About the Project

ABOUT THE PROJECT

The project is about mutual communication between IoT devices and user. Typically, an IoT is a world-wide network of uniquely addressable, and interconnected objects, based on standard communication rules or protocols.

The project is an API that acts has a generic scope for communication with any sort of hardware device irrespective of its functionality. This API helps developer (user) to control the device remotely. This API can be extended to predict the power consumption, over power usage and average life time of the device based on the previous data. Maintaining privacy and user-policies are also a few key aspects.

Developers face difficulty in building an IoT based systems that provide communication between hardware devices (sensors) and servers. This API provides a communication between any hardware devices and servers. It's make easier to develop an IoT based system by using this API. By using this API we get additional services such as user policies, sensor values monitoring, controlling the hardware devices, maintaining the data (by reading sensor values with minimum delay of 5 sec), plotting graphs from sensors data etc. We can Access those services throughout the world.

Project Scope

2. PROJECT SCOPE

2.1 Existing System:

- Developer role is establishment connection between Hardware and Cloud platform.
- Extra skills (MySQL, server-side scripting) needed.
- Handling different IoT project individually.
- Dealing with different Hardware code, communication, gui, etc. forever IoT project is needed.

2.2 Problems in the Existing System:

- Developer Stress.
- No Modularity.
- No Flexibility (Upgrading the existing project is difficult).
- Lack of Integrity.
- Extra skills needed.
- Difficult to manage other developers' projects.
- Can't use sensors data for prediction.

Proposed System:

- Modularity.
- Flexibility.
- Easy Debugging and analyzing.
- End user can handle easily.
- Predicting the better usage, given the sensor data.
- Managing the services easily worldwide.
- API usage makes easier.

Feasibility Report

3. FEASIBILITY REPORT

3.1 Technical Description:

Since this is a software, it lasts for a long time period. Since this is designed with the latest technology available now, it can be used more a few decades. The demand of IoT might fall down, but the demand for this type of requirement sustain till there is electricity in this world.

3.2 REQUIRED HARDWARE:

Hardware requirements

- ESP8266
- Infrared Sensor (IR)
- Servo Motor
- LED (or) Bulb (or) any electric appliances
- Relay

3.3 REQUIRED SOFTWARE:

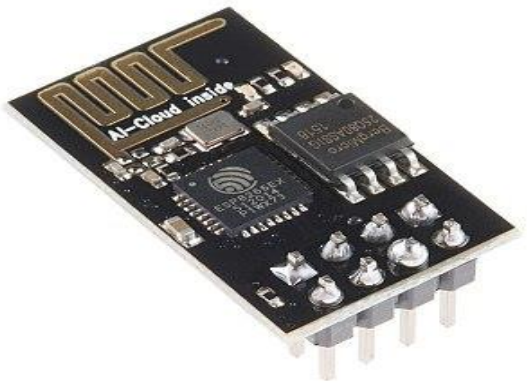
Software requirements

- C++
- Python
- PHP
- HTML / JS
- MySQL
- Public cloud (**000webhost**, AWS…)

Hardware

ESP8266

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by manufacturer Espressif Systems in Shanghai, China. The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker.



Memory: 32 KiB instruction, 80 KiB user data

CPU: @ 80 MHz (default) or 160 MHz

Manufacturer: Espressif Systems

Successor: ESP32

Input: 16 GPIO pins

Features of ESP8266

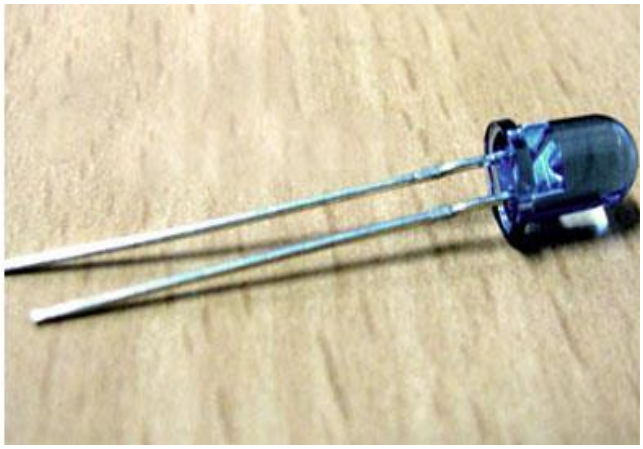
- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz[5]
- Memory:
  - 32 KiB instruction RAM
  - 32 KiB instruction cache RAM
  - 80 KiB user-data RAM
  - 16 KiB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- IEEE 802.11 b/g/n Wi-Fi
  - Integrated TR switch, balun, LNA, power amplifier and matching network
  - WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI
- I²C (software implementation)[6]
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

The pinout is as follows for the common ESP-01 module:

1. VCC, Voltage (+3.3 V; can handle up to 3.6 V)
2. GND, Ground (0 V)
3. RX, Receive data bit X
4. TX, Transmit data bit X
5. CH_PD, Chip power-down
6. RST, Reset
7. GPIO 0, General-purpose input/output No. 0
8. GPIO 2, General-purpose input/output No. 2

Infrared Sensor (IR)

IR sensor circuit is used to detect the presence of IR signals from electronic device**.**



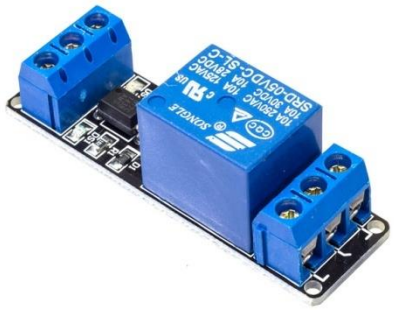
Servo Motor

A servo motor is an electrical device which can push or rotate an object with great precision. If you want to rotate and object at some specific angles or distance, then you use servo motor. It is just made up of simple motor which run through servo mechanism.

