light when activated. When a suitable current is applied to the leads, electronics are able to recombine with electron hole within the device, releasing energy in the form of photons. This effect is called electro-luminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor. LEDs are typically small (less than 1 mm^2) and integrated optical components may be used to shape the radiation pattern.

Specification:

Super Bright White:

Intensity: 6,000mcd
Colour Freq: 6000-6500
Viewing Angle: 28º
Lens: Water Clear

Voltage: 2.6v-2.9v

Typical: 2.8v
Current: 18mA

Fig 2.5.1 Super Bright White

Super Bright Warm White:

Intensity: 5,500mcd
Colour Freq: 2800-3200
Viewing Angle: 28º
Lens: Water Clear

Voltage: 2.5v-2.9v

Typical: 2.7v
Current: 18mA

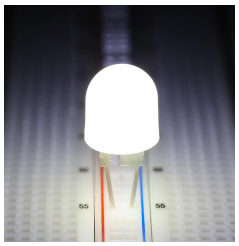


Fig 2.5.2 Super Bright warm White

Super Bright Blue:

Intensity: 2,000mcd
Colour Freq: 470-475nm
Viewing Angle: 28º
Lens: Water Clear

Voltage: 2.6v-2.8v

Typical: 2.8v
Current: 18mA

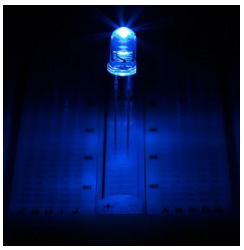


Fig 2.5.3 Super Bright Blue

Super Bright Green:

Intensity: 3,500mcd
Colour Freq: 520-570nm
Viewing Angle: 28º
Lens: Water Clear

Voltage: 2.4v-2.7v

Typical: 2.6v
Current: 20mA

Fig 2.5.4 Super Bright Green

Super Bright Yellow:

Intensity: 4,500mcd

Colour Freq: 580-590nm
Viewing Angle: 18º
Lens: Water Clear

Voltage: 1.8v-2.0v

Typical: 1.9v
Current: 18mA

Fig 2.5.5 Super Bright Yellow

Super Bright Red:

Intensity: 4,500mcd

Colour Freq: 620-628nm
Viewing Angle: 18º
Lens: Water Clear

Voltage: 1.6v-1.9v

Typical: 1.8v
Current: 18mA

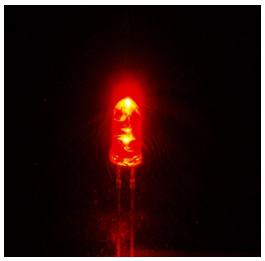


Fig 2.5.1 Super Bright Red

2.6 74hc595 IC:

The **74HC595** is a very handy **IC** used in many microcontroller projects. You clock in 8 bits of data (like, on/off settings for 8 LEDs) via two lines, and when you toggle a third line, it pops these settings out on 8 outputs on the **IC**.

Technical Specifications:

8-bit

Logic Family : HC

Logical Function : Shift Register

Operating Supply Voltage (Typ) : 5V

Output Type : 3-State

Package Type : DIP

Propagation Delay Time : 265ns

Operating Temp Range : -40C to 125C

- Operating Supply Voltage (Min) : 2V

Operating Supply Voltage (Max) : 6V

Fig 2.6 74hc595 IC

2.7 Resistor and M-M Wires:

A **resistor** is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, **resistors** are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.

In this project, we have used 1k ohm resistor and 220 ohm resistors.

Fig 2.7.1 1K Ohm Resistor

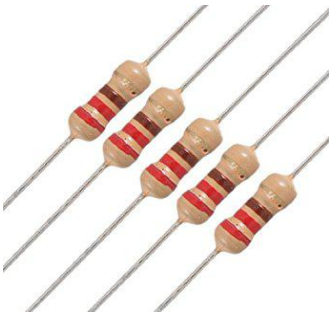


Fig 2.7.2 220 Ohm Resistors

A **jump wire** (also known as jumper, jumper wire, jumper cable, DuPont wire, or DuPont cable– named for one manufacturer of them) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them– simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

In this project, we have used only m-m jumper wires

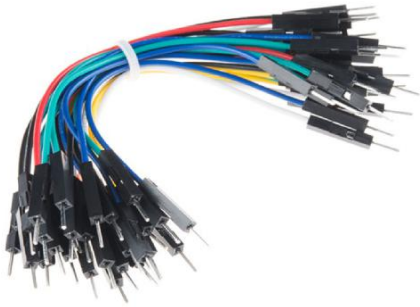


Fig 2.7.3 M-M Jumper Wire

Fig 2.7.4 F-F Jumper Wire

Fig 2.7.5 F-M Jumper Wire

2.8 9V 1A Adapter:

An (electrical) **adapter** or **adaptor** is a device that converts attributes of one electrical device or system to those of an otherwise incompatible device or system. Some modify power or signal attributes, while others merely adapt the physical form of one electrical connector to another.

Fig 2.8 9V 1A Adapter

2.9 USB Cable:

USB, short for **Universal Serial Bus**, is an industry standered that was developed to define cables, connectors and protocols for connection, communication, and power supply between personal computers and their peripheral devices

There have been three generations of USB specifications:

USB 1.*x*
USB 2.0, with multiple updates and additions
USB 3.*x*

Fig 2.9 USB Cable

Chapter 3

3 Software Requirement

3.1 Operating System – Ubuntu server 16.04 LTS:

What is operating System ?

An **operating system(OS)** is system software that manages computer hardware and software resources and provides common services for computer programs.Time-sharing operating systems schedule taskes for efficient use of the system and may also include accounting software for cost allocation of processor time,mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware,although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer– from cellular phones and video game consoles to web servers and supercomputers.

Example:- Microsoft Windows(like Windows 10,Windows 8,Windows 7,Windows vista, and,Windows XP, Apple's MacOS (formerly OS X), iOS, Chrome OS, BlackBerry Tablet OS, and flavors of the open source operating system Linux or Ubuntu

Why Ubuntu Linux is used in our project ?

Ubuntu is free. We can download any version of it and use it for any purpose. Upgrades are also free. There's no need for any anti-virus software or anti-spyware applications on Linux, which comes with a personal firewall, if we want to use it.

Linux also comes to us with a free office suite,OpenOffice.org that includes Microsoft-compatible applications. They look and behave so much like Microsoft's office suite that you may never realize any difference between the two.

3.2 MQTT (Message Queue Telemetry Transport):

What is MQTT ?

MQTT (MQ Telemetry Transport) is a lightweight messaging protocol that provides resource-constrained network clients with a simple way to distribute telemetry information. The protocol, which uses a publish/subscribe communication pattern, is used for machine-to-machine(M2M) communication and plays an important role in the internet of things(IOT).

MQTT enables resource-constrained IoT devices to send, or publish, information about a given topic to a server that functions as an MQTT message broker. The broker then pushes the

information out to those clients that have previously subscribed to the client's topic. To a human, a topic looks like a hierarchical file path. Clients can subscribe to a specific level of a topic's hierarchy or use a wild-card character to subscribe to multiple levels.

Why is MQTT needed ?

The MQTT protocol is a good choice for wireless networks that experience varying levels of latency due to occasional bandwidth constraints or unreliable connections. Should the connection from a subscribing client to a broker get broken, the broker will buffer messages and push them out to the subscriber when it is back online. Should the connection from the publishing client to the broker be disconnected without notice, the broker can close the connection and send subscribers a cached message with instructions from the publisher.

How does MQTT work?

An MQTT session is divided into four stages: connection, authentication, communication and termination. A client starts by creating a TCP/IP connection to the broker by using either a standard port or a custom port defined by the broker's operators. When creating the connection, it is important to recognize that the server might continue an old session if it is provided with a reused client identity.

The standard ports are 1883 for non-encrypted communication and 8883 for encrypted communication using SSL/TLS. During the SSL/TLS handshake, the client validates the server certificate to authenticate the server. The client may also provide a client certificate to the broker during the handshake, which the broker can use to authenticate the client. While not specifically part of the MQTT specification, it has become customary for brokers to support client authentication with SSL/TLS client-side certificates.

Because the MQTT protocol aims to be a protocol for resource-constrained and IoT devices, SSL/TLS might not always be an option and, in some cases, might not be desired. In such cases, authentication is presented as a clear-text username and password that is sent by the client to the server as part of the CONNECT/CONNACK packet sequence. Some brokers, especially open brokers published on the internet, will accept anonymous clients. In such cases, the username and password are simply left blank.

MQTT is called a lightweight protocol because all its messages have a small code footprint. Each message consists of a fixed header – 2 bytes-- an optional variable header, a message payload that is limited to 256 MB of information and a quality of service (QoS) level.

The three different quality of service levels determine how the content is managed by the MQTT protocol. Although higher levels of QoS are more reliable, they have more latency and bandwidth requirements, so subscribing clients can specify the highest QoS level they would like to receive.

The simplest QoS level is unacknowledged service. This QoS level uses a PUBLISH packet sequence; the publisher sends a message to the broker one time and the broker passes the message to subscribers one time. There is no mechanism in place to make sure the message has been received correctly, and the broker does not save the message. This QoS level may also be referred to as at most once, QoS0, or fire and forget.

The second QoS level is acknowledged service. This QoS level uses a PUBLISH/PUBACK packet sequence between the publisher and its broker, as well as between the broker and subscribers. An acknowledgement packet verifies that content has been received and a retry mechanism will send the original content again if an acknowledgement is not received in a timely manner. This may result in the subscriber receiving multiple copies of the same message. This QoS level may also be referred to as at least once or QoS1.

