



Generic IOT Platform V2.0

09.04.2017

Amruta Deshpande
Meghna Bhawe
Pradeep B

FOSSEE Project

INDEX

1. Overview
 - 1.1. Real-time Data Streaming
 - 1.2. Controlling Devices
 - 1.3. Energy Consumption
 - 1.4. Project Creation
2. Installation
 - 2.1. Prerequisites
 - 2.2. Steps to install
3. Goals
 - 3.1. Added functionality in V2.0 over V1.0
 - 3.2. Goals for V3.0
4. Specifications
 - 4.1. Technology Used
 - 4.2. NodeJS
 - 4.3. MongoDB
 - 4.4. Bootstrap
 - 4.5. Canvas Gauges
 - 4.6. JQuery
 - 4.7. Node-RED
 - 4.8. Mosquitto
5. Mongo DB Docs
 - 5.1. Auth Schema
 - 5.2. User Schema
 - 5.3. User Details Schema
 - 5.4. Notifications Schema
 - 5.5. Project Schema
 - 5.6. Sensor Schema
 - 5.7. Sensor Data Schema
6. Usage (Pages Overview)
 - 6.1. Sign Up Page
 - 6.2. Login Page
 - 6.3. Profile Page/Dashboard
 - 6.4. Devices Page
 - 6.4.1. Modal window for adding a device Screenshot
 - 6.5. Node-RED
 - 6.6. Visuals
 - 6.7. Date to Date Analysis
7. MQTT Communication
8. Encryption/ Decryption

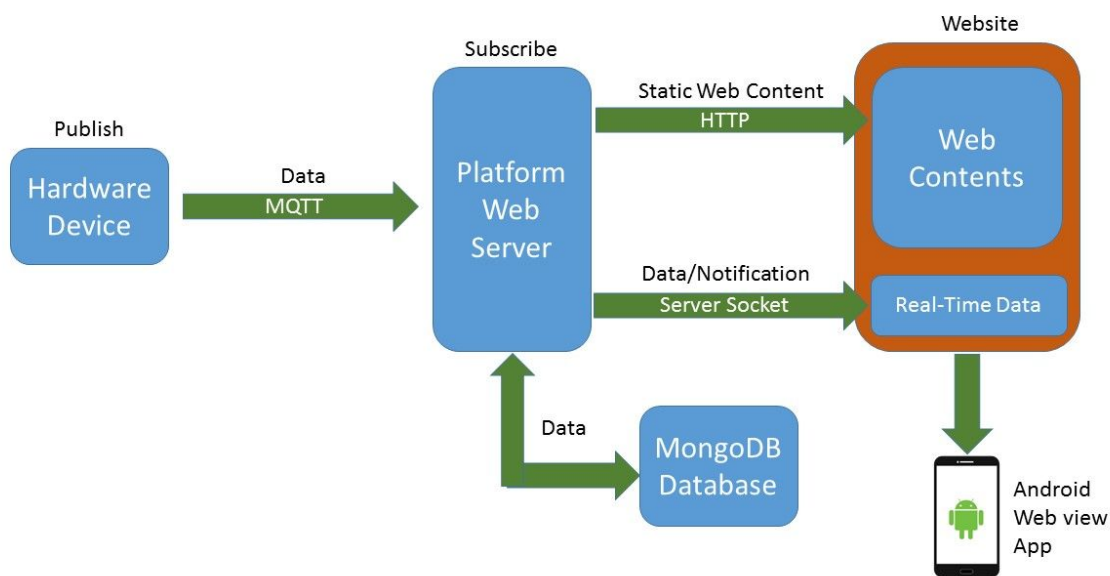
9. Android Mobile Application
 - 9.1. Technologies Used
 - 9.2. Code outline

Overview

The project aims at providing an **open source web and mobile platform** for all IoT based data analysis. The platform allows users to send data to the platform for **real time data visualisation**. Users can also **control the devices** through the platform. **Power consumption** of every device can also be viewed through this platform.

Real-time Data Streaming:

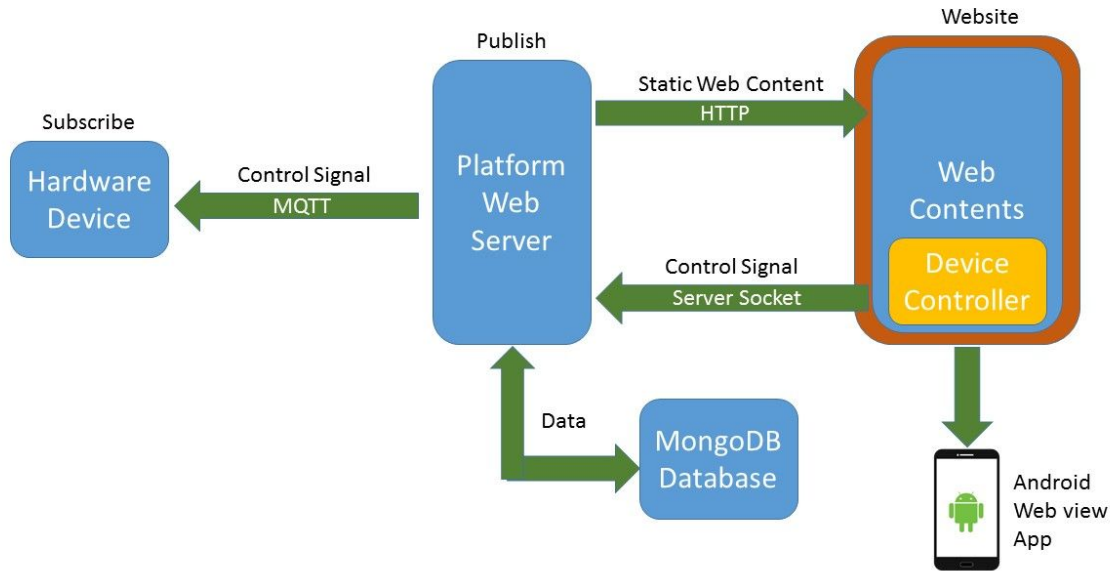
- The user can view the current reading of the sensors through the gauge chart.
- Time vs Value data of the device is plotted through line chart in real time.
- The user can see day-to-day analysis of the device readings through a line chart.



Real Time Data-Streaming

Controlling of the Device:

- The user can **turn ON/OFF** the device through a toggle switch.
- The user can **schedule a timer** for turning ON/OFF the device depending upon his requirements.
- The user can also **change the intensity** of the device through intensity slider.



Controlling the Device

Energy Consumption:

- The user can view the **energy consumed** by the device in a day

Project Creation

- The user can also create Projects and **aggregate different devices together** that have similar functionalities associated with the project.

Installation

Pre Requisites:

Node.js : <https://nodejs.org/en/>

MongoDB: <https://www.mongodb.com/what-is-mongodb>

NODE-RED: <https://nodered.org/>

Steps to Install:

- Download/ Clone the repository in the folder
`git clone <link>`
- To install all dependencies, go to the folder the repository is in and run
`npm install`

3. To start mongodb, open a new terminal and run the command

```
sudo mongod
```

(If it doesn't work, try running 'sudo service mongod stop' and run the command again)

4. To run NODE-RED on the website, open a new terminal and run

```
node-red
```

5. To run the server, use the command

```
node app
```

6. The website will run on <http://localhost:3000>

7. To test the device visuals, open another tab and run :

- a. **Through HTTP GET Request:**

http://localhost:3000/api/users?device_id=<device_id>&value=<value>

- b. **Through MQTT Protocol:**

- i. Open a new terminal and type

```
mqtt pub -t <device_id>/values -h <host_address> -m <value>
```

where device_id is the device id of the device where you are plotting.

value is the value you want to plot.

host_address is the IP address where the MQTT server is hosted.

Goals

V2.0 has the added functionality :

1. Allowing the user to **add various devices** based on his requirements.
2. Allowing the user to **create Projects** and add multiple devices under each project.
3. **Real time data streaming** on gauge and line charts.
4. **Real time energy analysis** on gauge graph.
5. **Date to date analysis** of device data based on starting and ending date set by the user on a line chart.
6. **Controlling remote devices** from the dashboard. (Activating or deactivating devices through the switch or by setting the timer).
7. Controlling the intensity of the devices from the dashboard.