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback.

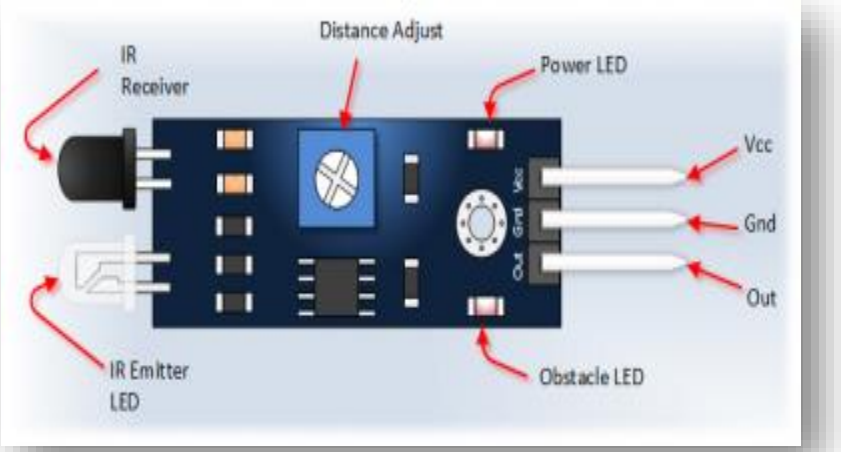
Relay

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays**.**



IR sensors:

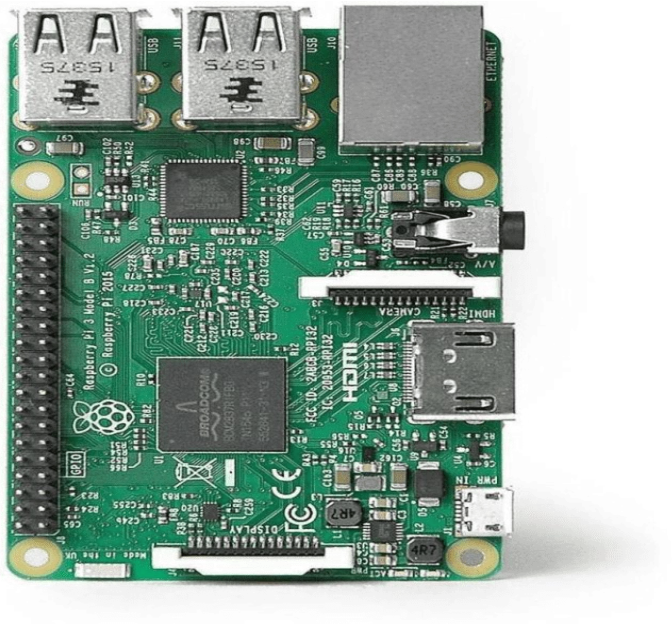
Infrared Obstacle Sensor Module has built-in **IR transmitter** and **IR receiver** that sends out IR energy and looks for reflected IR energy to detect presence of any obstacle in front of the sensor module. The module has on board potentiometer that lets user adjust detection range. The sensor has very good and stable response even in ambient light or in complete darkness.



Raspberry pi:

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals and cases.

According to the Raspberry Pi Foundation, more than 5 million Raspberry Pis were sold by February 2015, making it the best-selling British computer. By November 2016 they had sold 11 million units, and 12.5m by March 2017, making it the third best-selling "general purpose computer". In July 2017, sales reached nearly 15 million. In March 2018, sales reached 19 million.



3.4 FEASIBILITY TYPES:

Technical feasibility:

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- This project is the necessary technology suggested.
- The proposed equipments have the technical capacity to hold the data required to use the new system.
- The proposed system provides adequate response to inquiries, regardless of the number or location of users.
- This technically guarantees the accuracy, reliability, ease of access and data security.

The system is self-explanatory and does not need any extra sophisticated training. As the system has been built by concentrating on the Graphical User Interface Concepts, the application can also be handled very easily with a nowise User.

The system has been added with features of menu driven and touch interaction methods, which makes the user the master as he/she starts working through the environment. The net time the customer should concentrate is on the installation time.

Financial Feasibility:

The system as a whole sees a very highly integrated time saving construct and is compatible. This being software is not cost effective but the developers who want to use these services need to subscribe with a nominal charges.

What Is an Example of an API?

When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

To explain this better, let us take a familiar example.

Imagine you're sitting at a table in a restaurant with a menu of choices to order from. The kitchen is the part of the "system" that will prepare your order. What is missing is the critical link to communicate your order to the kitchen and deliver your food back to your table. That's where the waiter or API comes in. The waiter is the messenger – or API – that takes your request or order and tells the kitchen – the system – what to do. Then the waiter delivers the response back to you; in this case, it is the food.