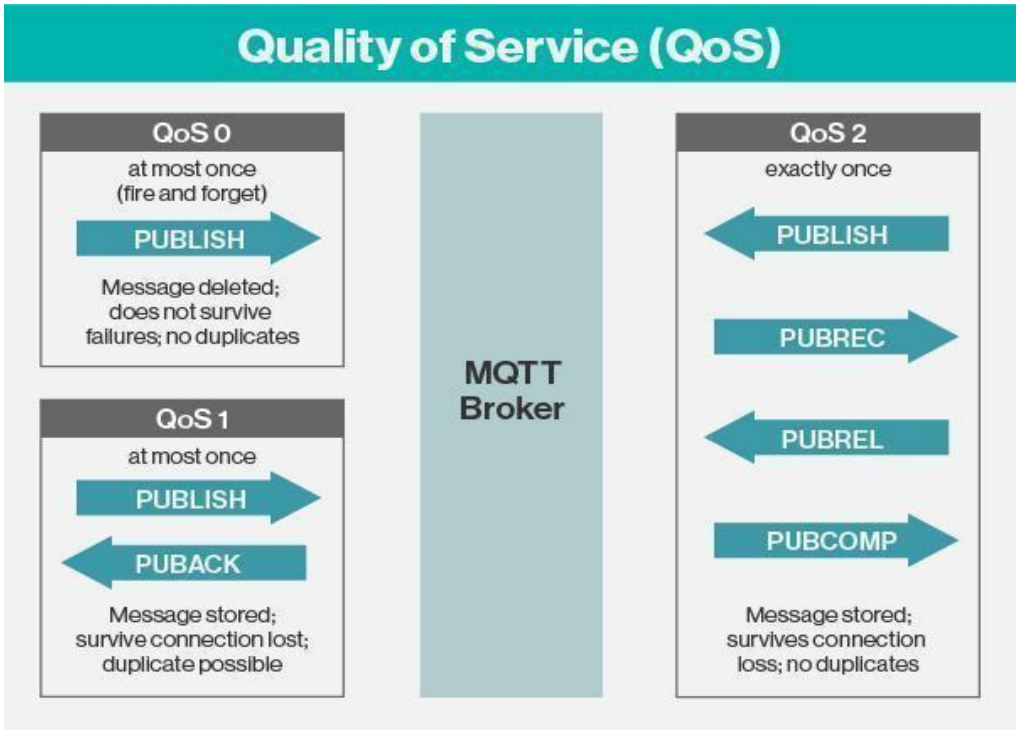


Fig 3.2 MQTT Broker

The third QoS level is assured service. This QoS level delivers the message with two pairs of packets. The first pair is called PUBLISH/PUBREC, and the second pair is called PUBREL/PUBCOMP. The two pairs ensure that, regardless of the number of retries, the message will only be delivered once. This QoS level may also be referred to as exactly once or QoS2.

3.3 Apache Kafka:

What is Apache Kafka ?

Apache Kafka is a distributed streaming platform capable of handling trillions of events a day. Initially conceived as a messaging queue, Kafka is based on an abstraction of a distributed commit log. Since being created and open sourced by LinkedIn in 2011, Kafka has quickly evolved from messaging queue to a full-fledged streaming platform.

As a streaming platform, Apache Kafka provides low-latency, high-throughput, fault-tolerant publish and subscribe pipelines and is able to process streams of events. Kafka provides reliable, millisecond responses to support both customer-facing applications and connecting downstream systems with real-time data.

How does Apache kafka work ?

Kafka is simply a collection of topics split into one or more partitions. A Kafka partition is a linearly ordered sequence of messages, where each message is identified by their index (called as offset). All the data in a Kafka cluster is the disjointed union of partitions. Incoming messages are written at the end of a partition and messages are sequentially read by consumers. Durability is provided by replicating messages to different brokers.

Kafka provides both pub-sub and queue based messaging system in a fast, reliable, persisted, fault-tolerance and zero downtime manner. In both cases, producers simply send the message to a topic and consumer can choose any one type of messaging system depending on their need. Let us follow the steps in the next section to understand how the consumer can choose the messaging system of their choice.

Work-flow of Pub-Sub Messaging:

Following is the step wise work-flow of the Pub-Sub Messaging −

Producers send message to a topic at regular intervals.

Kafka broker stores all messages in the partitions configured for that particular topic. It ensures the messages are equally shared between partitions. If the producer sends two messages and there are two partitions, Kafka will store one message in the first partition and the second message in the second partition.

Consumer subscribes to a specific topic.

Once the consumer subscribes to a topic, Kafka will provide the current offset of the topic to the consumer and also saves the offset in the Zoo-keeper ensemble.

Consumer will request the Kafka in a regular interval (like 100 Ms) for new messages.

Once Kafka receives the messages from producers, it forwards these messages to the consumers.

Consumer will receive the message and process it.

Once the messages are processed, consumer will send an acknowledgement to the Kafka broker.

Once Kafka receives an acknowledgement, it changes the offset to the new value and updates it in the Zoo-keeper. Since offsets are maintained in the Zoo-keeper, the consumer can read next message correctly even during server outrages.

This above flow will repeat until the consumer stops the request.

Consumer has the option to rewind/skip to the desired offset of a topic at any time and read all the subsequent messages.

Role of Zoo-Keeper:

A critical dependency of Apache Kafka is Apache Zoo-keeper, which is a distributed configuration and synchronization service. Zoo- keeper serves as the coordination interface between the Kafka brokers and consumers. The Kafka servers share information via a Zoo-keeper cluster. Kafka stores basic meta-data in Zoo-keeper such as information about topics, brokers, consumer offsets (queue readers) and so on.

Since all the critical information is stored in the Zoo-keeper and it normally replicates this data across its ensemble, failure of Kafka broker / Zoo-keeper does not affect the state of the Kafka cluster. Kafka will restore the state, once the Zoo-keeper restarts. This gives zero downtime for Kafka. The leader election between the Kafka broker is also done by using Zoo-keeper in the event of leader failure.

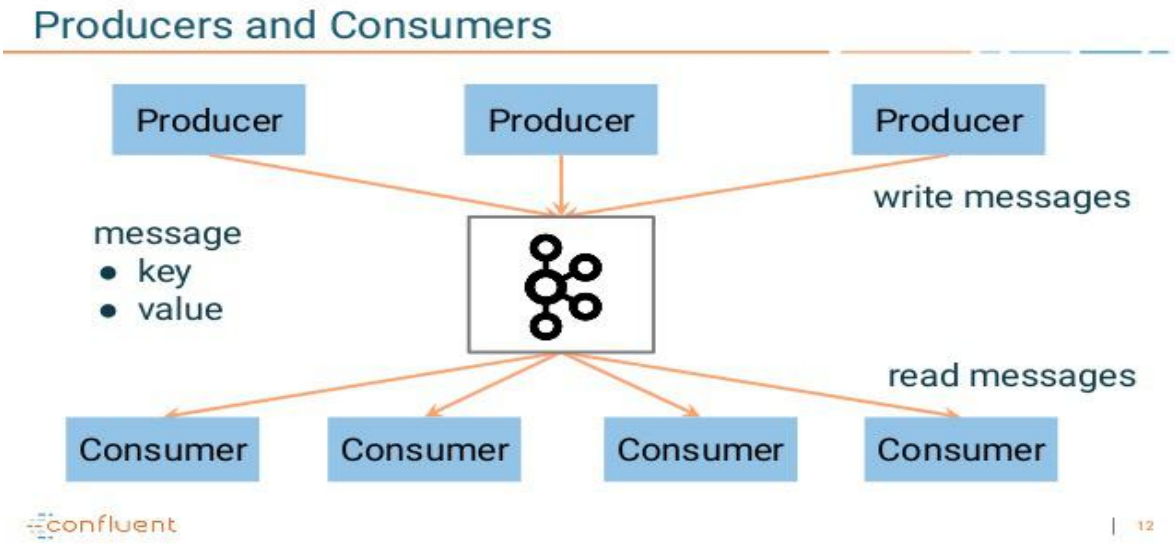


Fig 3.3 Kafka Message Broker

3.4 Node Js:

What is Node Js ?

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36.

Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

I/O bound Applications

•Data Streaming Applications

Data Intensive Real-time Applications (DIRT)

JSON APIs based Applications

Single Page Applications

Why is NodeJs used in IOT ?

Devices like sensors, beacons, transmitters, motors have a tendency of generating a large volume of data thereby generating a large number of request, Node.Js is well equipped to handle this request through streams. Streams offer both readable and writable channels which help in piping the request to destination without temporary storing the data. Streams are basically Unix pipe and can directly connect to destination.

The key feature of any IoT driven application is collecting data, communicating, analyzing and acting. Node js makes a perfect partner for all above features.

Sockets and MQTT protocol are well suited in Node js which are generally used for continuous data transmission in IoT application.

IoT application cable boards, such as Intel Edison, BeagleBone Black, and Raspberry Pi, can easily install Node js as a programming environment. Node js comes with NPM packages manager which contains many useful IoT modules, which can be used for rapid and robust application development.

Node js is known for its speed, scalability, and efficiency making it the key player for data-intensive real time application. This makes Node js well suited for IoT which relies on data intensive real-time traffic.
IoT devices command are generally written in low-level languages like C and C++ which itself are difficult to learn, Node js comes with the power of JavaScript which is pretty easy to learn and understand.

Node js open source community NPM (Node package manager) contains more than 80 for Arduino controllers, raspberry pi, Intel IoT Edison. It contains more than 30 packages for different sensors and Bluetooth devices. These modules make application development fast and easy.

3.5 Angular Js

What is Angular Js ?

AngularJS is a client side JavaScript MVC framework to develop a dynamic web application. AngularJS was originally started as a project in Google but now, it is open source framework.

AngularJS is entirely based on HTML and JavaScript, so there is no need to learn another syntax or language.

AngularJS changes static HTML to dynamic HTML. It extends the ability of HTML by adding built-in attributes and components and also provides an ability to create custom attributes using simple JavaScript.

Why is Angular Js used ?

AngularJS is the widely used and has become developers favorite JavaScript Framework for Web Applications Development. This has happened because the AngularJS has given developers the enormous flexibility to develop interactive, user-friendly and platform-independent web applications. The whole development process can be completed in less time and with easy going mode without any hurdles. The whole application is developed and deployed within time and budget. AngularJS is the best choice for developing web applications for the following features.

a) MVC Framework

AngularJS provides with a smooth Model View Control Architecture which is also very dynamic in nature.

As we know any application is build up from combining different modules together. These modules work with different logics. These are initialized differently from each other. But still, these modules are connected with each other by some logic. The developers have to build all the components separately and then have to combine them together with some code and applied logic to fix them in a single application. This, of course, is an overhead for the developers while using an MVC Framework.

MVC makes it easier for developers to build client side web application. All the MVC elements which are developed separately are combined automatically using AngularJS Framework. There is no need for developers to write extra code to fit all the elements together. It allows you to put MVC element separately and automatically sets them together accordingly.

b) User Interface With HTML

AngularJS uses the HTML (Hyper Text Markup Language) to build the user interface. The HTML language is a declarative language which uses very short tags and it is very simple to understand. It gives better simple and organized user interface. JavaScript Interfaces are more complicated to reorganize and to develop. HTML are provided with the special attributes. The controller for every element is made easy to define using these attributes.

c) POJO Model

Plain Old JavaScript Objects are very self-sufficient in terms of functionality. The earlier data models use to keep monitoring the data flow. A POJO data model gives the spontaneous and very well planned objects and logics. AngularJS Developers have to just create loops of objects and array with the required properties and have to play around it for the required results. It gives us the spontaneous and clear code which helps in highly user-friendly and interactive Web-based apps using AngularJS.

d) Less Code To Write

AngularJS development requires very less coding as there is no need to write the separate code for the MVC. HTML is used for building the user interface.HTML is very simple and has very short and simple tags and attributes for the elements. The data binding has reduced the work for moving the data manually in the view. The app code and the directives have their own different code, so these two can be written simultaneously by two different teams.