8. Creating a mobile application encompassing all the features of the web application.

V3.0 will have the added functionality :

1. Adding data analytics techniques.
2. More sophisticated encryption/decryption protocol.
3. Login via facebook, twitter, and google.

Specifications

Technology Used

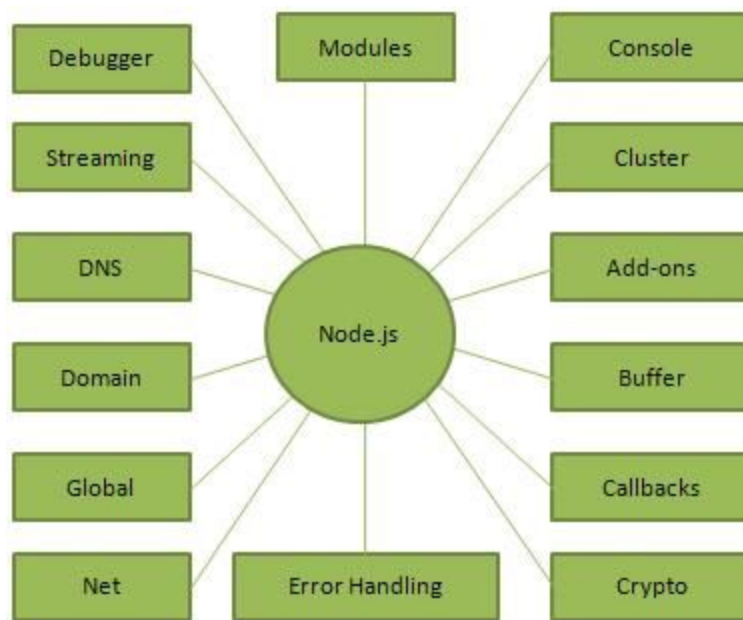
1. Bootstrap
2. Mongo DB
3. Canvas Gauges
4. JQuery: 3.1.1
5. Node JS
 - bcrypt-nodejs: 0.0.3,
 - body-parser: ^1.17.2,
 - canvas-gauges: ^2.1.4,
 - chart.js: ^2.5.0,
 - chartjs: ^0.3.24,
 - connect-flash: ^0.1.1,
 - cookie-parser: ^1.4.3,
 - cookieparser: ^0.1.0,
 - email-verification: ^0.4.6,
 - epoch-charting: ^0.8.4,
 - express: ^4.15.2,
 - express-session: ^1.15.2,
 - format: ^0.2.2,
 - logger: 0.0.1,
 - mongoose : ^4.9.8,
 - morgan: ^1.8.1,
 - mqtt: ^2.8.2,
 - nodemailer: ^4.0.1,
 - passport: ^0.3.2,
 - passport-local: ^1.0.0,
 - sleep: ^5.1.1,
 - socket.io: ^2.0.1,
 - ejs: ^2.5.6
6. Node-RED

7. Mosquitto

NodeJS

¹Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. It is an open source server framework that allows us to use javascript on the server. Node.js runs the script server side allowing the user to create dynamic web pages. It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

For more information, visit: <https://nodejs.org/>



MongoDB

MongoDB is a document based, NoSQL , free and open-source, published under a combination of the [GNU Affero General Public License](#) and the [Apache License](#). database written in C++. MongoDB is based on documents instead of tables which provides it an added flexibility. In MongoDB one collection holds different documents whose number of fields, size etc. can vary.

The data in the database is stored in the form of JSON files.

For more information, visit: <https://www.mongodb.com/what-is-mongodb>

Bootstrap

¹ Tutorialspoint - https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

²Bootstrap is a free and open-source front-end web framework for designing responsive websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

For more information,visit: <http://getbootstrap.com/>

Canvas Gauges

Canvas gauges are open source minimalist HTML5-based components for web applications.

For more information,visit: <https://canvas-gauges.com/>

JQuery

JQuery is one of the most popular javascript library used to make the client-side scripting of HTML easier.It makes it easier to create [Document Object Model\(DOM\)](#) based elements,handle page navigations and create plugins.

For more information,visit:<https://jquery.com/>

Node-RED

Node-Red is a software tool built on Node.js developed by IBM which makes it easier to wire together hardware devices. It provides simple browser based editing facility which makes it easier to create flows just by a simple drag of the mouse. The flows can then be deployed at runtime.

For more information,visit:<https://nodered.org/>

Mosquitto

Eclipse Mosquitto is an open source message broker which implements the [MQTT](#) protocol of communication. A broker is primarily responsible for receiving messages from the client,filtering them and sending them to appropriate subscribed clients.It is the counterpart of a MQTT client.

³MQTT provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for "Internet of Things" messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers like the Arduino.

For more information,visit:<https://mosquitto.org/>

² [https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))

³ Mosquitto: <https://mosquitto.org/>

Mongo DB Docs

Mongoose Documentation: <http://mongoosejs.com/index.html>

Passport Documentation: <http://passport.org>

Auth Schema

This schema is used for the user verification through email address. Whenever a new user is created the details are stored here and the state is assigned to false. After the email verification the state is changed to true and the same details are stored in user schema.

```
var authSchema = mongoose.Schema({  
  
  local      : {  
    api_key   : String,  
    // a unique string to identify each user(generated server-side)  
    email     : String,  
    // email id of the user: verification email will be sent on this mail  
    password  : String,  
    // password of the user: for logging in after verification, hashed  
    url       : String,  
    // url generated server-side to be sent to the email for verification  
    state     : Boolean  
    // If verified: state = true -> schema is deleted from collection, else false  
  
  },  
  
  //for future login functionality using facebook, google and twitter  
  
  facebook   : {  
    api_key   : String,  
    id        : String,  
    token     : String,  
    email     : String,  
  
  },  
  
  google     : {  
    id        : String,  
    token     : String,  
    email     : String,  
  
  },  
  
  twitter    : {  
    id        : String,  
    token     : String,  
    email     : String,  
  
  },  
  
});
```

State	Meaning	Action
true	The user email is verified	Document deleted from auth Schema and added to the userSchema
false	The user email is still unverified (default setting)	-NA-

```

    name      : String
  },
  twitter    : {
    api_key   : String,
    id        : String,
    token     : String,
    displayName : String,
    username  : String
  },
  google     : {
    api_key   : String,
    id        : String,
    token     : String,
    email     : String,
    name      : String
  }
},
{
  collection: 'authSchema',
  safe: true
});

```

User Schema

This schema is used storing the user details that is required for login. Data is stored in this schema after the verification of the email address.

```

var userSchema = mongoose.Schema({
  local      : {
    api_key   : String,
    // a unique string to identify each user(generated server-side)

    email     : String,
    // email id of the user: verification email will be sent on this mail

    password  : String,
    // password of the user: for logging in after verification, hashed using b-crypt node
  },
  //for future login functionality using facebook, google and twitter
});

```

```
facebook    : {
  api_key    : String,
  id         : String,
  token      : String,
  email      : String,
  name       : String
},
twitter     : {
  api_key    : String,
  id         : String,
  token      : String,
  displayName : String,
  username   : String
},
google      : {
  api_key    : String,
  id         : String,
  token      : String,
  email      : String,
  name       : String
}
};
```

User Details Schema

This schema is used to store the details of the users. It contains unique api_key which is used to identify the user. It also contains list of devices and projects the user is working on.

```
var userDetailsSchema = mongoose.Schema({
```

```
  api_key : {type:String, unique:true},
  Unique string to identify each user
```

```

email : String,
    Email of the user

devices : Array,
    Array of device ids added by the given user

projects : Array
    Array of projects ids created by the given user
},
{
    collection:'userDetails', //Stored in collection userDetails
    safe:true
});