Here is a real-life API example. You may be familiar with the process of searching flights online. Just like the restaurant, you have a variety of options to choose from, including different cities, departure and return dates, and more. Let us imagine that you're booking you are flight on an airline website. You choose a departure city and date, a return city and date, cabin class, as well as other variables. In order to book your flight, you interact with the airline's website to access their database and see if any seats are available on those dates and what the costs might be.

However, what if you are not using the airline's website—a channel that has direct access to the information? What if you are using an online travel service, such as Kayak or Expedia, which aggregates information from a number of airline databases?

The travel service, in this case, interacts with the airline's API. The API is the interface that, like your helpful waiter, can be asked by that online travel service to get information from the airline's database to book seats, baggage options, etc. The API then takes the airline's response to your request and delivers it right back to the online travel service, which then shows you the most updated, relevant information.

What an API Also Provides Is a Layer of Security

Your phone's data is never fully exposed to the server, and likewise the server is never fully exposed to your phone. Instead, each communicates with small packets of data, sharing only that

which is necessary—like ordering takeout. You tell the restaurant what you would like to eat, they tell you what they need in return and then, in the end, you get your meal.

APIs have become so valuable that they comprise a large part of many business' revenue. Major companies like Google, eBay, Salesforce.com, Amazon, and Expedia are just a few of the companies that make money from their APIs. What the "API economy" refers to is this marketplace of APIs.

The Modern API

Over the years, what an "API" is has often described any sort of generic connectivity interface to an application. More recently, however, the modern API has taken on some characteristics that make them extraordinarily valuable and useful:

- Modern APIs adhere to standards (typically HTTP and REST), that are developer-friendly, easily accessible and understood broadly
- They are treated more like products than code. They are designed for consumption for specific audiences (e.g., mobile developers), they are documented, and they are versioned in a way that users can have certain expectations of its maintenance and lifecycle.
- Because they are much more standardized, they have a much stronger discipline for security and governance, as well as monitored and managed for performance and scale
- As any other piece of productized software, the modern API has its own software development lifecycle (SDLC) of designing, testing, building, managing, and versioning. Also, modern APIs are well documented for consumption and versioning.

Analysis Report

4. ANALYSIS REPORT

4.1 SRS (Software Requirements Specification) DOCUMENT

The document is prepared keeping is view of the academic constructs of my Bachelors Degree from university as partial fulfillment of my academic purpose the document specifies the general procedure that that has been followed by me, while the system was studied and developed. The general document was provided by the industry as a reference guide to understand my responsibilities in developing the system, with respect to the requirements that have been pin pointed to get the exact structure of the system as stated by the actual client.

The system as stated by my project leader the actual standards of the specification were desired by conducting a series of interviews and questionnaires. The collected information was organized to form the specification document and then was modeled to suite the standards of the system as intended.

Document Conventions:

The overall documents for this project use the recognized modeling standards at the software industries level.

1. The Physical dispense, which state the overall data search for the relational key whereas a transaction is implemented on the wear entities.
2. Unified modeling language concepts to give a generalized blue print for the overall system.
3. The standards of flow charts at the required states that are the functionality of the operations need more concentration.

4.2 SCOPE OF DEVELOPMENT

Future scope:

This project has scope for improvement and many enhancements can be done to make it more reliable and interesting