No Need to write the different code set for the applications according to the devices to be used for running the application. A single code set for one application makes your task done. One application can be used over different platforms.

e) Filters

Filters work is to just filter the data before it reaches the view. It performs the paginations and also filters a data array with respect to the parameters. The functions are modified according to the ideal parameters but these are only data transformation task done. It works with regards to putting the data in the proper format before it is actually placed in the view. For example putting decimal point in a number or reversing the order of numbers/serial in the desired range

f) Unit Testing

Earlier the testing used to be performed by creating a test page and then invoking it to test the behavior of the component. With the use of AngularJS the testing part has become simpler. The Application uses Dependency injection to bind the entire application together. It helps to function and manage the scope and the control with very much ease. All the controllers are dependent on the dependency injection to pass the data, here the Dependency injection is used to insert the mock or we can say testing data inside the controller to find out the behavior and what output is generated.

g) DOM Manipulation

The view modifies and manipulates the DOM to update the data and the behavior. But with the use of AngularJS development, the DOM manipulation is the task of the directives and not the view. The DOM manipulations are taken out that has made the designers and the users to focus more on the view. The MVC functions purely on the view and the data flow over it. There is no need to worry about the DOM manipulation.

Chapter 4

Smart Lighting system Using IOT

4.1 Introduction:

Automated street lights are necessary while we are trying to survive in the era of smart world. As automation provides perfection and efficiency. In this paper we are focusing on automated street lighting, as current system is facing many problems. Here we are considering the problems which are done manually. A user has to deal with numerous problem like maintenance problem, timer problem, connectivity problem, display problem.

4.1.1 Motivation:

Street lights are one of the main city's assets which provides safe roads, inviting public areas, and enhanced security in homes, businesses, and city centers. As they use in average 40% of a city's electricity spending which leads to power consumption. Following are the issues of existing electric system.

Connectivity issue-In existing system, connections of street light are done manually. As each connection requires different contractors and if any one of them is not available then it will leads to functionality problem of street lights.

Timer Problem-Contractors needs to manage timer settings manually. As timer requires twelve hour of continuous electricity supply, and if in case it is not available, it will delay further timer settings.

Maintenance problem-If any of the street light gets failed or any problem occurs, it's not resolved immediately.

Incorrect Readings-Sometimes exact readings are not shown on to the display. So we can not conclude how much energy is being consumed which give rise in high billing.

Street lights are among a city's strategic assets providing safe roads, inviting public areas, and enhanced security in homes, businesses, and city centers. However they are usually very costly to operate, and they use in average 40% of a city's electricity spending. As the cost of electricity continues to rise and as wasting energy is a growing concern for public and authorities, it's becoming crucial that municipalities, highway companies and other streetlight owners deploy control systems to dim the lights at the right light level at the right time, to automatically identify lamp and electrical failures and enable real time control.

Street Light Monitoring & control is an automated system designed to increase the efficiency and accuracy of an industry by automatically timed controlled switching of street lights. This project describes a new economical solution of street light control systems. The control system consists of wireless technology. Base server can control the whole city's street lights by just sending a notification using network. The main motive behind implementing this project to save energy.

4.2 Problem Statement:

Smart lighting system considering the intensity of sunlight.

4.3 Data Flow Diagram:

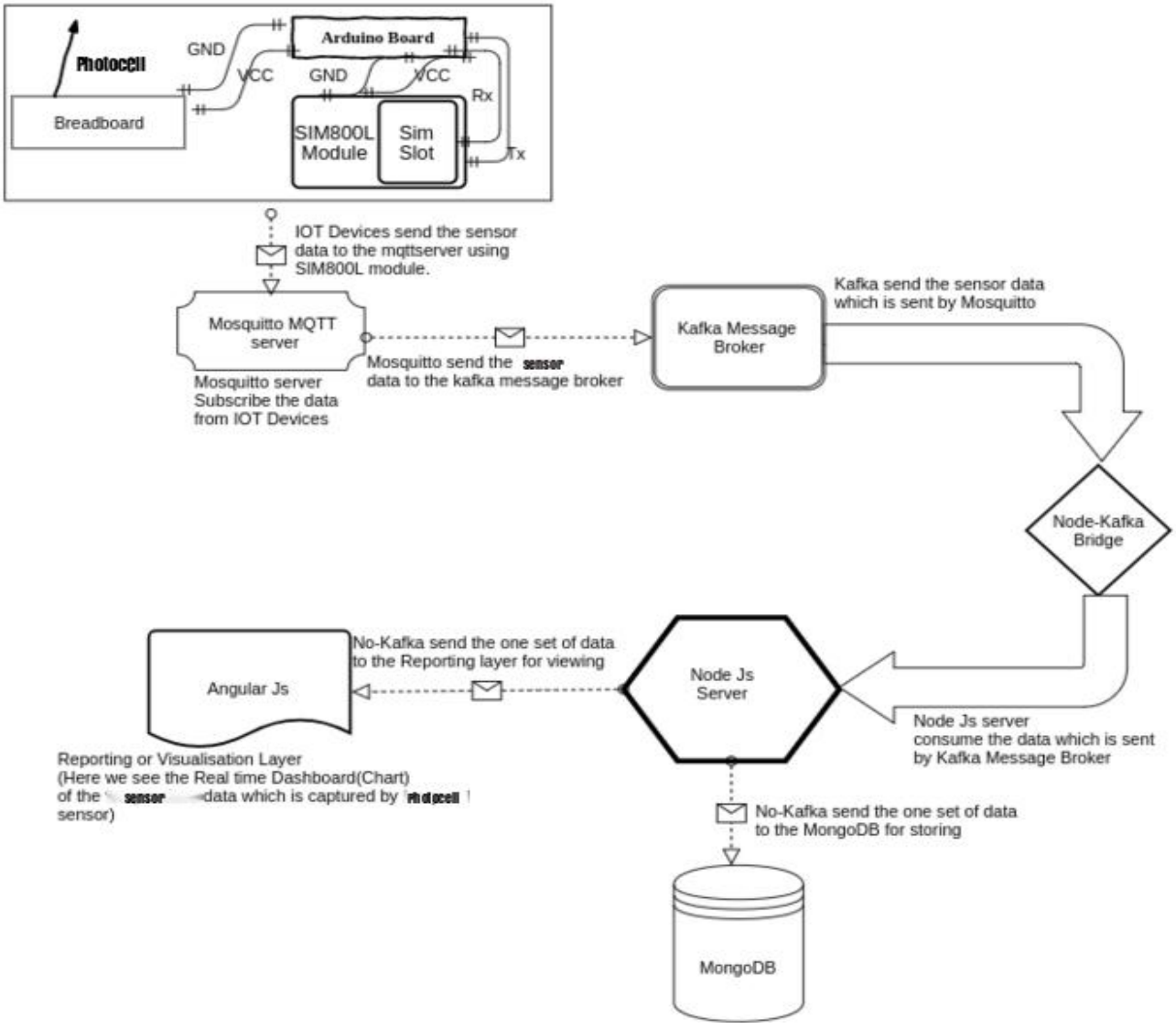


Fig 4.3 DFD of Smart Lighting

In this project, we made IOT devices using Arduino Mega 2560, Breadboard, Resistors, Led, M-M Jumper wire, Sim800l module, Photocell / Photoresistor and 74HC595 IC.

This Device send the sensor data to the mosquitto mqtt server using SIM800L module. Mosquitto mqtt server subscribe this data inside a topic and send this topic (data) to the kafka message broker. Kafka send this sensor data inside a topic via No-Kafka (Node-Kafka).Then Node js consume this topic (data) from the bridge. After processing node js create Two sets , One set of data inside a topic send to the NoSql database, MongoDB directly and other set of data inside the topic send to the Reporting or Visualization layer, In this layer, we create a real-time dashboard of the real time sensor data, Which can we view in the web. We created this real-chart using Angular Js.

4.4 IOT Devices:

Here we measure light intensity using an Analog Input. we use the level of light to control the number of LEDs to be lit.

Here we use 74HC595 Shift Register, This shift register is a type of chip that holds what can be thought of as eight memory locations, each of which can either be a 1 or a 0. To set each of these values on or off, we feed in the data using the 'Data' and 'Clock' pins of the chip.

The clock pin needs to receive eight pulses. At each pulse, if the data pin is high, then a 1 gets pushed into the shift register; otherwise, a 0. When all eight pulses have been received, enabling the 'Latch' pin copies those eight values to the latch register. This is necessary; otherwise, the wrong LEDs would flicker as the data is being loaded into the shift register. The chip also has an output enable (OE) pin, which is used to enable or disable the outputs all at once. You could attach this to a PWM-capable MEGA2560 pin and use 'analogWrite' to control the brightness of the LEDs. This pin is active low, so we tie it to GND.

We also use the light intensity sensor photocell. The photocell is at the bottom of the breadboard, where the pot was above.

The photocell used is of a type called a light dependent resistor, sometimes called an LDR. As the name suggests, these components act just like a resistor, except that the resistance changes in

response to how much light is falling on them. This one has a resistance of about 50 kΩ in near darkness and 500 Ω in bright light. To convert this varying value of resistance into something we can measure on an MEGA2560 R3 board's analog input, it needs to be converted into a voltage. The simplest way to do that is to combine it with a fixed resistor.