```

Notification Schema

This schema is used for storing the notification details of the events according to the time. The notifications are sent by the device. Some examples are: Already On/off, Device is switched on/off, Device is connected/disconnected etc. These notifications are displayed in the notifications panel on the device visuals page.

```

var notification = mongoose.Schema({
    time : {type : Date, default: Date.now},
        //time of receiving the notification from the device

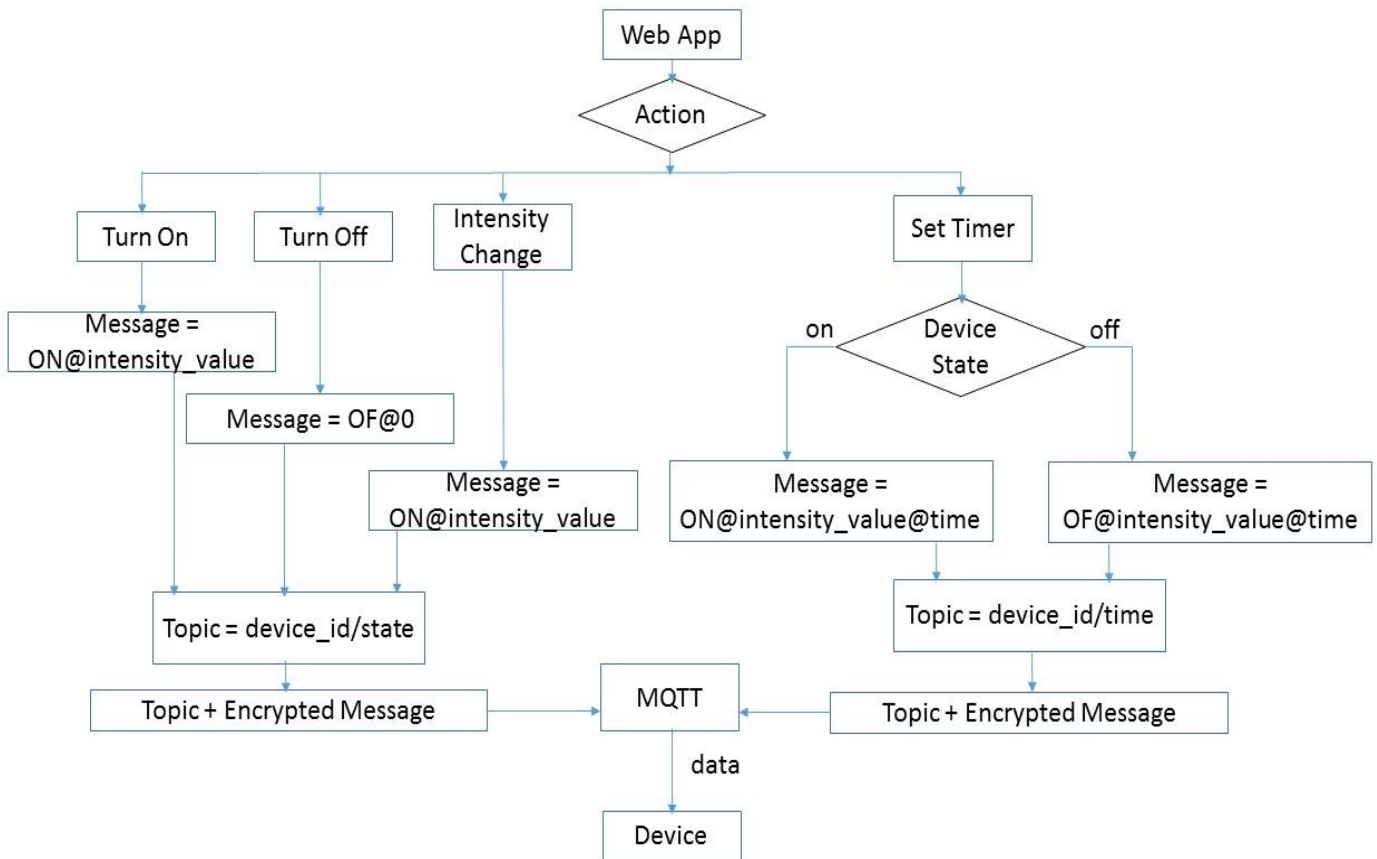
    message: String
        // message conveyed by the notification

});

```

Notificati on sent by Device	Message	Meaning
AO@12	Device was already on at intensity 12%	A signal was sent from the application to switch on the device, but the device was already on.
AF@0	Device was already off.	A signal was sent from the application to switch off the device, but the device was already off.
ON@12	Device is switched on at intensity 12%	The device was switched on at intensity 12%

OF@0	Device switched off	The device was switched off.
AU	Authenticated User Scanned	The user is authenticated by scanning an RFID card.
UU	Unauthenticated User Scanned	The RFID card scanned by the user is not valid.
CN	Device Connected	MQTT connection has been established between the device and the application.



Project Schema

This schema contains the details of the project the user is working on. It contains list of all the devices that are used in the project.

```

var projectData = mongoose.Schema({
  project_name :String,
    // a user inputted string to identify the project
  project_id : String,
    // a server generated string to uniquely identify a project.

```

```

description: String,
    // a user inputted description of the project
device_id: Array
    // array of device_ids corresponding to the devices in the project

},
{
  collection:'projectData',
  safe:true
});

```

Sensor Schema

This schema is used to store the device details. It also contains list of project IDs where the sensor is associated with.

```

var sensorSchema = mongoose.Schema({
  device_name : String,
    // user inputted string to identify the device
  device_id  : { type: String, unique: true },
    // server generated string to uniquely identify a device
  description : String,
    // user inputted description of the device
  last_access : {type: Date, default: Date.now},
    // time of creation of the device (name is misleading)
  device_type : String,
    // user inputted : type of device
  timer      : {type:String,default:"false"},
    // boolean, if timer is set or unset
  timerDate  : String,
    // if timer is set, the date to which it is set to
  project_id  : {type:String,default:""},
    // if the device belongs to a project, this field stores the id of the project it belongs
    // to, else stores nothing
  intensity   : {type:String,default:"0"},
    // last set intensity of the device

```

```
state    : {type: String, default: "false"},  
    // state is true if device is on, else false
```

State	Meaning
True	Device is on
False	Device is off

```
activity : [notification],  
    // array of all the notifications sent by the given device to the application.  
values : [sensorDataSchema],  
    // stores all the time-value pairs used to plot the data sent by the device  
energy : []  
    // Array of numbers, stores the energy values for the last 30 days.  
    // The indices of the array correspond to the days of a month.  
},{  
  collection:'sensor',  
  safe:true  
});
```

Sensor Data Schema

This schema is used to store the sensor data values and the time when the device has to be turned on/off.

```
var sensorData = mongoose.Schema({  
  time : {type : Date, default: Date.now},  
    // timestamp of the instant when the data value to be plotted is received by the  
application  
  value : Number  
    // data value to be plotted, sent by the device  
});
```

Usage

I. Sign Up Page

➡ Signup

Email

Password

Signup

Already have an account? [Login](#)

Or go [home](#).

This page is used for signing up of new users to access the platform. The user needs to enter his email ID and password which is used for user authentication during the login to the platform. To authenticate the user, an email is sent to the registered email. On clicking on the link provided in the email, the user account setup is completed and the user can then login to the application.

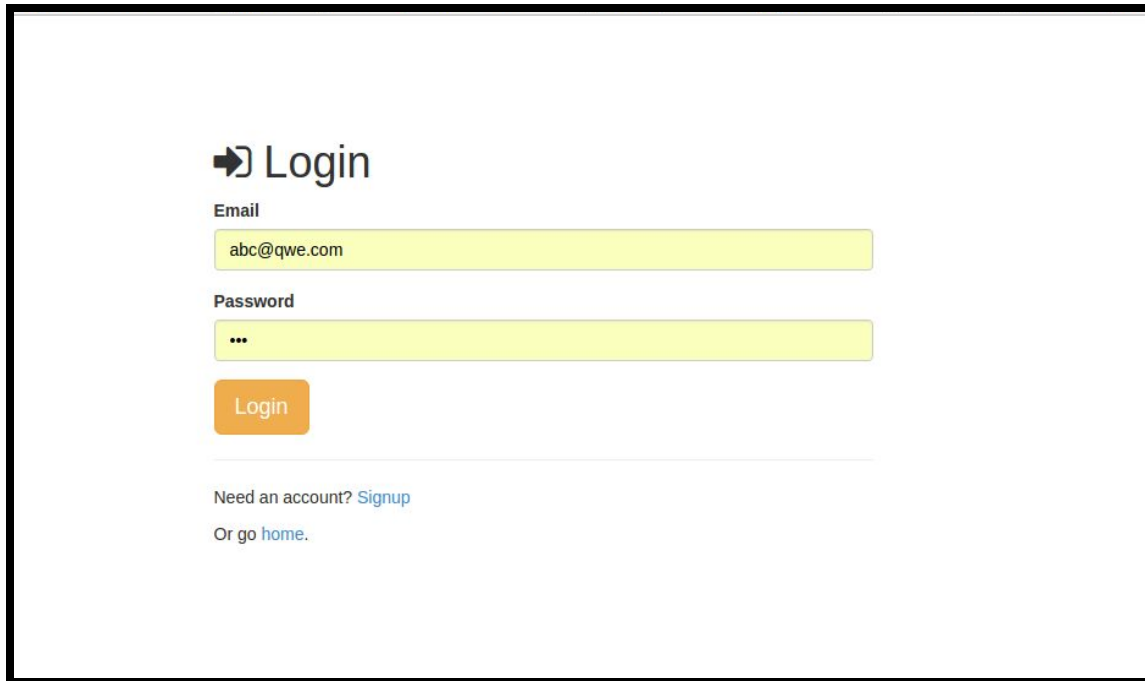
Frontend Checks:

1. Email is invalid: asked to enter again.
2. Both email and password are required.

Backend Checks

1. Email exists in database: user already exists, shows message.
2. Email does not exist in database: create a new entry in the authschema in the database, hash the password and send email to the registered email account for account verification. Email is sent through the email - iot.fossee@gmail.com.
3. If the user clicks on the sent link, delete user entry from the authSchema and enter in the userSchema. User account verification is successful.

II. Login Page

A login form with a black border. At the top, there is a logo consisting of a square with a right-pointing arrow, followed by the text "Login". Below the logo, there are two input fields. The first is labeled "Email" and contains the text "abc@qwe.com". The second is labeled "Password" and contains three dots. Below the password field is an orange button with the text "Login". At the bottom, there is a link "Need an account? Signup" and another link "Or go home.".

Logs in the user if email-password pair is present in the users Schema and gives access to the platform to the users.If the user authentication is not completed(If user has not clicked on the link set through email), access is denied.

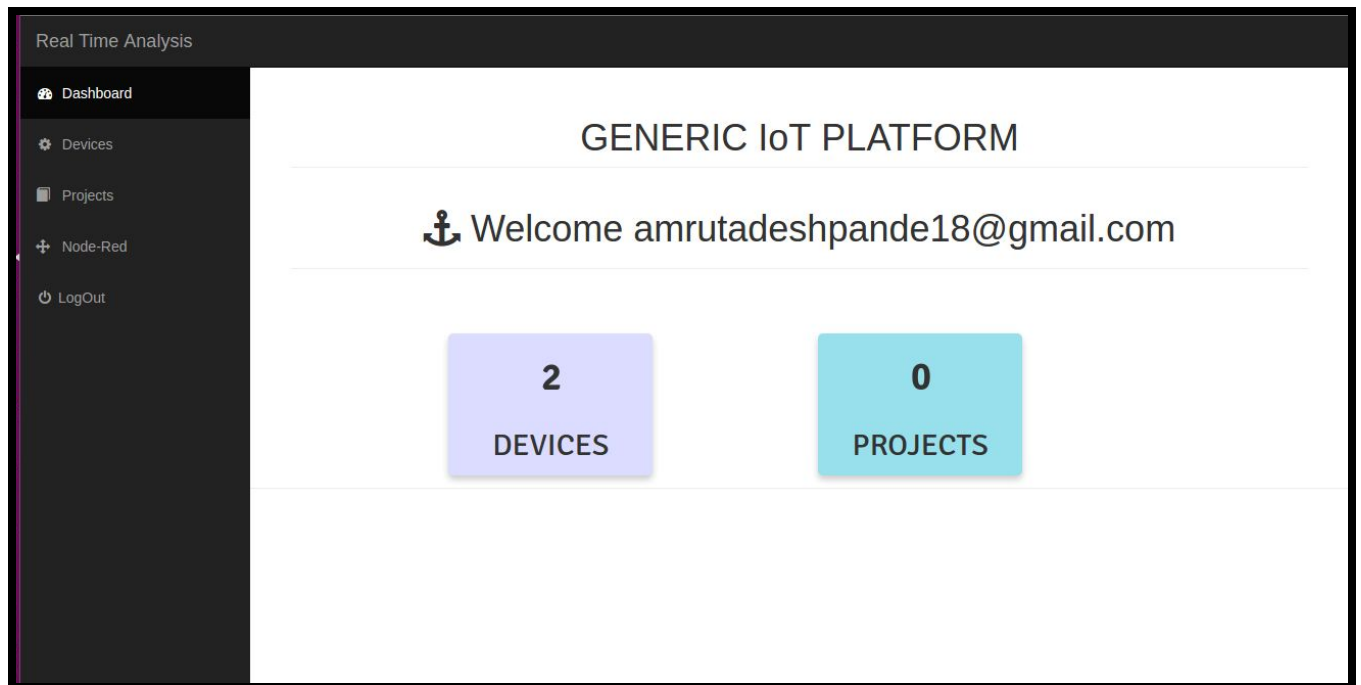
Frontend Checks:

1. Email is invalid: asked to enter again.
2. Both email and password are required.

Backend Checks:

1. Checks if the entered email is registered in the platform by checking the userSchema. If it is not present shows "User does not exist".
2. Checks if the entered password matches to the email from the userSchema. If the password is not matched it shows "Oops! Wrong password"

III. Profile Page/ Dashboard

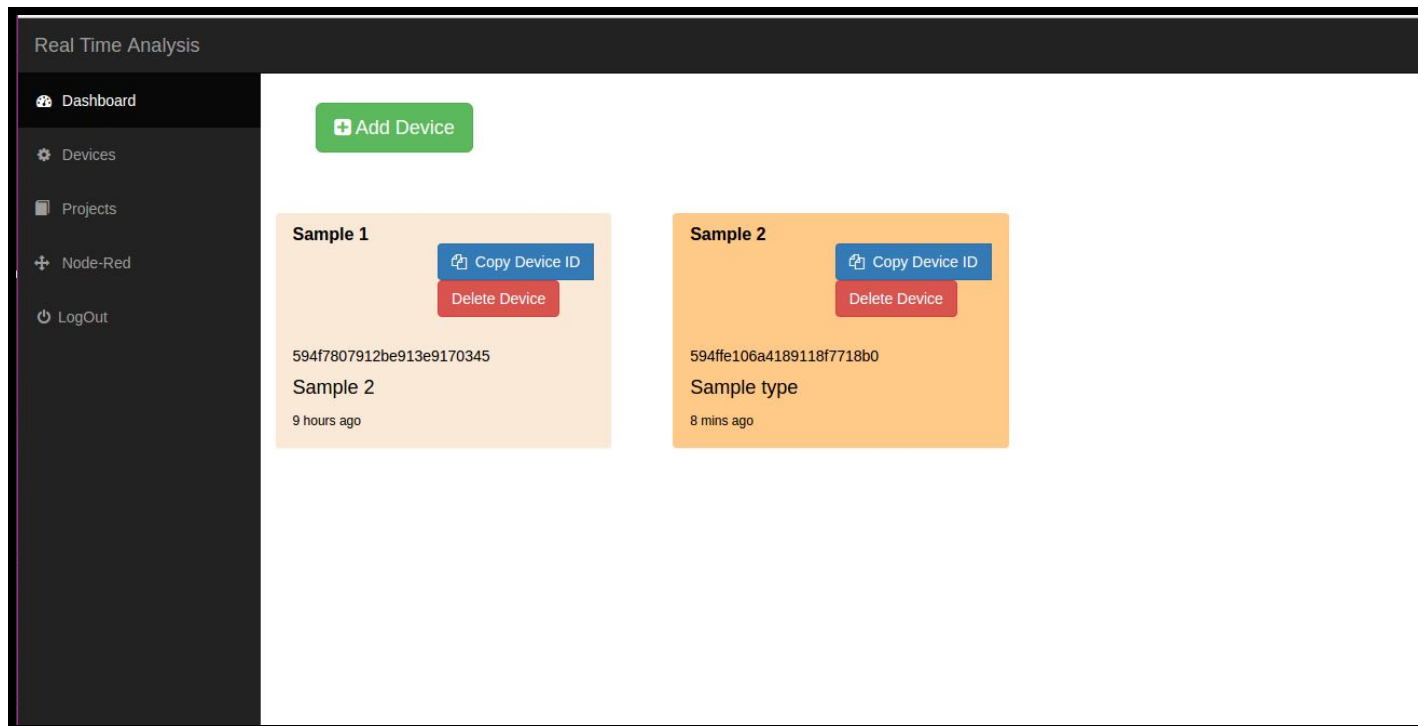


The dashboard or the profile page of the user shows the number of devices added by the user and the number of projects added by the user.