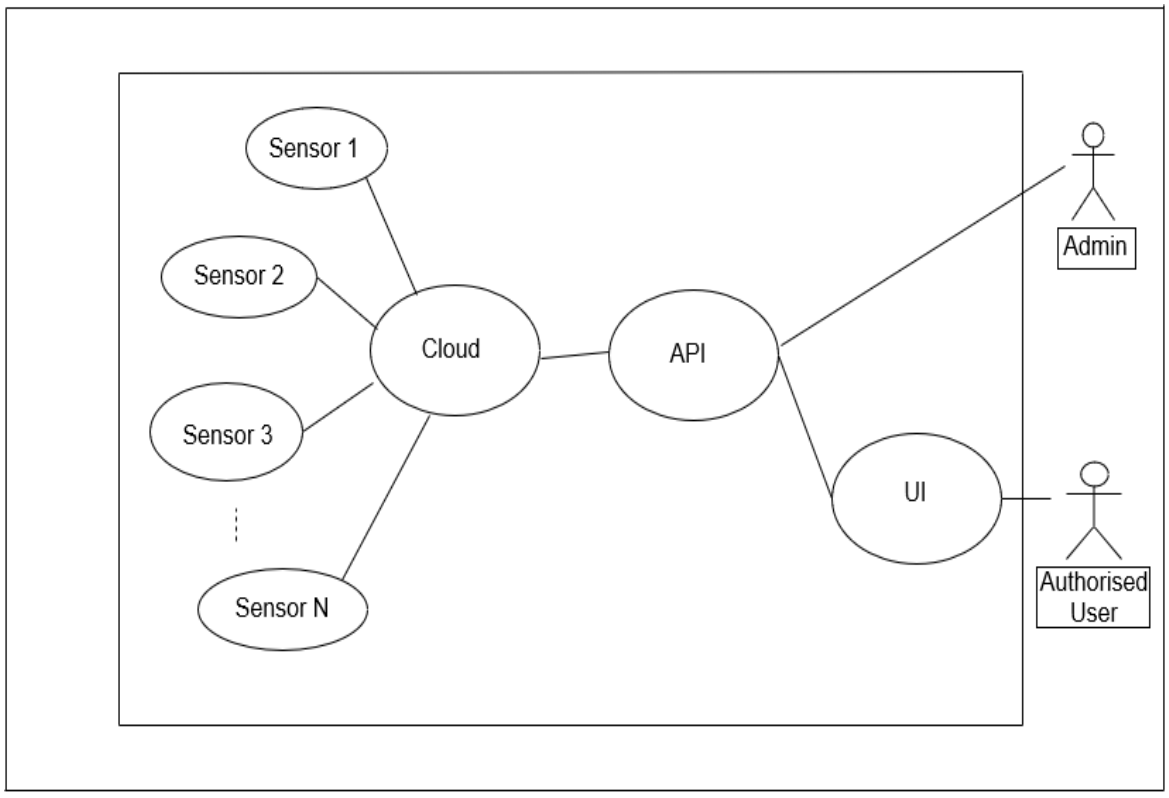
Design Document

5. Design Document

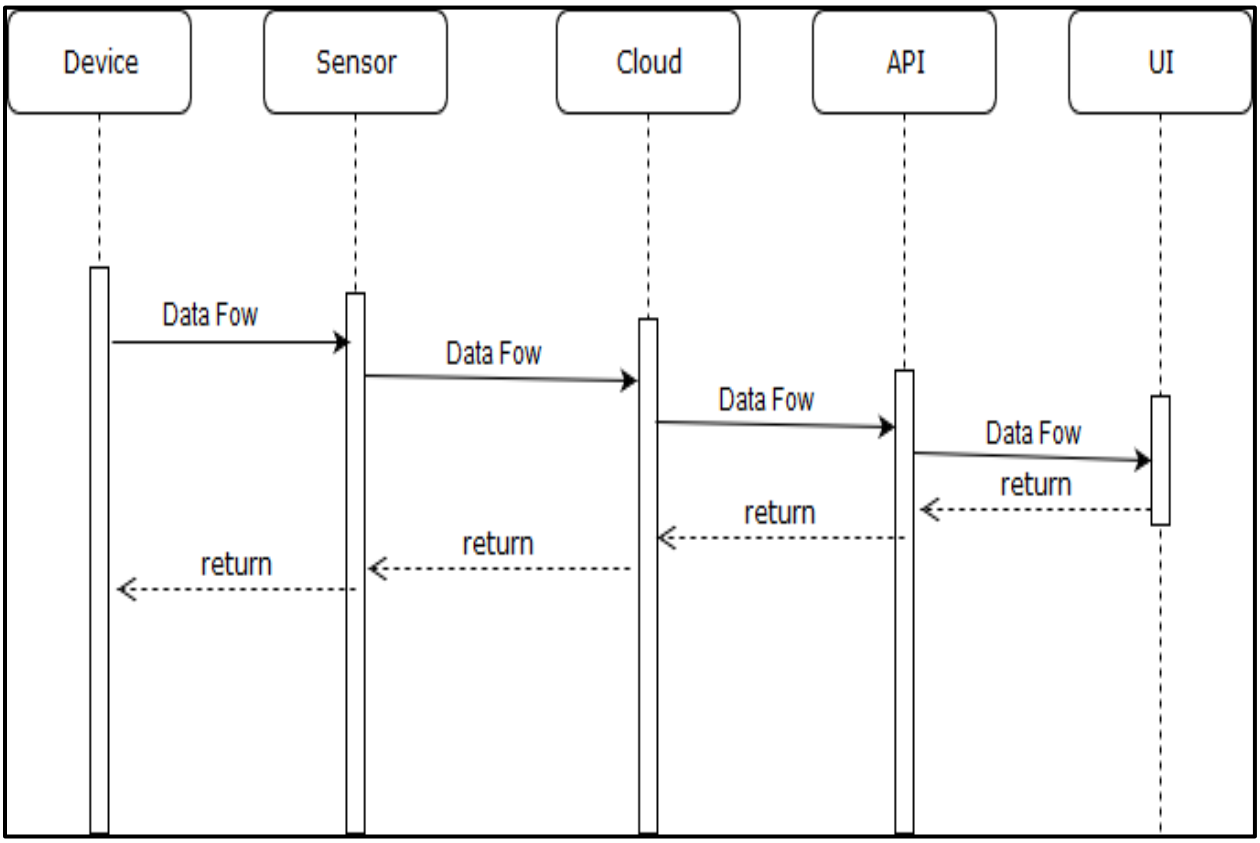
5.1 Unified Modeling Language Specifications

UML:

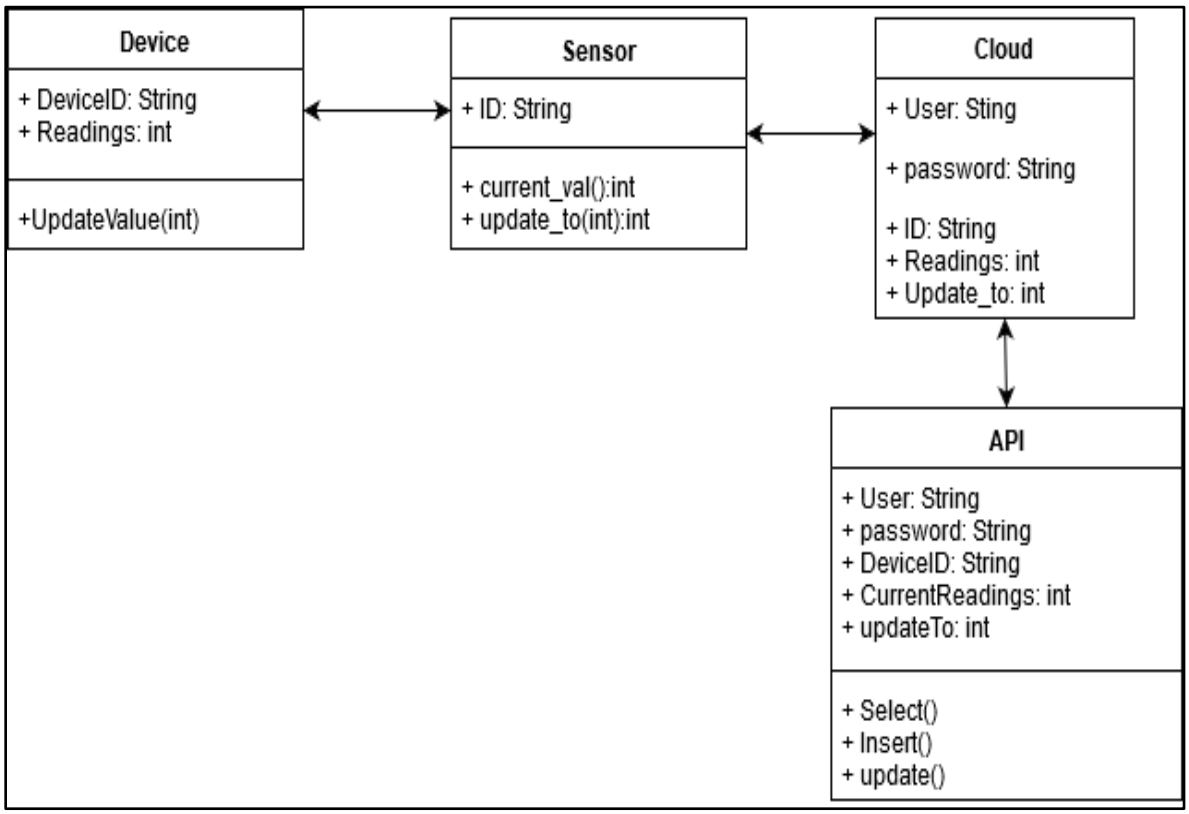
USE CASE DIAGRAM:



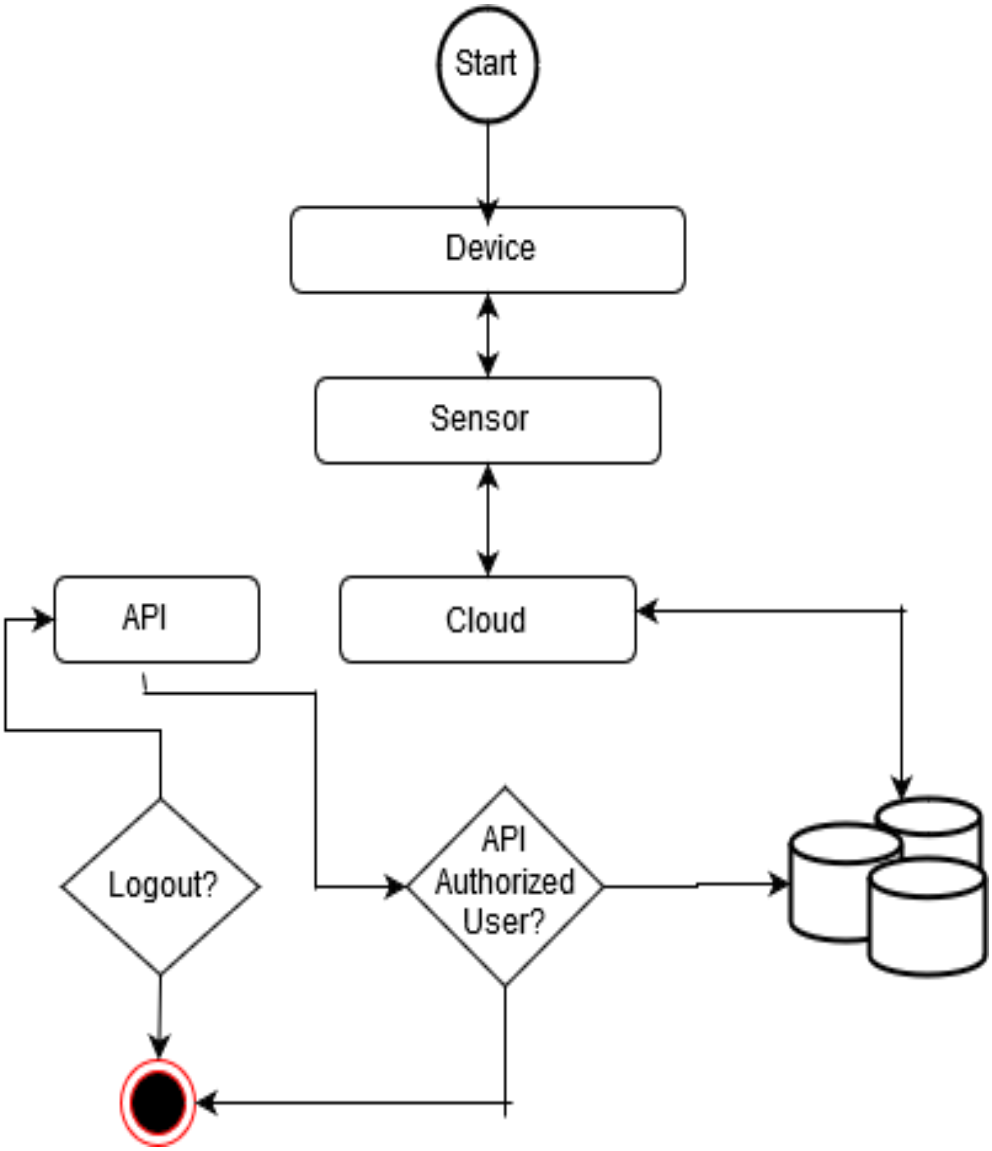
5.2. SEQUENCE DIAGRAM:



5.3 CLASS DIAGRAM:



5.4 ACTIVITY DIAGRAM:



Coding

6. CODING

Python API:

```
import urllib.request


class gvpIotApi:
    host="https://parna-tech-gvp.000webhostapp.com"
    url="/api/"
    rowCount=0
    access=False
    table=list(list())
    def __init__(self):
        try:
            urllib.request.urlopen(self.host, timeout=1)
            self.access=True
        except urllib.error.HTTPError as err:
            self.access=False
        if self.access:
            print("have a internet and host")
        else:
            print("don't have a internet or host")
    def select(self,api,deviceID,w_data=None):
        if not self.access:
```

```
                print("don't have a internet or host")
                return
        f=""
        if w_data is None:
                f =
urllib.request.urlopen(self.host+self.url+"select.php?api="+api+"&id="+deviceID)
        else:
                f =
urllib.request.urlopen(self.host+self.url+"select.php?api="+api+"&id="+deviceID+"&w_
data="+w_data)
        responce=str(f.read())
        data=responce.split('$')
        self.rowCount=int(data[1])
        data=data[2:len(data)-1]
        for i in data:
                self.table.append(i.split(","))

    def update(self,api,deviceID,data,w_data=None):
        if not self.access:
                print("don't have a internet or host")
                return
        f=""
        if w_data is None:
```

```
                f =
urllib.request.urlopen(self.host+self.url+"update.php?api="+api+"&id="+deviceID+"&da
ta="+data)
            else:
                f =
urllib.request.urlopen(self.host+self.url+"update.php?api="+api+"&id="+deviceID+"&da
ta="+data+"&w_data="+w_data)
            responce=str(f.read())
            data=responce.split('$')
            return data[1]
        def insert(self,api,deviceID,data):
            if not self.access:
                print("don't have a internet or host")
                return
            f=""
            f =
urllib.request.urlopen(self.host+self.url+"insert.php?api="+api+"&id="+deviceID+"&dat
a="+data)
            responce=str(f.read())
            data=responce.split('$')
            return data[1]
```

Cpp API:

```
#include <ESP8266WiFi.h>

class gvpIotApi{
  public: String ssid="",password="",host="parna-tech-
gvp.000webhostapp.com",url="/api/";
  public: WiFiClient client;
  public: int rowCount=0;
  gvpIotApi(String ssid,String password){
    this->ssid=ssid;
    this->password=password;
    Serial.begin(115200);
    delay(20);
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
```

```
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
 }

 public:void select(String api,String deviceID,String w_data=""){
  String url_temp="";
  Serial.print("connecting to ");
  Serial.println(host);
  if (!client.connect(host, 80)) {
   Serial.println("connection failed");
   return;
  }
  if(w_data.equals("")){
   url_temp=url+"select.php?api="+api+"&id="+deviceID;
  }else{
   url_temp=url+"select.php?api="+api+"&id="+deviceID+"&w_data="+w_data;
  }
  Serial.print("Requesting URL: ");
  Serial.println(url);
  client.print(String("GET ") + url_temp +" HTTP/1.1\r\n" +"Host: " + host + "\r\n"
+"Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  while (client.available() == 0) {
   if (millis() - timeout > 5000) {
```

```
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }
  String line="";
  if(client.available())
  line= client.readStringUntil('$');
  if(client.available())
  line = client.readStringUntil('$');
  rowCount=line.toInt()+1;
  /*int c=0;
  while(client.available()){
    String line = client.readStringUntil('$');
    if(c>=1){
      Serial.println(line);
    }
    c++;
  }*/
}

public: String getNextLine(){

    if(client.available()){
```

```
      String line = client.readStringUntil('$');
      return line;
    }
    return "";
  }

  public:int update(String api,String deviceID,String data,String w_data=""){
    String url_temp="";
    Serial.print("connecting to ");
    Serial.println(host);
    if (!client.connect(host, 80)) {
      Serial.println("connection failed");
      return -1;
    }
    if(w_data.equals("")){
      url_temp=url+"update.php?api="+api+"&id="+deviceID+"&data="+data;
    }else{

url_temp=url+"update.php?api="+api+"&id="+deviceID+"&data="+data+"&w_data="+
w_data;
    }
    Serial.print("Requesting URL: ");
    Serial.println(url);
```