Connection

Schematic

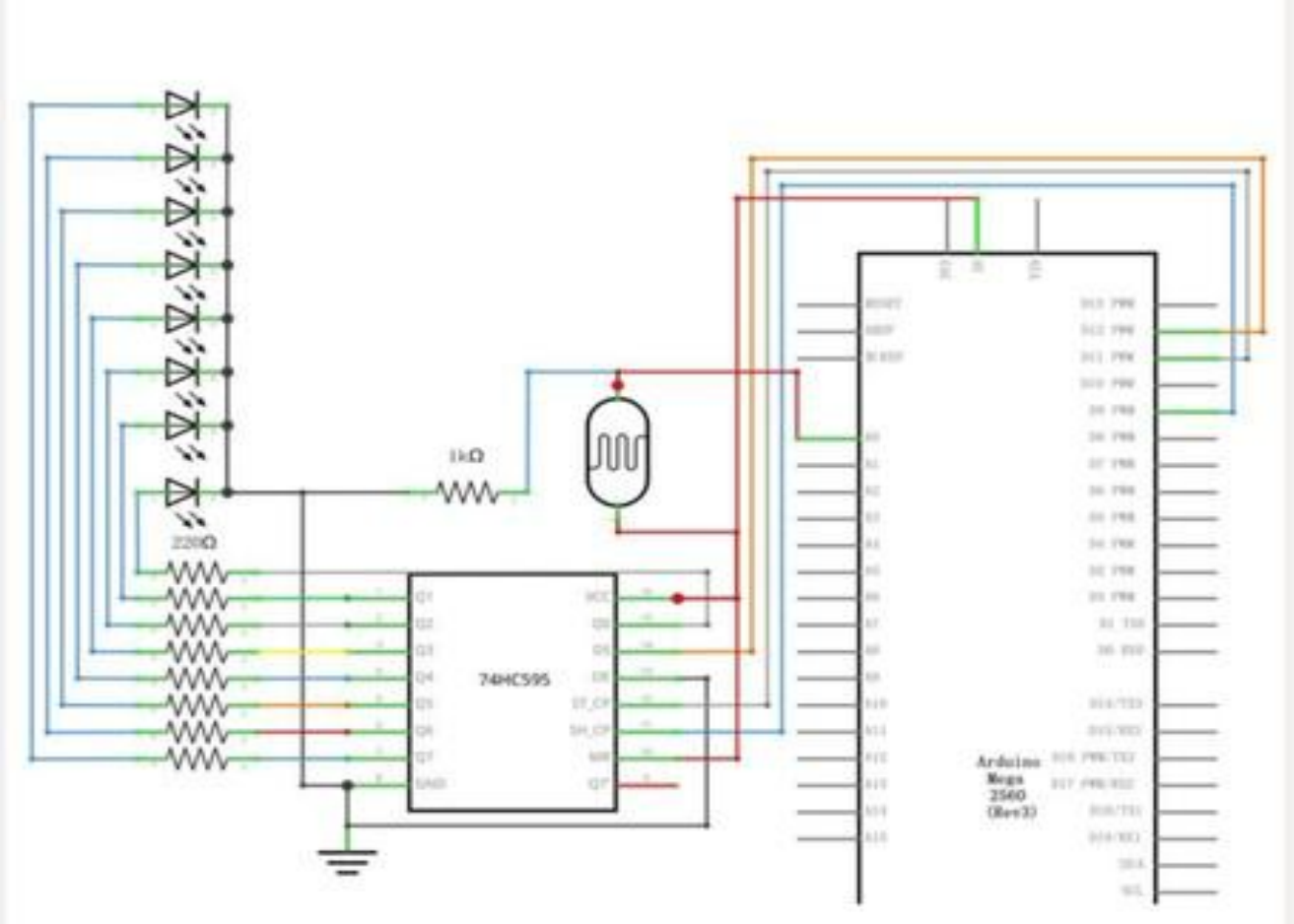


Fig 4.4.1 Schematic Diagram

As we have eight LEDs and eight resistors to connect, there are actually quite a few connections to be made. It is probably easiest to put the 74HC595 chip in first, as pretty much everything else

connects to it. Put it so that the little U-shaped notch is towards the top of the breadboard. Pin 1 of the chip is to the left of this notch.

Digital 12 from the MEGA2560 goes to pin #14 of the shift register

Digital 11 from the MEGA2560 goes to pin #12 of the shift register

Digital 9 from the MEGA2560 goes to pin #11 of the shift register

All but one of the outputs from the IC is on the left side of the chip. Hence, for ease of connection, that is where the LEDs are, too. After the chip, put the resistors in place. You need to be careful that none of the leads of the resistors are touching each other. You should check this again before you connect the power to your MEGA2560. If you find it difficult to arrange the resistors without their leads touching, then it helps to shorten the leads so that they are lying closer to the surface of the breadboard. Next, place the LEDs on the breadboard. The longer positive LED leads must all be towards the chip, whichever side of the breadboard they are on. Attach the jumper leads as shown above. Do not forget the one that goes from pin 8 of the IC to the GND column of the breadboard. Load up the sketch listed a bit later and try it out. Each LED should light in turn until all the LEDs are on, and then they all go off and the cycle repeats.

The resistor and photocell together behave like a pot. When the light is very bright, then the resistance of the photocell is very low compared with the fixed value resistor, and so it is as if the pot were turned to maximum. When the photocell is in dull light, the resistance becomes greater than the fixed 1 kΩ resistor and it is as if the pot were being turned towards GND. Load up the sketch given in the next section and try covering the photocell with your finger, and then holding it near a light source.

Here we use Sim800l module to transfer the sensor data from arduino mega 2560 to mosquitto mqtt server via Rx and Tx pin.

Sim800l module GND pin is connected with Mega 2560 GND.

Sim800l module VCC pin is connected with Mega 2560 5V.

Sim800l module Tx pin is connected with Mega 2560 Rx0 port.

Sim800l module Rx pin is connected with Mega 2560 Tx0 port.

Wiring diagram:

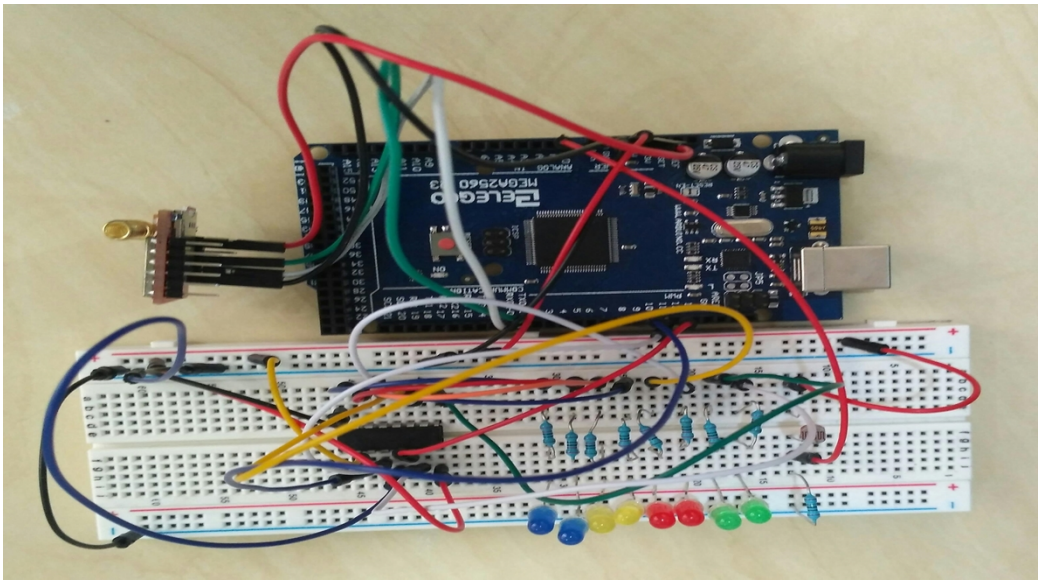


Fig 4.4.2 Wiring Diagram

4.4.1 Arduino Code:

In this project, the number of led lit on depending on the calculation of intensity of light. The code is given below:

```
File Edit Sketch Tools Help

ra

//SendSubscribePacket();
while (1)
{
  int i;
  int reading1  = analogRead(lightPin);
  int reading = reading1 - 80;
  int numLEDSLit = reading / 30; //1023 / 9 /2
  if (numLEDSLit > 8) numLEDSLit = 9;
  leds = 0;   // no LEDs lit to start
  for (i = 0; i < (9 - numLEDSLit); i++)
  {
    leds = leds + (1 << i);  // sets the i'th bit
  }
  updateShiftRegister();
  // Serial.println(reading);
  char buf[20];
  strcpy_P(temperature, (const char*) F("Light Intensity ="));
  dtostrf(reading1, 6, 0, buf);// float to character array
  strcat(temperature, buf);
  strcat_P(temperature, (const char*) F(": Number of LED ="));
  char buf1[20];
  dtostrf(i, 3, 0, buf1);
  strcat(temperature, buf1);
  strcat_P(temperature, (const char*) F(":"));
  SendPublishPacketTemp();

Done Saving.
```