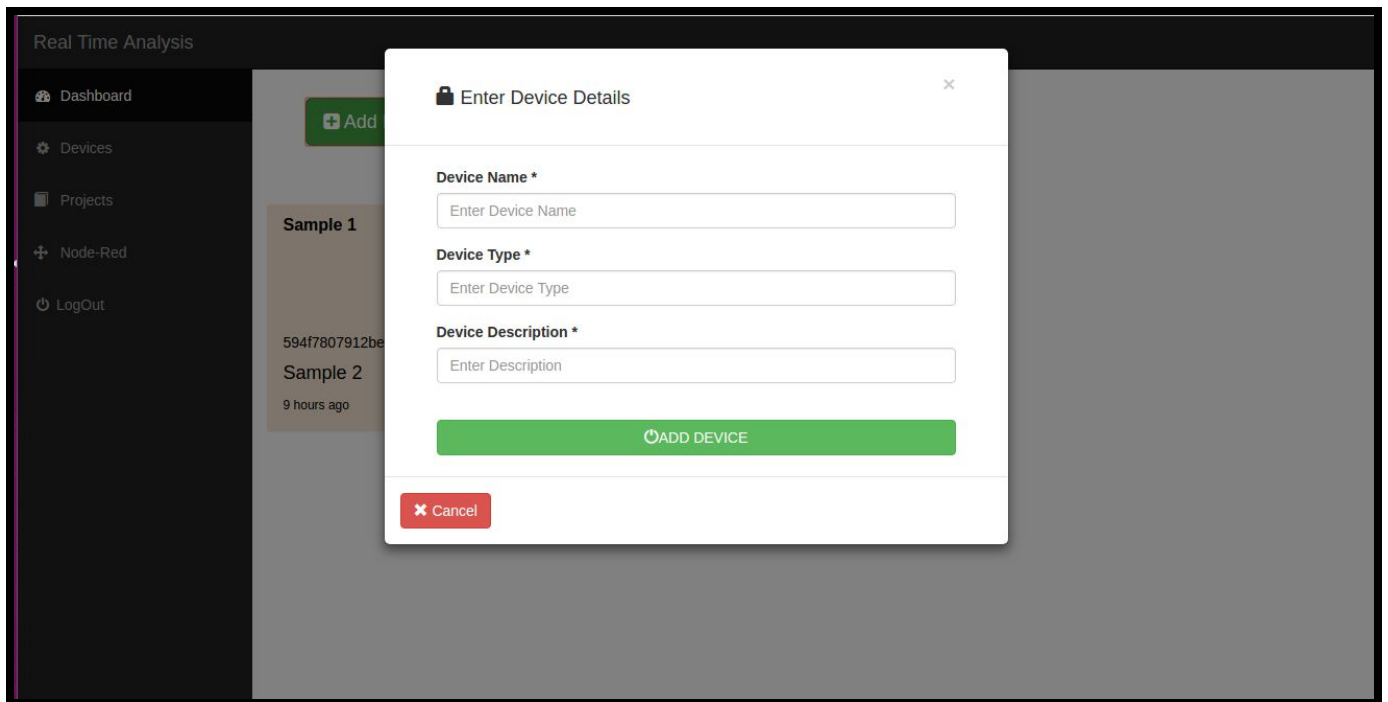
The side navigation bar has links to the following:

1. **Devices** : Displaying all the devices added by the user and a button for adding more devices.
2. **Projects** : Displaying the list of all the projects created by the user and a button for creating more projects.
3. **Node-Red**: [Node-Red](#) is a programming tool for adding together various hardware devices.
4. **LogOut** : For logging out of the account.

IV. Devices Page

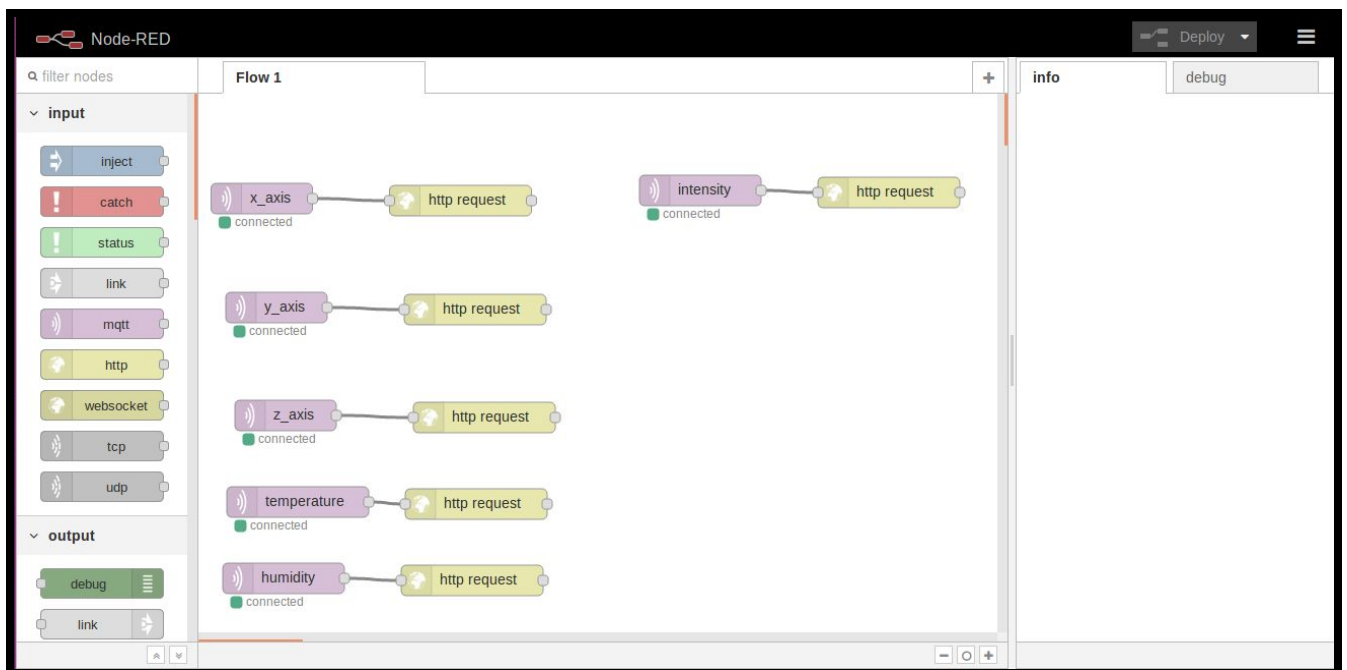


1. Shows all the devices added by the user in the form of attractively colored cards. The cards take up random colors each time the page is refreshed.
2. The cards display device name, id and device type.
3. The cards are clickable, on clicking the card, the visual page for the device opens.
4. The card also has '**Copy Device ID**' button which allows the user to copy the device id for use in the hardware code.
5. The '**Delete Device**' button allows the user to delete the particular device.
 - a. A modal window pops up on clicking the 'Delete Device' button, confirming the deletion.
6. The '**Add Device**' button allows the user to add new device. On clicking the button, a modal form appears which looks like this:



The modal window for adding a device. Opens when the 'Add Device' button is clicked.