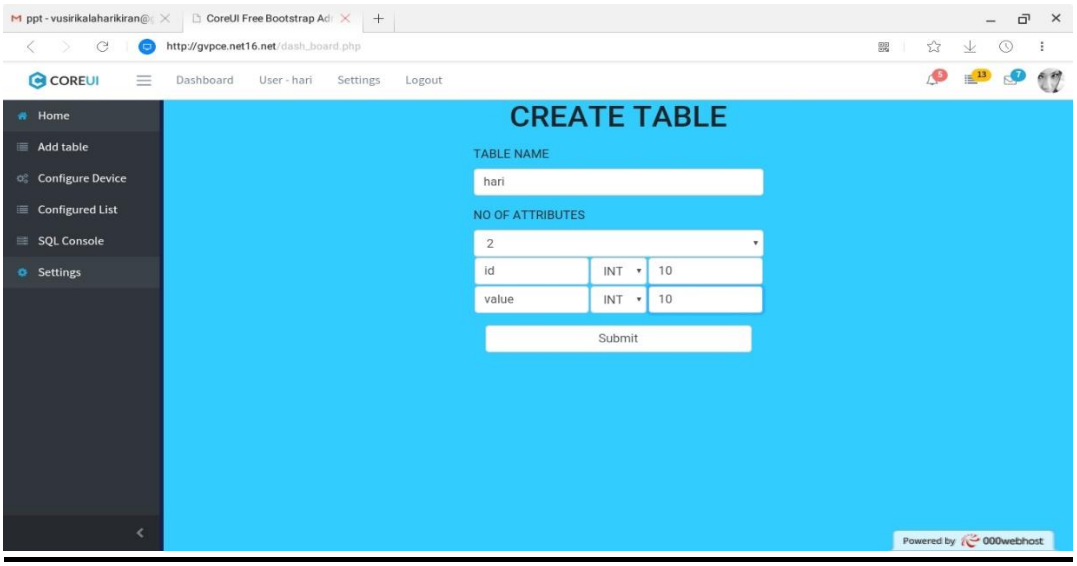
```
    client.print(String("GET ") + url_temp +" HTTP/1.1\r\n" +"Host: " + host + "\r\n"
+"Connection: close\r\n\r\n");
    unsigned long timeout = millis();
    while (client.available() == 0) {
      if (millis() - timeout > 5000) {
        Serial.println(">>> Client Timeout !");
        client.stop();
        return -2;
      }
    }
    String line="";
    if(client.available())
    line= client.readStringUntil('$');
    if(client.available())
    line = client.readStringUntil('$');
    int status=line.toInt();
    return status;
  }
  public:int insert(String api,String deviceID,String data){
    String url_temp="";
    Serial.print("connecting to ");
    Serial.println(host);
    if (!client.connect(host, 80)) {
      Serial.println("connection failed");
```

```
    return -1;
  }
  url_temp=url+"insert.php?api="+api+"&id="+deviceID+"&data="+data;
  Serial.print("Requesting URL: ");
  Serial.println(url);
  client.print(String("GET ") + url_temp +" HTTP/1.1\r\n" +"Host: " + host + "\r\n"
+"Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return -2;
    }
  }
  String line="";
  if(client.available())
  line= client.readStringUntil('$');
  if(client.available())
  line = client.readStringUntil('$');
  int status=line.toInt();
  return status;
 }
```

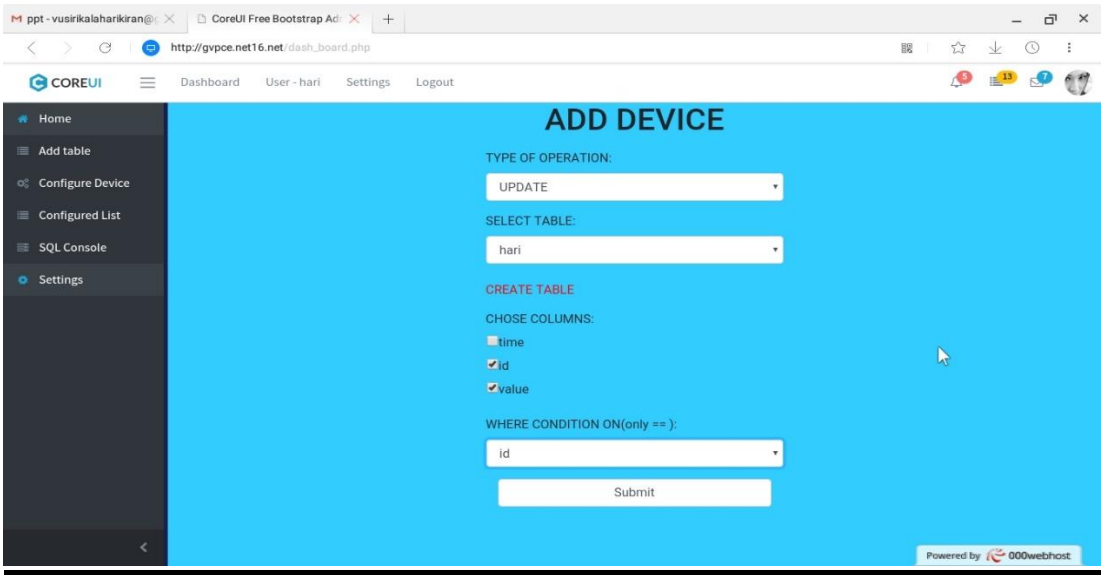
};