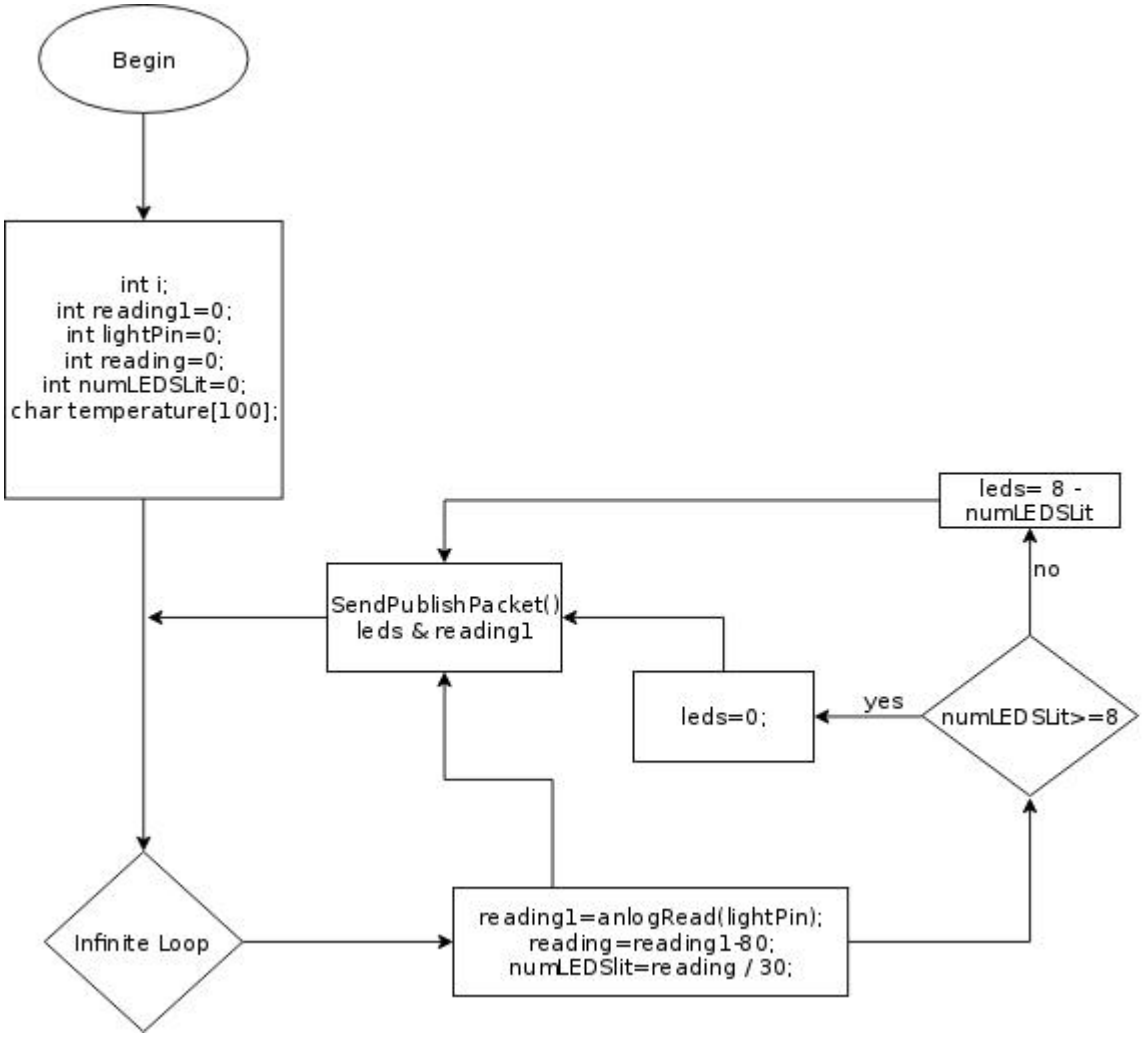


Fig 4.4.3 Flow-chart of the used code

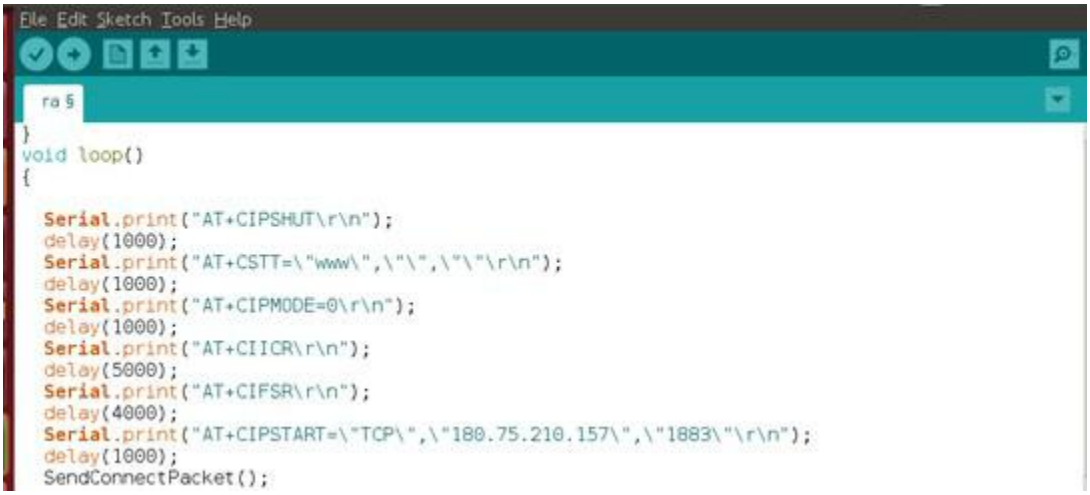
4.5 Usage of MQTT Server:

MQTT (Message Queuing Telemetry Transport) is a publish/subscribe messaging protocol for constrained Internet of Things devices and low-bandwidth, high-latency or unreliable networks. Because MQTT specializes in low-bandwidth, high-latency environments, it is an ideal protocol for machine-to-machine (M2M) communication.

At the heart of using MQTT as a communication avenue is the topic. It's a remarkably simple idea not unique to MQTT; however, the MQTT protocol leverages the power of this quite nicely. A topic implicitly accomplishes several tasks, most importantly ensuring that a message is

delivered to the correct listeners. MQTT treats a topic as a file path. When thinking of a topic as a simple communication filter, the path application can become very powerful.

In this project, We used Mosquitto mqtt server. Arduino send the sensor data to the mosquitto mqtt inside a topic, named “testing”. Then mosquitto send this data inside the same topic to the Apache Kafka. We sent the data to the mosquitto by using sim800l module. The ip address of the mqtt server is 180.75.210.157 (for example) and the port is 1883 (for example).



Here we use “AT Command” to continue the Sim800l service. The description of the “AT Command” is given below:

AT command:

What is AT command ?

AT commands are instructions used to control a modem. AT is the abbreviation of ATtention. Every command line starts with "AT" or "at". That's why modem commands are called AT commands.

The following AT commands is used our project.

AT+CIPSHUT

This commannd is used to shut down the all GPRS connection. If sending is successful,it will respond “SHUT OK”.

AT+CSTT

This command is used to start task and set APN user name and password. If sending is successful,it will respond “OK”.

AT+CIPMODE=0

Command AT+CIPMODE=<n> is used for selsecting TCPIP application mode, when n=0,it is non-transparent mode(normal mode); when n=1, it is transparent mode. The default configuration is non transparent mode. If sending is successful,it will respond “OK”.

AT+CIICR

This command is used for bring up wireless connection(GPRS or CSD). If sending is successful,it will respond “OK”.

AT+CIFSR

This command is used to get local IP address. If sending is successful,it will send 10.11.172.45(depends on local IP address).

AT+CIPSTART="TCP","180.75.210.157","1883"

This command is used to start up TCP connection to remote server. If sending is successful,it will respond “OK”, then “CONNECT”.

AT+CIPSEND

This command is used to send data to remote server. If sending is successful,it will respond “SEND OK”

Data has been sent out from the serial port,but it is unknown if the data reaches the UDP server. For UDP,”SEND OK” just means data has been sent out from the serial port of module ,not meaning data reaching the server, due to the UDP simpler message-based connectionless protocol.

```
root@core-MS-7267:/home/core# mosquitto_sub -h [redacted] -t testing
Light Intensity =   192: Number of LED =  6:
Light Intensity =   159: Number of LED =  7:
Light Intensity =   175: Number of LED =  6:
Light Intensity =   159: Number of LED =  7:
Light Intensity =   179: Number of LED =  6:
Light Intensity =   163: Number of LED =  7:
Light Intensity =   179: Number of LED =  6:
Light Intensity =   163: Number of LED =  7:
Light Intensity =   168: Number of LED =  7:
Light Intensity =   165: Number of LED =  7:
Light Intensity =   165: Number of LED =  7:
Light Intensity =   158: Number of LED =  7:
Light Intensity =   132: Number of LED =  8:
Light Intensity =   156: Number of LED =  7:
Light Intensity =   144: Number of LED =  7:
Light Intensity =   152: Number of LED =  7:
Light Intensity =   273: Number of LED =  3:
Light Intensity =   205: Number of LED =  5:
Light Intensity =   122: Number of LED =  8:
Light Intensity =   745: Number of LED =  0:
Light Intensity =    29: Number of LED =  8:
```

Fig 4.5 Mosquitto Subscribe the data inside the topic 'testing'

4.6 MQTT - Kafka Bridge:

This is not an official connector from Apache Kafka, instead it comes from the community. It allows us to move data from the MQTT broker into Apache Kafka.

We can send the data inside the different topic from Mqtt to the Single topic of Kafka. But We can't send the data inside the different topic from mqtt to the Different topic of Kafka.

In this project, we created a topic called “testing” in the mosquitto mqtt broker and sane topic created in the Apache Kafka. We sent the sensor data to the mosquitto mqtt and mosquitto mqtt send the same data in the same topic named “testing”. We created the bridge between Mosquitto Mqtt and Apache Kafka in the following way.

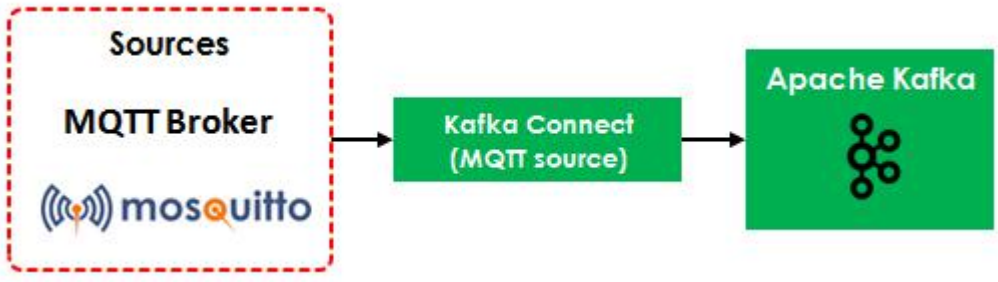


Fig 4.6.1 Apache Kafka Connect MQTT Source

```
name=mqtt
connector.class=com.evokly.kafka.connect.mqtt.MqttSourceConnector
tasks.max=1

kafka.topic=testing
mqtt.client_id=12345

mqtt.clean_session=true
mqtt.connection_timeout=30
mqtt.keep_alive_interval=60

mqtt.server_uris=tcp://localhost:1883
mqtt.topic=testing
~
~
```

Fig 4.6.2 Mqtt - Kafka Bridge

4.7 Usage of Apache Kafka:

Kafka works well as a replacement for a more traditional message broker. Message brokers are used for a variety of reasons (to decouple processing from data producers, to buffer unprocessed messages, etc). In comparison to most messaging systems Kafka has better throughput, built-in partitioning, replication, and fault-tolerance which makes it a good solution for large scale message processing applications.

In our experience messaging uses are often comparatively low-throughput, but may require low end-to-end latency and often depend on the strong durability guarantees Kafka provides.