V. Node-RED



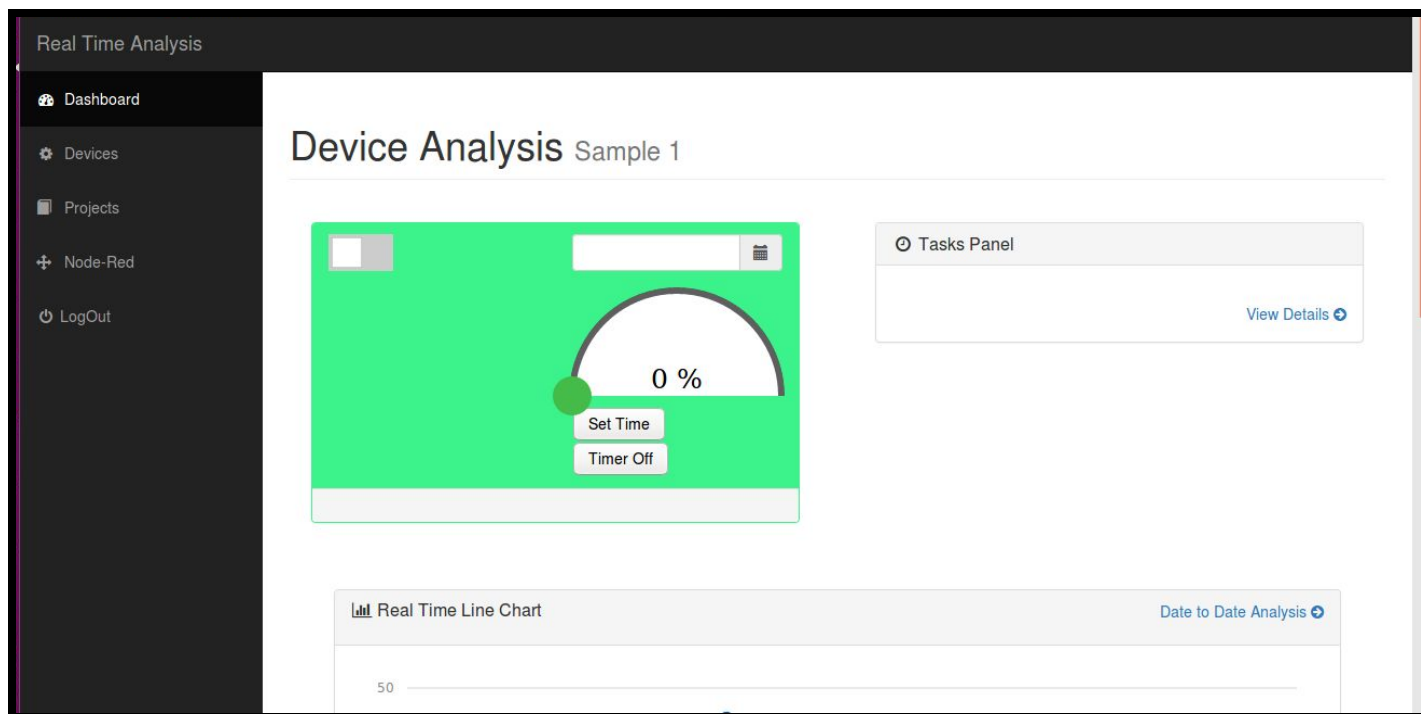
⁴[Node-RED](https://nodered.org/) :Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

The user can use this tool to experiment with various devices by sending and receiving the data over MQTT protocol.

VI. Visuals

The visual pages of the device is shown in the following screenshots.

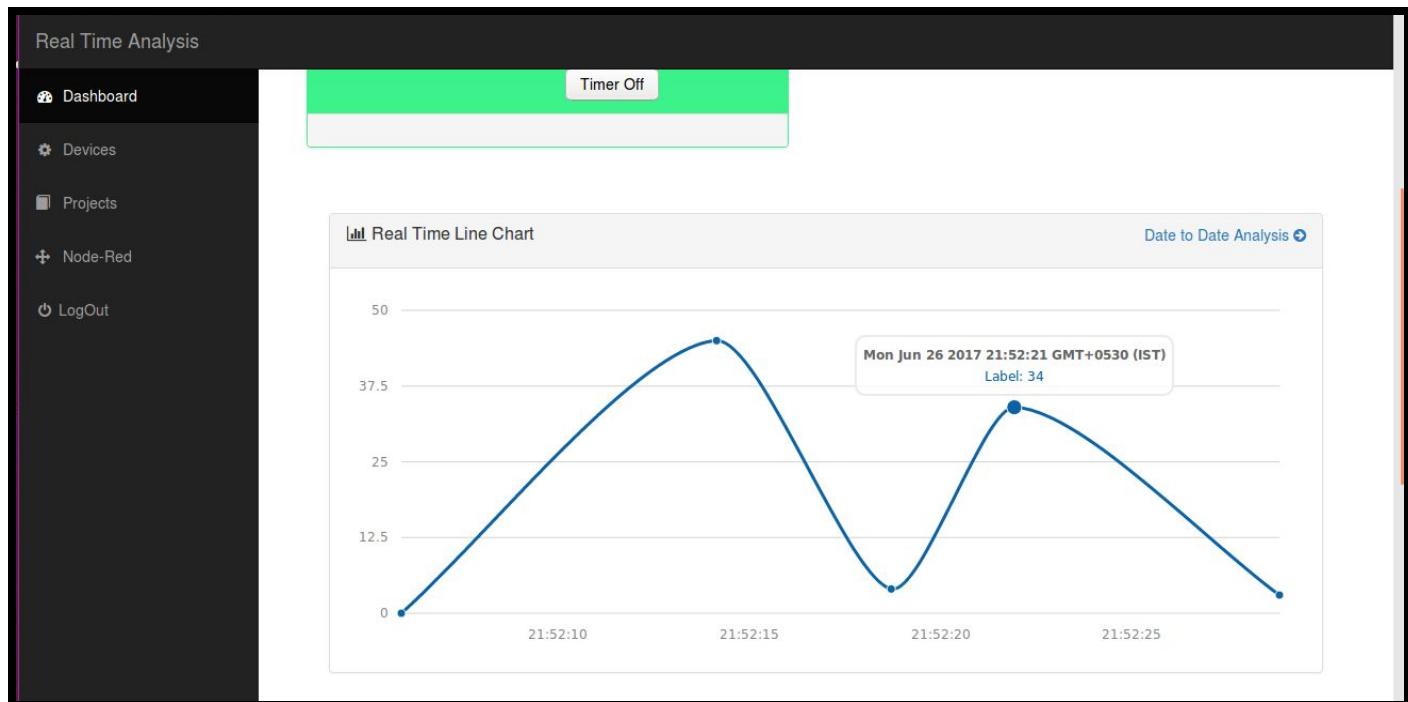


The top portion of the visual page shows controls for switching the device ON/OFF and controlling the intensity of the device.

The user can control the device directly by the switch or can set the timer.

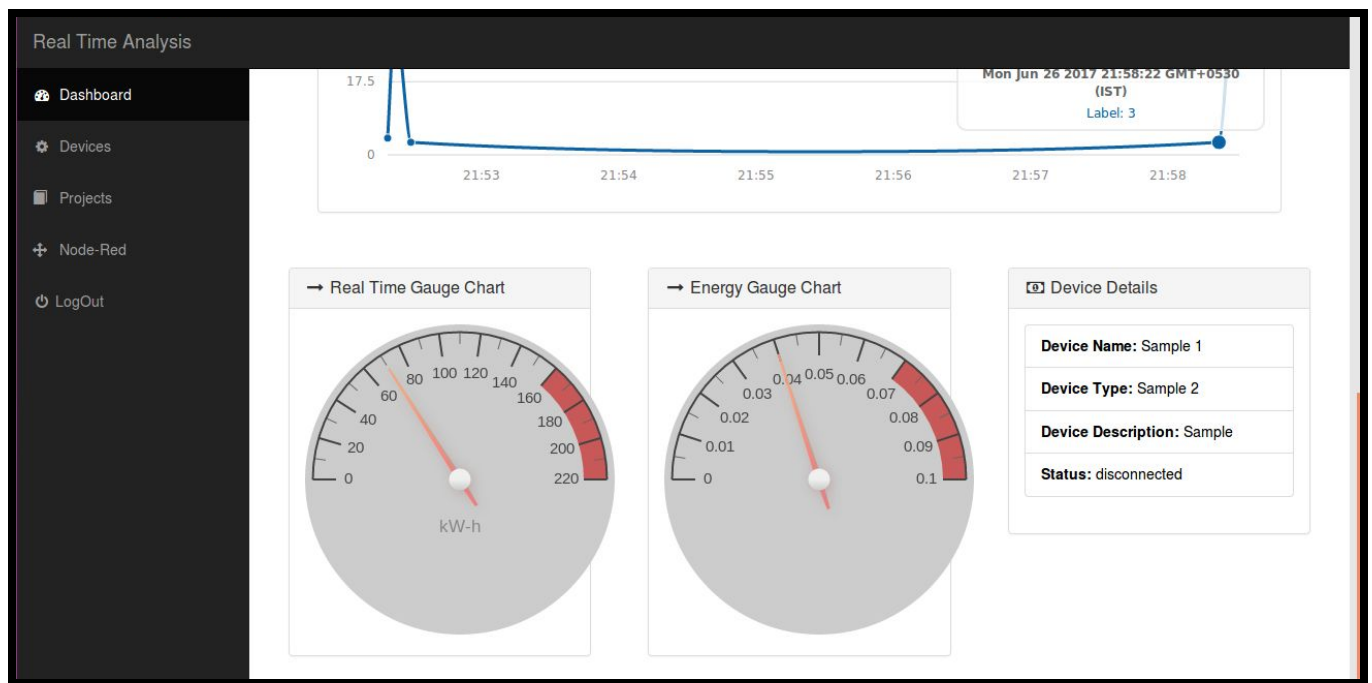
When the timer is set then the datepicker is disabled. On expiration of the timer an expire message is displayed on the screen. To set the timer again or control the device press the 'Timer Off' button. The right side shows a task panel which displays the notifications sent by the device.