OUTPUT:

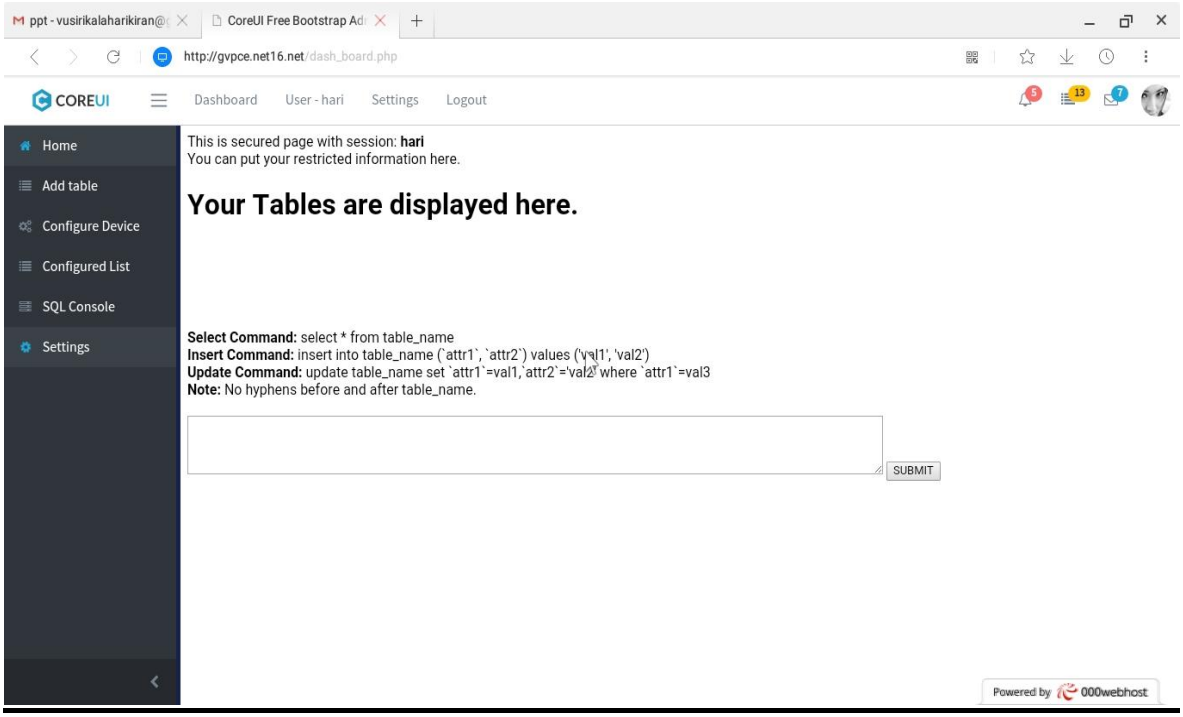
Create Table:



Add Devices



SQL Console:



Testing

7. TESTING

Testing:

The testing of API involves its compatibility to any IoT device

Purpose of Testing:

The Purpose of testing is to find whether are not exact sensor values are retrieved.

Testing Objectives:

To make the API to work for all IoT devices.

7.1 Levels of Testing

In order to uncover the errors, present in different phases we have the concept of levels of testing. The basic levels of testing are as shown below…

Code Testing

The code is tested against IoT devices and sensors giving a bug free and accurate results.

C O N C L U S I O N

8.CONCLUSION

- This API facilitates a flexibility for friendly environment
- Mutual communication between sensors and user interface.
- Acting as a generic API for all IoT devices.

B I B L I O G R A P H Y

9. BIBLIOGRAPHY

9.1 REFERENCES

- "APIs: A Strategy Guide" Book by Dan Woods, Daniel Jacobson, and Greg Brail.
- "Raspberry Pi For Dummies" Book by Mike Cook and Sean McManus

9.2 LIST OF WEBSITES:

- https://www.arduino.cc/en/Main/Docs
- https://pythonprogramming.net/introduction-raspberry-pi-tutorials/
- https://www.tutorialspoint.com/rest_api/index.asp
- https://maker.pro/esp8266/tutorial/how-to-program-esp8266s-onboard-gpio-pins
- https://www.mulesoft.com/resources/api/what-is-an-api