In this project, we used apache kafka as messaging queue or message broker which can fetch data inside the topic “testing” from the mosquitto mqtt server and send this data in the node js server via no-kafka bridge in real-time.

```
root@core-MS-7267:/home/core# mosquitto_sub -h [redacted] -t testing
Light Intensity =   211: Number of LED =  5:
Light Intensity =   155: Number of LED =  7:
Light Intensity =   211: Number of LED =  5:
Light Intensity =   156: Number of LED =  7:
Light Intensity =   210: Number of LED =  5:
Light Intensity =   157: Number of LED =  7:
Light Intensity =   208: Number of LED =  5:
Light Intensity =   156: Number of LED =  7:
Light Intensity =   209: Number of LED =  5:
Light Intensity =   155: Number of LED =  7:
Light Intensity =   211: Number of LED =  5:
Light Intensity =   155: Number of LED =  7:
Light Intensity =   209: Number of LED =  5:
Light Intensity =   157: Number of LED =  7:
Light Intensity =   209: Number of LED =  5:
Light Intensity =   155: Number of LED =  7:
Light Intensity =   211: Number of LED =  5:
Light Intensity =   159: Number of LED =  7:
Light Intensity =   229: Number of LED =  5:
Light Intensity =   174: Number of LED =  6:
Light Intensity =   200: Number of LED =  5:
```

Fig 4.7.1 Mosquitto Subscribe the data inside the topic 'testing'

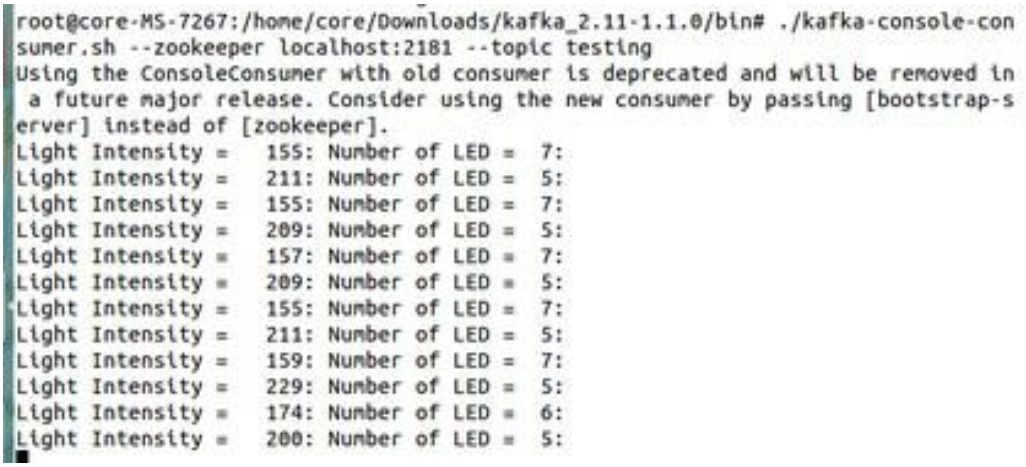


Fig 4.7.2 Kafka Fetch the same data from Mosquitto mqtt server inside the topic testing

4.8 Usage of Node Js:

Devices like sensors, beacons, transmitters, motors have a tendency of generating a large volume of data thereby generating a large number of request, Node.js is well equipped to handle this request through streams. Streams offer both readable and writable channels which help in piping the request to destination without temporary storing the data. Streams are basically Unix pipe and can directly connect to destination.

The key feature of any IoT driven application is collecting data, communicating, analyzing and acting. Node js makes a perfect partner for all above features.

Sockets and MQTT protocol are well suited in Node js which are generally used for continuous data transmission in IoT application.

IoT-application cable boards, such as Intel Edison, BeagleBone Black, and Raspberry Pi, can easily install Node js as a programming environment. Node js comes with NPM packages manager which contains many useful IoT modules, which can be used for rapid and robust application development.

Node js is known for its speed, scalability, and efficiency making it the key player for data-intensive real time application. This makes Node js well suited for IoT which relies on data intensive real-time traffic.

IoT devices command are generally written in low-level languages like C and C++ which itself are difficult to learn, Node js comes with the power of JavaScript which is pretty easy to learn and understand.

Node js open source community NPM (Node package manager) contains more than 80 for Arduino controllers, raspberry pi, Intel IoT Edison. It contains more than 30 packages for different sensors and Bluetooth devices. These modules make application development fast and easy.

In this Project, we used Node Js as server side application which takes the data in 9092 port where apache kafka running and send this data through the port 8091 to the Angular js for reporting layer or Visualization layer.

server.js ×

```
'use strict';
let app = require('express')();
let http = require('http').Server(app);
let io = require('socket.io')(http);
var Kafka = require('no-kafka');
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://1[illegible]:27017/";


io.on('connection', (socket) => {
  console.log('USER CONNECTED');

  socket.on('disconnect', function(){
    console.log('USER DISCONNECTED');
  });

});
```

Fig 4.8.1 Modules use in Node js

let app=require('express')();

Basically when we do require('express') it imports a function. The following ()calls the function as well. Basically it instantiates an express app.

let http=require('http').Server(app);

It creates a http.Server object with all of its fields and functions.

```
let io=require('socket.io')(http);
io.on('connection',function(socket){
console.log("user connected");
}
```

We initialize a new instance of socket.io by passing the http (the HTTP server) object. Then we listen on the connection event for incoming sockets, and we log it to the console.

```
Var kafka = require('no-kafka');
```

The above code will connect to kafka in our application.

```
var Mongoclient=require('mongodb').MongoClient;
```

Node.js can use this module to manipulate MongoDB database.

```
Var url=mongodb://180.75.210.157:27017/
```

To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database we want to create. MongoDB will create the database if it does not exist, and make a connection to it.

```
http.listen(8091, () => {
  console.log('started on port 8091');
  var consumer = new Kafka.SimpleConsumer({
        connectionString: '              :9092',
        clientId: 'no-kafka-client'
    });
```

Fig 4.8.2 Usable Port

Http.listen(8091.....)

We create a server that listens on port 8091.

Why use SimpleConsumer()?

The main reason to use a SimpleConsumer implementation is we want greater control over partition consumption than Consumer Groups give us.

The SimpleConsumer does require a significant amount of work not needed in the Consumer Groups:

1.We must keep track of the offsets in your application to know where you left off consuming.

2.We must figure out which Broker is the lead Broker for a topic and partition

3.We must handle Broker leader changes.

```
// data handler function can return a Promise
        var dataHandler = function (messageSet, topic, partition) {
                messageSet.forEach(function (m) {
                        console.log(topic, partition, m.offset, m.message.value.toString('utf8'));
                        var string= m.message.value.toString('utf8');
                        console.log(string);
                        var fields = string.split(':');
                        var temperature = fields[0];
                        var Counter = fields[1];
                        if(topic=="testing")
                        {
                                var temp=Counter.split('=');
                                io.emit('message', {x:(new Date()).getTime(), y: temp[1]});
                        }

                        /*else
                        {
                                io.emit('sampleMessage', {x:(new Date()).getTime(), y:
m.message.value.toString('utf8')});
                        }
                */
```

Fig 4.8.3 Data handler function

var fields = string.split(':');

The above line is used to break the data which is comeing continuously from topic “testing” when ':' operator is found on that dataand that data will be stored in an array named 'fields' here.

var temperature= fields[0];

var Counter = fields[1];

The data which is stored in the first index of fields,will be copied in “temperature” variable. The Light Intensity data will be stored in temperature variable in our project. Similarly in 'Counter' variable,How many led lit on that moment will be stored.

if(topic==”testing”)

{

var temp=Counter.split(“=”);

io.emit('message',{x:(new Date()).getTime(),

y:temp[1]}); }

Here, if the data comes from testing topic then the data will split and will be stored in temp array when "=" operator is found.Then io.emit sends the data to angularJS as named 'message'.the data are the current time and the number of led lit on at that moment.

```
return consumer.init().then(function () {
        // Subscribe partitons 0 and 1 in a topic:
        var v1= consumer.subscribe('testing', [0, 1], dataHandler);
        var v2= consumer.subscribe('test', [0, 1], dataHandler);
        var arr=[];
        arr.push([v1,v2]);
        console.log("val:"+arr);
        return arr;

});
);
```

Fig 4.8.4 Consumed topics

In the above code, nodeJs subscribe the partions 0 ad 1 of the two topics "testing" and "test".It is an initialization part of NodeJS.

4.9 Usage of NoSql Database (MongoDB):

IoT applications process great volumes of data through sensors so your system will need to scale quickly and cheaply. One of the advantages of MongoDB is the ability to scale out on inexpensive commodity hardware in your data center or in the cloud. Analyze any kind of data with MongoDB.

In this project we use Mongo db for storing real time sensor data.

```
MongoClient.connect(url, function(err, db) {
if (err) throw err;
var dbo = db.db("mydb5");
var myobj = { temperature: m.message.value.toString('utf8'), date: (new Date())};
dbo.collection("project").insertOne(myobj, function(err, res) {
if (err) throw err;
console.log("1 document inserted");
db.close();
});
});

//
});
};
```

Fig 4.9.1 MongoDb in Node js

```
var dbo = db.db("mydb5");
```

The above code is used to create a database named "mydb5" and also created an object named "dbo" for the database.

Var myobj={ temperature: m.message.value.toString('utf8'), date: (new Date())}

We created an object named 'myobj' in which the data is storded in key value pair.

Dbo.collectiion(“project”).insertone(myobj, function(err , res){}

To insert the myobj document into our database we use insertone() method.The first parameter of that method is an object containing the name(s) and value(s) of each field in the document we want to insert.

It also takes a callback function where you can work with any errors, or the result of the insertion.

Here we insert the document in a collection, named “project”.