⁴ NODE-RED: <https://nodered.org/>



The above screenshot shows the middle portion of the visual page displaying the line chart. The line chart shows the real time data sent by the device over MQTT.

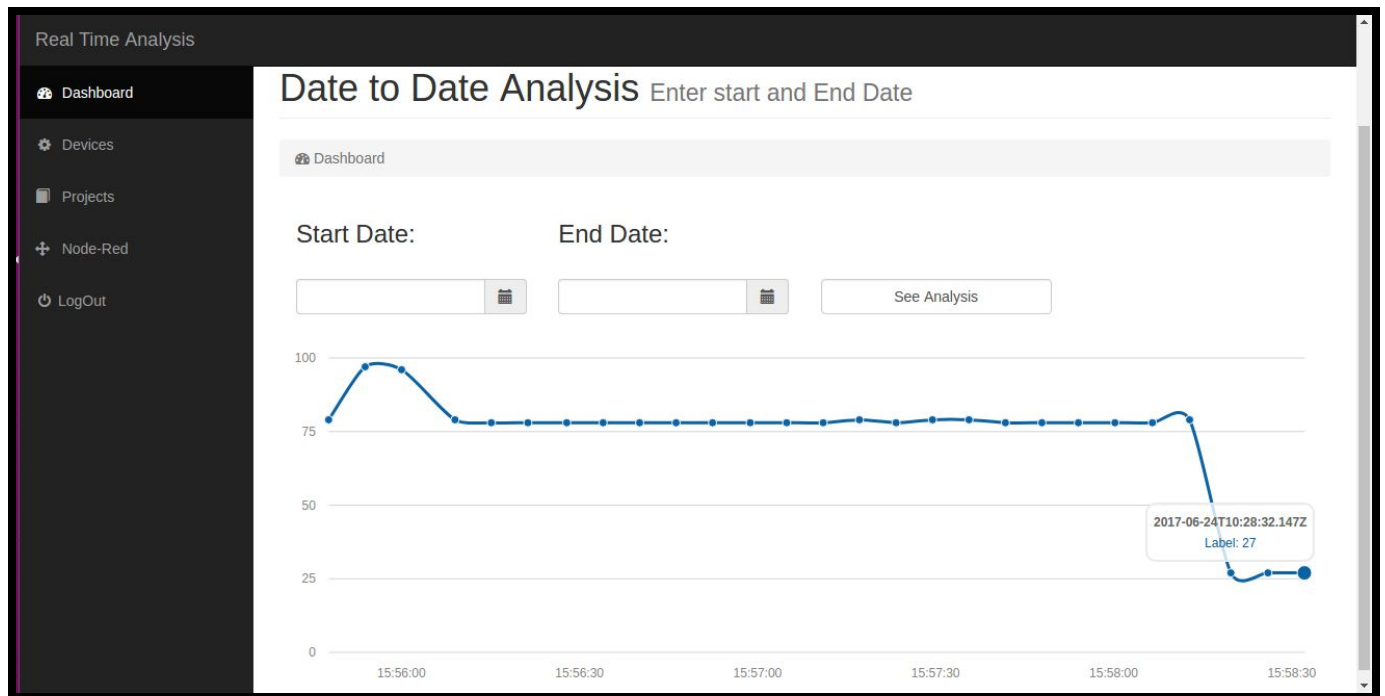
The line chart is made using the [morris.js](https://dmitrybaranovskiy.github.io/morris.js/) library.



The above screenshot shows the bottom portion of the visual page.

It shows a **real time gauge graph** plotting real time device data. The **energy gauge graph** in the middle portion shows energy consumption of the device till that time. The bottom right side shows the **device details**.

VII. Date to Date Data Analysis



This page shows the analysis of data from a starting date to an ending date by plotting a line chart of the data between that period.

The user can enter the starting and ending date and the appropriate values are taken from the database and plotted on the chart.

Data of the same date as current cannot be displayed if the sensors are sending data to plot real-time on the visuals page.

VIII. Projects

Real Time Analysis			
<div>Dashboard</div> <div>Devices</div> <div>Projects</div> <div>Node-Red</div> <div>LogOut</div>	<div>Create Project</div>		
	Sno	project ID	project Name
	project Description		
	1	59500b356a4189118f7718b2	Project 1
			Project 1 Description

The page shows a list of projects added by the user, clicking on the project shows the devices added under that project.

The Projects Section can be used when some devices have similar uses.

The devices are grouped together under a common project, can be seen together under one project name.

These devices will also be visible individually in the devices section.

Clicking on a device leads to a page like the Visuals page. It gives the details and controls of that device.

MQTT Communication

All the communication between the device and the application is through the MQTT protocol. ⁵MQTT^[1] (MQ Telemetry Transport or Message Queue Telemetry Transport) is an ISO standard (ISO/IEC PRF 20922)^[2]publish-subscribe-based "lightweight" messaging protocol for use 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. The broker is responsible for distributing messages to interested clients based on the topic of a message.

All the messages sent on the topic are encrypted using the encryption protocols described in the Encryption/Decryption section.

⁵ Wikipedia: <https://en.wikipedia.org/wiki/MQTT>

The following are the details of the messages being sent and received between the application and the device:

Topic	Message	Meaning	Published/ Subscribed
device_id/state	T@12 F@0	T@12: Device is switched on at intensity 12% from the application.	published
device_id/time	T@12@123	Device is to be switched on at intensity 12% after 123 seconds. (Set Timer at the application)	published
device_id/notify	ON@12	Device is switched on at intensity 12%. (More details below)	subscribe
device_id/values	45	Value to be plotted, sent by device	subscribe
device_id/energy	30	30 kW-h of energy has been consumed by the device from start of the day till current instant(the instant when the notification is received by the application.	subscribe

→ The /state topic :

◆ **Data Published** : ON/OFF signal with intensity

◆ **Format** : <state>@<intensity>

- state :
 - T => device is switched on
 - F => device is switched off
- Intensity: intensity at which device is to be switched on.

◆ **Function** :

- When the device is switched on/off from the device dashboard (Visuals Page) a message is sent to the actual device on this topic.
- User can set the intensity at which the device is to be switched on.
- The signal sent on this topic is executed by the device instantaneously.

→ The /timer topic:

◆ **Data Published** : ON/OFF signal with intensity and timer

◆ **Format :** <state>@<intensity>@<time>

- state :
 - T => device is switched on
 - F => device is switched off
- Intensity: intensity at which device is to be switched on.
- Time: Scheduling time after which the state has to change

◆ **Function:**

- When the timer to switch the device on/off is set from the device dashboard (Visuals Page) a message is sent to the actual device on this topic.
- User can set the intensity at which the device is to be switched on.
- The signal sent on this topic is executed by the device after the specified time interval.