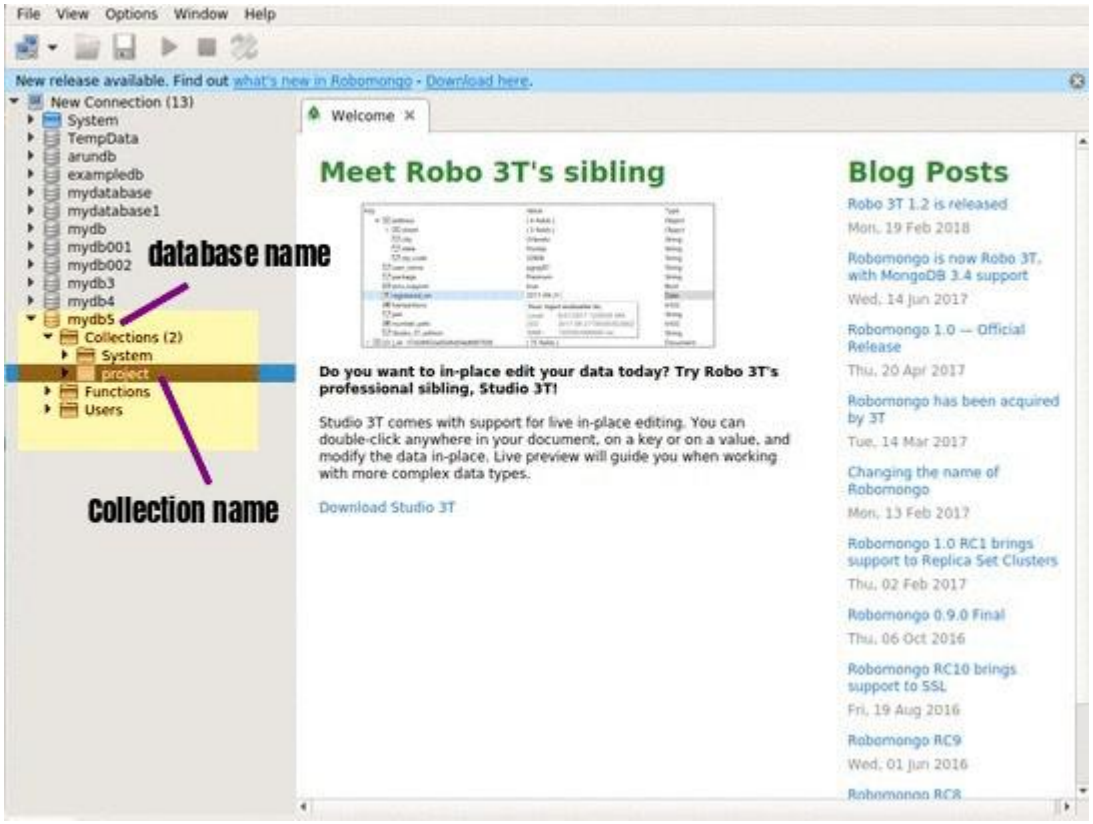


Fig 4.9.2 Database and Collection in Robo MongoDb

4.10 Usage of Angular Js:

Data visualization can help users see patterns in an often overwhelming amount of data. One key technique for visualizing data is to display that data in a set of charts. You can use Angular-charts to build pie, bar, line, point, and area charts.

In this project we used angularjs 2 for creating the real time sensor data chart. Basically we used angularjs2-highcharts. When Node js send the sensor data to the port 8091, angular js consumed this port to catch this real time sensor data and display the graph in 4200 port.

```
export class ChatService {
  private url = 'http://                :8091';
  private socket;

  sendMessage(message) {
    this.socket.emit('add-message', message);
    console.log("MESSAGE SENT");
  }
```

Fig 4.10.1 Subscribe 8091 port

```
capabilities: {
  'browserName': 'chrone'
},
directConnect: true,
baseUrl: 'http://180.75.210.157:4200/',
framework: 'jasmine',
jasmineNodeOpts: {
  showColors: true,
  defaultTimeoutInterval: 30000,
  print: function() {}
},
```

Fig 4.10.2 Display in 4200 port

```
renderChart() {
  this.options = {

    rangeSelector: {
      buttons: [{
        count: 1,
        type: 'minute',
        text: '1M'
      }, {
        count: 5,
        type: 'minute',
        text: '5M'
      }, {
        type: 'all',
        text: 'All'
      }],
      inputEnabled: false,
      selected: 0
    },

    title: {
      text: 'Live random data'
    },
```

Fig 4.10.3 Render Chart

```
    xAxis: {
      opposite: true
    },
    yAxis: { opposite: true },
    exporting: {
      enabled: false
    },

    series: [{
      name: 'Live data 1',
      data: []
    },
    {
      name: 'Live data 2',
      data: []
    }
    ]

  }
}
chart: Object;

loadChart(chartInstance) {
  this.chart = chartInstance;
}

}
```

Fig 4.10.4 Axis

```
  getLiveData1() {
    let observable = new Observable(observer => {
      this.socket = io(this.url);
      this.socket.on('message', (data) => {

        observer.next(data);
      });
      return () => {
        this.socket.disconnect();
      }
    })
    return observable;
  }
  getLiveData2() {
    let observable = new Observable(observer => {
      this.socket = io(this.url);
      this.socket.on('sampleMessage', (data) => {

        observer.next(data);
      });
      return () => {
        this.socket.disconnect();
      }
    })
    return observable;
  }
}
```

Fig 4.10.5 Capturing code of Live data

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
import { AppComponent } from './app.component';
import { ChartModule } from 'angular2-highcharts/index';
declare var require: any;

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule,
        ChartModule.forRoot(require('highcharts/highstock')),
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Fig 4.10.6 Chart module calling

4.11 Experimental Results:

For our experiment, we have used the hardware which are mentioned in hardware requirement part and programs are written in Arduino programming language. We use MQTT server, Kafka message broker , Node Js, Angular Js and MongoDB.The following screenshots show that the streaming data is flowing through the different layer.

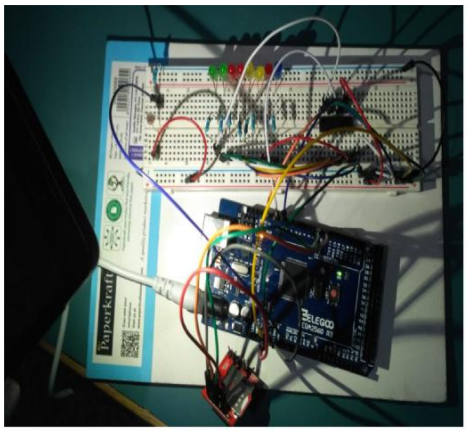


Fig 4.11.1 No Led On

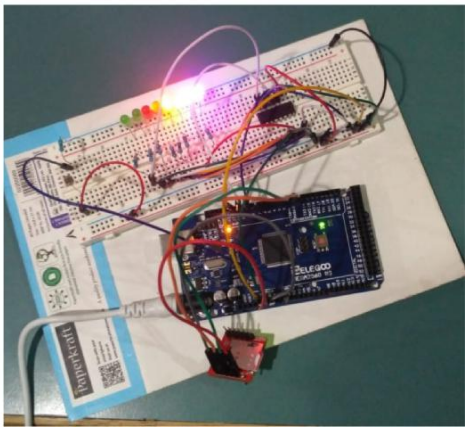
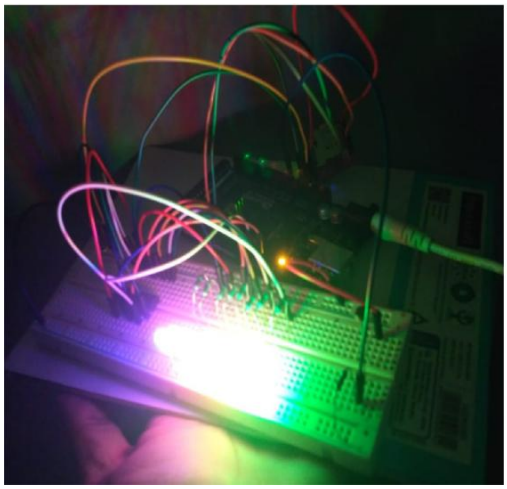


Fig 4.11.2 4 Led On



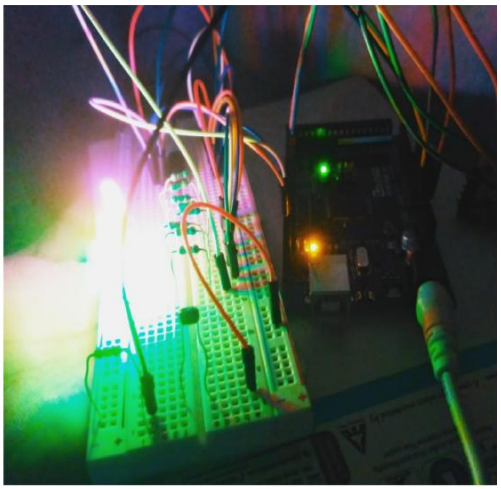


Fig 4.11.3 All Led On

```
2018-05-26T11:44:45.207Z DEBUG no-kafka-client Subscribed to testing:0 offset 54
 leader core-MS-7267:9092
2018-05-26T11:44:45.212Z DEBUG no-kafka-client Subscribed to test:0 offset 0 lea
der core-MS-7267:9092
testing 0 54 Light Intensity =   113: Number of LED =  8:
Light Intensity =   113: Number of LED =  8:
1 document inserted
testing 0 55 Light Intensity =   112: Number of LED =  8:
Light Intensity =   112: Number of LED =  8:
1 document inserted
testing 0 56 Light Intensity =   113: Number of LED =  8:
Light Intensity =   113: Number of LED =  8:
1 document inserted
testing 0 57 Light Intensity =   114: Number of LED =  8:
Light Intensity =   114: Number of LED =  8:
1 document inserted
testing 0 58 Light Intensity =   119: Number of LED =  8:
Light Intensity =   119: Number of LED =  8:
1 document inserted
testing 0 59 Light Intensity =   109: Number of LED =  8:
Light Intensity =   109: Number of LED =  8:
1 document inserted
testing 0 60 Light Intensity =   119: Number of LED =  8:
Light Intensity =   119: Number of LED =  8:
1 document inserted
testing 0 61 Light Intensity =   118: Number of LED =  8:
Light Intensity =   118: Number of LED =  8:
1 document inserted
testing 0 62 Light Intensity =   119: Number of LED =  8:
Light Intensity =   119: Number of LED =  8:
1 document inserted
testing 0 63 Light Intensity =   119: Number of LED =  8:
Light Intensity =   119: Number of LED =  8:
1 document inserted
testing 0 64 Light Intensity =   120: Number of LED =  8:
Light Intensity =   120: Number of LED =  8:
1 document inserted
testing 0 65 Light Intensity =   118: Number of LED =  8:
Light Intensity =   118: Number of LED =  8:
1 document inserted
```

Fig 4.11.4 Real Time data in Node-Kafka

Fig 4.11.5 Real Time data in Live Charts

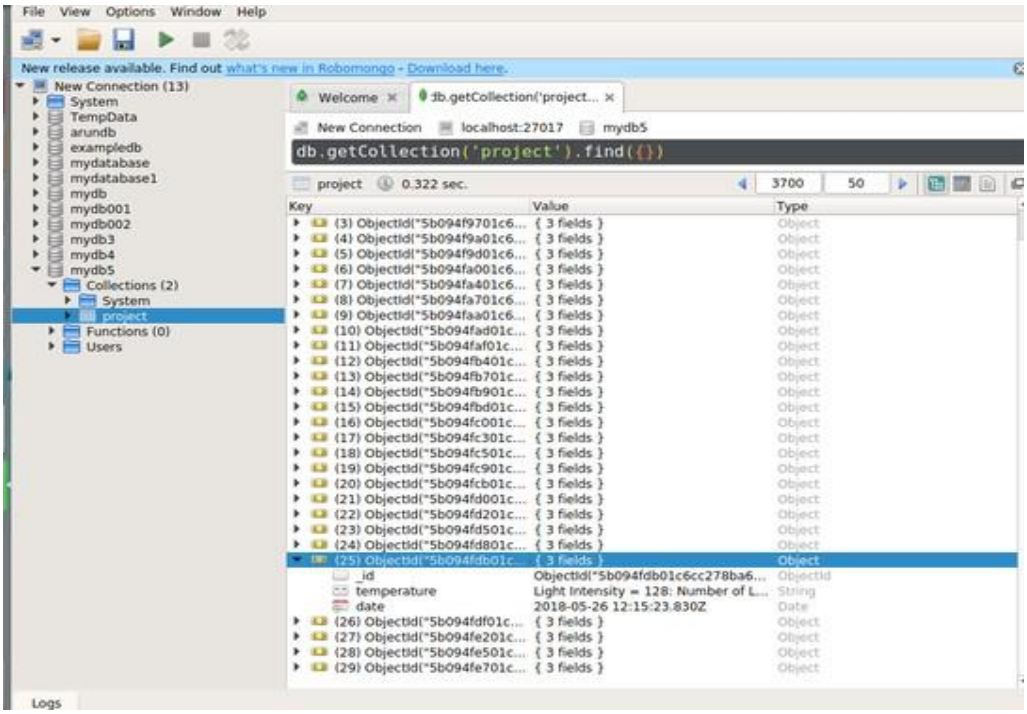


Fig 4.11.6 Real Time Data in MongoDB

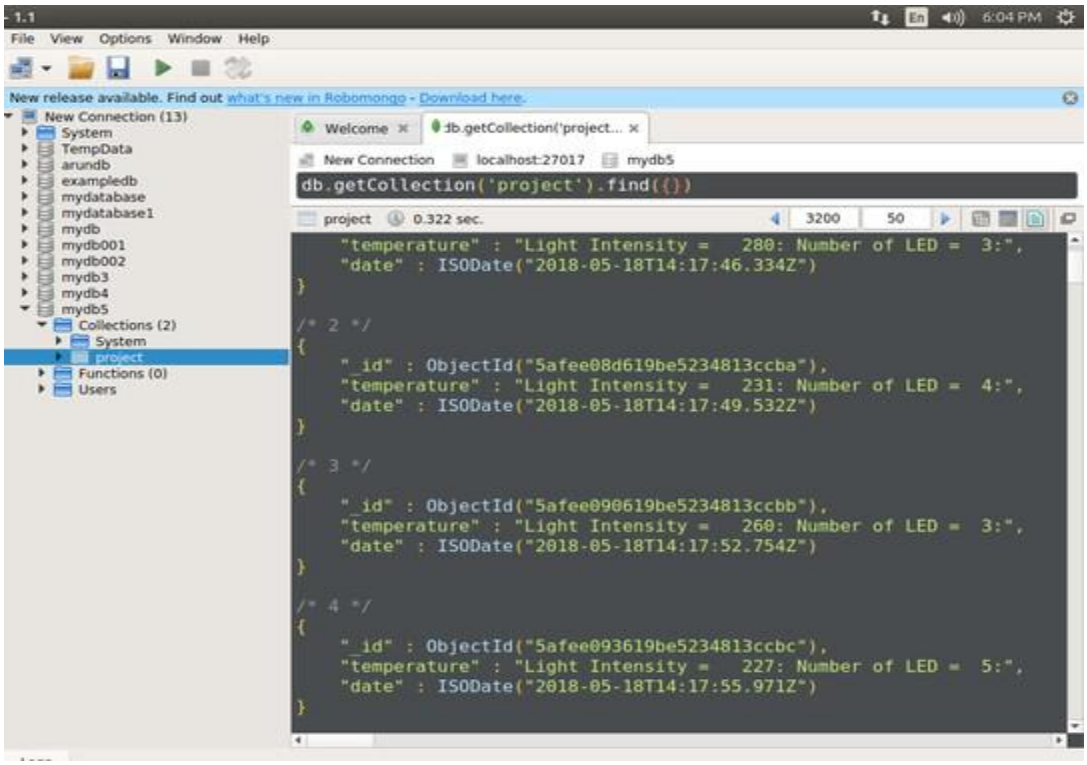


Fig 4.11.7 Real Time Data in MongoDB

4.12 Conclusion:

In this project the smart lighting system describes how new trends in technology can help us in saving the energy. The system works solely in the darkness, avoiding waste of energy throughout sunlight hours. Sensors enable the system to operate solely when necessary. System employs highly economical LEDs to ensure correct illumination and assure energy savings.

Another advantage obtained by the control system is the intelligent management of the lamps by sending data to a central station by wireless communication. The system maintenance can be easily and efficiently planned from the central station, allowing additional savings. Finally we can conclude that we can save 70-75% of energy by using this technology and can increase the life time of leds.

chapter 5

Conclusion and Future Work

5.1 Conclusion:

This project “IoT Based Smart lighting system “ is a cost effective, practical, eco-friendly and the safest way to save energy and this system the light status information can be accessed from anytime and anywhere. It clearly tackles the two problems that world is facing today, saving of energy and also disposal of incandescent lamps, very efficiently. Initial cost and maintenance can be the draw backs of this project.

5.2 Future Work:

There is a lot of scope for future works.

1. Electric energy consumption and measurement:

In future we will try to measure how much electric is used in a lamp per day, month or year as well as we can save energy and increase the lamp life using day light intensity sensor (photocell).

2. Fault Detection:

How many LEDs are working properly and the intensity of the individual LED, we will measure from here as well as we can easily detect which LEDs are going to die and lits off.

3. Controll:

The lights can be controlled from anywhere.

4. Time measurement:

How much time a led is on. That is in a single day or week or month, now much time a light is on.

Bibliography

Prasetyo, William Tandy, Petrus Santoso, and Resmana Lim. "Adaptive Cars Headlamps System with Image Processing and
Lighting Angle Control." Proceedings of Second International Conference on Electrical Systems, Technology and Information 2015 (ICESTI 2015). Springer Singapore, 2016.

Wang Xiao-Yuan, Andrew L. Fitch, Herbert H. C. Iu, Victor Sreeramand Qi Wei-Gui "Implementation of an analogue model of a memristor based on a light-dependent resistor"2012 Chinese Physical Society and IOP Publishing Ltd Chinese Physics B, Volume 21,
Number 10

International Journal of Engineering Research and General Science Volume 4, Issue 2, March-April, 2016
ISSN 2091-2730

Tran, Duong, and Yen Kheng Tan. "Sensorless illumination control of a networked LED-lighting system using feedforward
neural network." Industrial Electronics, IEEE Transactions on 61.4 (2014): 2113-2121.

A.A.Nippun Kumaar, Kiran.G, Sudarshan TSB," Intelligent Lighting System Using Wireless Sensor Networks", International
Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.1, No.4, December 2010, pp 17-27

Y. K. Tan; T. P. Huynh; Z. Wang," Smart Personal Sensor Network Control for Energy Saving in DC Grid Powered LED Lighting
System" , IEEE Trans. Smart Grid

Divya, Guddeti, et al. "Design and Implement of Wireless Sensor Street Light Control and Monitoring Strategy along with GUI."
IJITR (2016): 78-81.

Qi, Liang, MengChu Zhou, and WenJing Luan. "Emergency Traffic-Light Control System Design for Intersections Subject to
Accidents." (2016).

O'Reilly, Fergus, and Joe Buckley. "Use of wireless sensor networks for fluorescent lighting control with daylight substitution."
Proceedings of the Workshop on Real-World Wireless Sensor Networks (REANWSN). 2005.
Tran, Duong, and Yen Kheng Tan. "Sensorless illumination control of a networked LED-lighting system using feedforward
neural network." Industrial Electronics, IEEE Transactions on 61.4 (2014): 2113-2121.
Subramanyam, B. K., K. Bhaskar Reddy, and P. Ajay Kumar Reddy. "Design and development of intelligent Wireless Street light
control and monitoring system along with GUI." International Journal of Engineering Research and Applications (IJERA) Vol 3
(2013): 2115-2119.
Fitch, Andrew Lewis, et al. "Realization of an analog model of memristor based on light dependent resistor." Circuits and
Systems (ISCAS), 2012 IEEE International Symposium on. IEEE, 2012.
Kim, Sangil, et al. "Design of lighting Fluorescent lamp and AC LED control board using MCU." Journal of the The Korean
Institute of Power Electronics (2010): 252-253.
Song, Sang-bin, Woo-young Cheon, and Young- 녀 . Yu. "Design of a LED lighting bar replacement neon sign." Journal of the
Korean Institute of Electrical Engineers (2006): 1671-1672.
Huynh, T. P., Y. K. Tan, and K. J. Tseng. "Energy-aware wireless sensor network with ambient intelligence for smart LED
lighting system control." IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society. IEEE, 2011.
Introduction: https://www.lifewire.com/introduction-to-the-internet-of-things-817766
https://www.cisco.co m/c/dam/en_us/solutions/trends/iot/introduction_to_IoT_november.pdf
[16]Mega 2560: http://www.mantech.co.za/datasheets/products/A000047.pdfsheets/products/A000047.pdf
Breadboard: https://en.wikipedia.org/wiki/Breadboard
https://www.jameco.com/Jameco/Products/ProdDS/2125026.pdf

[18]SIM800L:http://linksprite.com/wiki/index.php5?title=SIM800L_Quad-band_Network_Mini_GPRS_GSM_Breakout_Module

LED:https://info.pcboard.ca/led-specifications/3mm-led-technical-specifications/

IC: https://solarbotics.com/product/74hc595 /

Resistor: https://en.wikipedia.org/wiki/Resisto r

Jumper Wire :https://en.wikipedia.org/wiki/Jump_wire

Adapter: https://en.wikipedia.org/wiki/Adapter

USB: https://en.wikipedia.org/wiki/USB

[25]Arduino: https://learn.sparkfun.com/tutorials/what-is-an-arduino

[26]Sensors:http://www.sensuron.com/industry-news/sensors-in-the-industrial-iot/

[27]Apache kafka:https://en.wikipedia.org/wiki/Apache_Kafka

https://www.confluent.io/what-is-apache-kafka/

https://www.tutorialspoint.com/apache_kafka/apache_kafka_workflow.htm

[28]Operating system:https://en.wikipedia.org/wiki/Operating_system

[29]MQTT:https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport

[30]Node js:http://www.tothenew.com/blog/iot-and-nodejs/

[31]Angular Js:

http://www.tutorialsteacher.com/angularjs/what-is-angularjs

https://www.angularminds.com/blog/article/best-angularjs-features-for-web-application-development.html

Motivation: https://www.ijraset.com/fileserve.php?FID=5943