→ The /value topic :

◆ **Data Subscribed :** value sent by the device to be plotted on the device dashboard.

◆ **Format :** <value>

- Value : value to be plotted on the graphs on the device dashboard.

◆ **Function :**

- A value is sent by the sensors connected on the device to the dashboard.
- This value is instantaneously published on the /value topic by the device, and subscribed by the server when the device visuals page is open. Values are not subscribed if the device visuals page is not open.
- The value is sent through a socket to the graphs (Line and Gauge Chart) where it is instantaneously plotted.
- The value of the device is stored in the MongoDB server.
- The stored values can be replotted by entering the start and end date in the Date to Date Analysis section. (Link over the line chart)

→ The /energy topic:

- ◆ **Data Published :** Energy consumed by the device from the start of the day till the current instant
- ◆ **Format :** <energy>
 - Energy: The value of the energy consumed by the device.
- ◆ **Function :**
 - Shows the energy consumed by the device on that day.
 - The energy is plotted on the Energy gauge on the device visuals page.
 - The energy of the past month is stored in the mongoDB database.
 - An array of 31 elements is created in the sensor collection of the MongoDB database.
 - The date on which the energy is sent is found using Date().getDate() function.
 - The energy value is stored at the array index corresponding to the (date -1)
 - In future versions, these can be accessed and plotted on a line graph.

→ The /notify topic:

- ◆ **Data Published :** The messages about the actual device state from the device are received on the /notify topic.
- ◆ **Format :** Various messages have various formats. All messages start with the message code explaining the meaning and usage of the message.

Notificati on sent by Device	Format	Message	Meaning
AO@12	AO@<intensity>	Device was already on at intensity 12%	A signal was sent from the application to switch on the device, but the device was already on.
AF@0	AF@0	Device was already off.	A signal was sent from the application to switch off the device, but the device was already off.
ON@12	ON@<intensity>	Device is switched on at intensity 12%	The device was switched on at intensity 12%

OF@0	OF@0	Device switched off	The device was switched off.
AU	AU	Authenticated User Scanned	The user is authenticated by scanning an RFID card.
UU	UU	Unauthenticated User Scanned	The RFID card scanned by the user is not valid.
CN	CN	Device Connected	MQTT connection has been established between the device and the application.

◆ Function :

- For switching a device on and off, the notify acts as a handshake between the device and the application.
- For example, to switch on the device, the user can use the slider switch on the visuals page. Switching it on publishes on the “state” topic a message of the format “ON@intensity” when intensity is the current intensity set on the intensity slider. When such a notification is received by the device, it switches the device on and publishes on the “notify” topic that the device was switched on in the format “ON@intensity”. This is especially useful if the device fails to switch on the device, or some error has occurred, and hence it fails to publish on the “notify” topic. Since the handshake was never completed, the state of the device remains unchanged in the database and no changes are reflected on the application.
- All the messages are stored in the notifications array in the sensor collection. The most recent notifications are shown under the ‘Tasks Panel’ on the device visuals page along with the time.

Encryption/ Decryption

It is important to secure the messages that are being sent and received at application level. Since we are using MQTT protocol for communication between a client and server, two methods can be used to secure the payload. They are as follows:

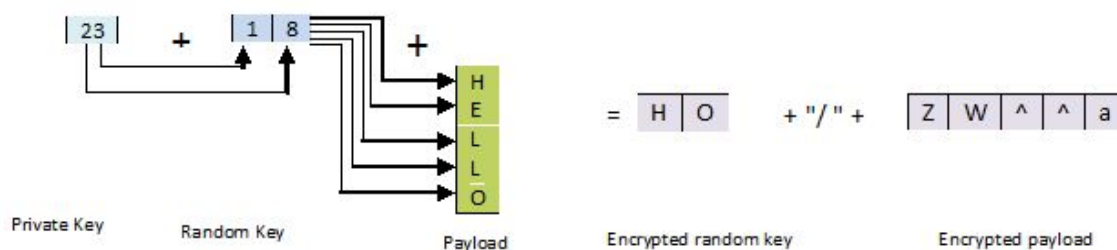
1. SSL / TLS (Network layer Security)
2. Payload Encryption and Decryption (Application layer Security)

Since our project is application based, we chose encryption and decryption to secure the payload. Encryption-Decryption follows a simple algorithm. In order to avoid a third party from viewing the messages that you are publishing to a particular topic, we

need to hide the messages or in other way, display them such that it makes no sense to the third party and only the authorized user can decrypt and get the correct message. This doesn't allow the third party to view or change the messages.

To encrypt a message:

1. A randomly generated integer (called the **random key**) is added to each character of the payload string. This random key is actually added to the ASCII value corresponding to that payload character. The character corresponding to this sum (in ASCII table) is stored in a character array. This character array is now the encrypted payload.
2. The random key is also sent with payload to decrypt the message at other end. To do so, a constant key (called the **private key**) which is known only to the two parties authorized to communicate. This secret key is called private key and is different for different setups.
3. The random key generated in first step is encrypted using the private key (using the same algorithm used to encrypt the payload) .
4. The encrypted random key and encrypted payload strings are then concatenated with a special character separating them. This helps the decryptor to identify the random key and payload separation. Eg. : "Encrypted random key" + "/" + "Encrypted payload";
5. This encrypted string is now secure and can be published to the required topic.

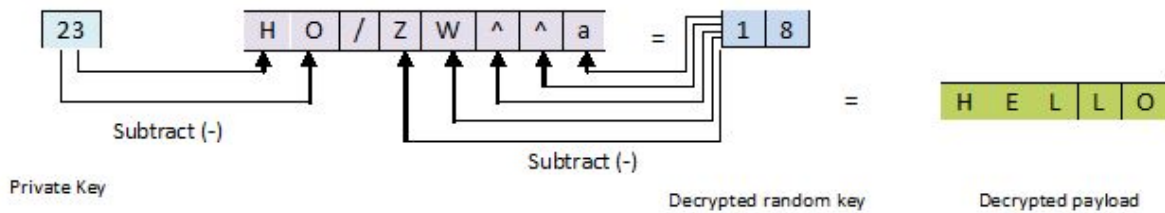


To decrypt a message:

1. The encrypted message is scanned till the special character separating the encrypted random key and encrypted payload ("/"). Subtract the private key from ASCII value of all the characters of encrypted message until the special character ("/"). The difference gives the ASCII value corresponding to the random key used to encrypt the payload at the other end.
2. The character array is converted corresponding to this difference into an integer. This integer is now the random key.
3. The decrypted random key is now used to decrypt the encrypted payload . This is done by subtracting the random key from each character of the encrypted payload . The characters corresponding to this difference (ASCII table) gives the decrypted

payload.

4. The decrypted payload can be further used in application by the decrypting party.



Message Format

There are some limitations on the messages that can be sent by encrypting them with the above scheme. These are:

1. Messages should have characters below the ASCII value of 91(including 91).
2. Extended ASCII is not supported. All encrypted messages will also have characters of normal ASCII.
3. Private and random keys can be any number between 1 to 36.

Android Mobile Application

The Android app for the platform is created by using webview. A "webview" is a program packaged within a portable application delivering a hybrid application. Utilizing a webview permits versatile applications to be constructed utilizing Web advancements (HTML, JavaScript, CSS, and so on). Since the platform is mobile responsive webview is used to make the android app.

Whenever the app is opened the login page is visible to the user, where he needs to authenticate his identity. The rest of the pages are exactly same as that of the web application. This mobile app provides all the functionalities of the web application. The user need not even update the app in the future because the update in the server also changes the interface in the application.

Technologies Used

1. [Android Studio](#) - It is the official IDE for android which provides fast tools for building apps on every type of android device.

How to install Android-studio?

Follow the guidelines in the following link:

<https://developer.android.com/studio/install.html>

2. WebView - WebView is used to run web application inside the android app. Steps for building the webview app is given in the following link below
<https://developer.android.com/guide/webapps/webview.html>

The content of the webview is set to the login page using the following code

```
WebView myWebView = (WebView) findViewById(R.id.webview);
```

```
myWebView.loadUrl("http://<web address of the platform>");
```