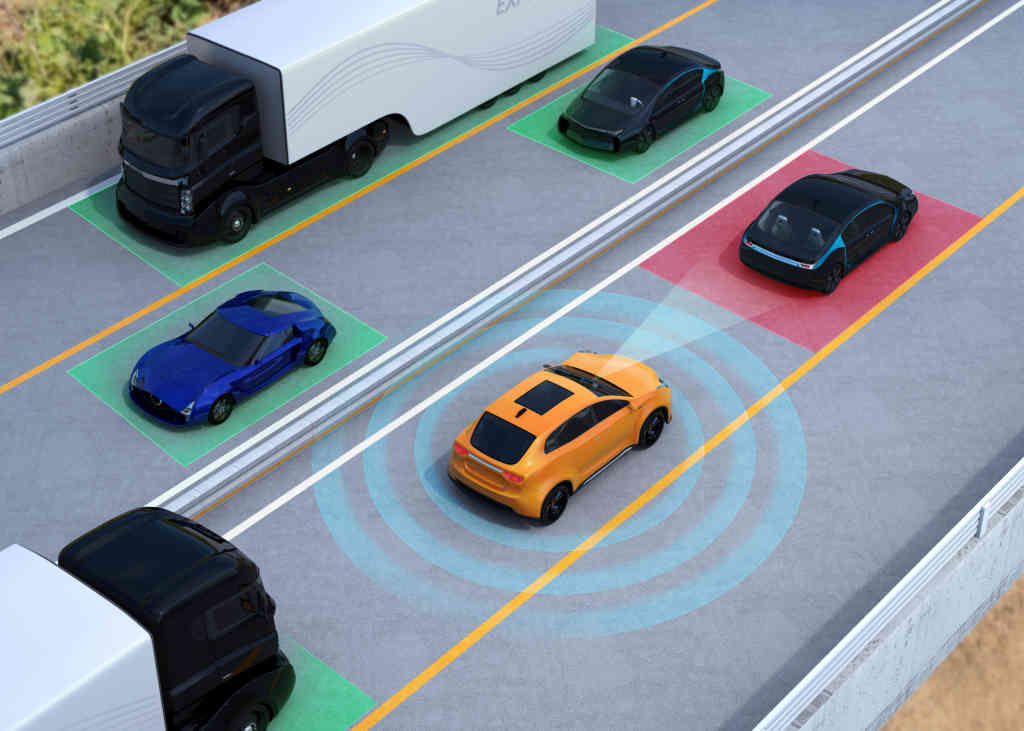
**SMART TRANSPORTATION SYSTEM**



TEAM NAME: TECH DUDES

DONE BY: SHIVA KRISHNA

ASWINI REDDY

SAI CHANDANA

ABSTRACT

Big data analytics helps in analyzing a huge set of data whereas IoT is about data, devices and connectivity. Internet of Things (IoT) involves connecting physical objects to the Internet to build smart systems and universal mobile accessibility advanced technologies like Intelligent Transportation System (ITS). IoT solutions are playing a major role in driving the global IoT in Intelligent Transportation System. Communication between vehicles using IoT will be a new era of communication that leads to ITS. IoT is a combination of storing and processing sensor data and computing using data analytics to achieve and assist in managing the Traffic system effectively. IoT based Intelligent transportation system (IoT-ITS) helps in automating railways, roadways, airways and marine which enhance customer experience about the way goods are transported, tracked and delivered. A case study on Intelligent Traffic Management System based on IoT and big data, which will be a part of, smart traffic solutions for smarter cities. The ITS-IoT system itself forms an eco-system comprising of sensor systems, monitoring system and display system. There are several techniques and algorithms involved in full functioning of IoT-ITS. The proposed case study will examine and explain a complete design and implementation of a typical IoT-ITS system for a smart city scenario set on typical Indian subcontinent. This case study will also explain about several hardware and software components associated with the system. How concepts like Multiple regression analysis, Multiple discriminant analysis and logistic regression, Co-joint analysis, Cluster analysis and other big data analytics techniques will merge with IoT and help to build IoT-ITS will also be emphasized. The case study will also display some big data analytics results and how the results are utilized in smart transportation systems.

INTRODUCTION

Transportation is an important factor that affects the quality of life. Its development is regarded as the most significant driver for social progress. Transport network connects cities, nations, manufacturers, and retailers. Transportation enables mobility, allows people to interact, and facilitates the growth of economy since it enables delivery of goods and services around the world. Modes of transportation include air, sea, and land. “To accommodate increasing transportation demands and provide safe and efficient travel in U.S. communities, it is estimated the nation’s highway and bridges will need $290 billion in investments, transit systems $18 billion and airports $50 billion over the next five years” . A number of cities have started using smart transportation as a solution to transportation-related problems. Smart transportation (and its cousin, intelligent transportation system) offers a means of providing innovative services on different modes of transportation and traffic management. It is an important area in the smart grid and an extension of smart cities. Its components include infrastructure, vehicles, and users.

**HARDWARE COMPONENTS:**

* NODE MCU
* GPS MODULE

**NODE MCU:**



NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board.

Since NodeMCU is open source platform, their hardware design is open for edit/modify/build.

NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The ESP8266  is a low-cost [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) chip developed by Espress ,if Systems with TCP/IP protocol. For more information about ESP8266, you can refer [ESP8266 WiFi Module](http://www.electronicwings.com/sensors-modules/esp8266-wifi-module).

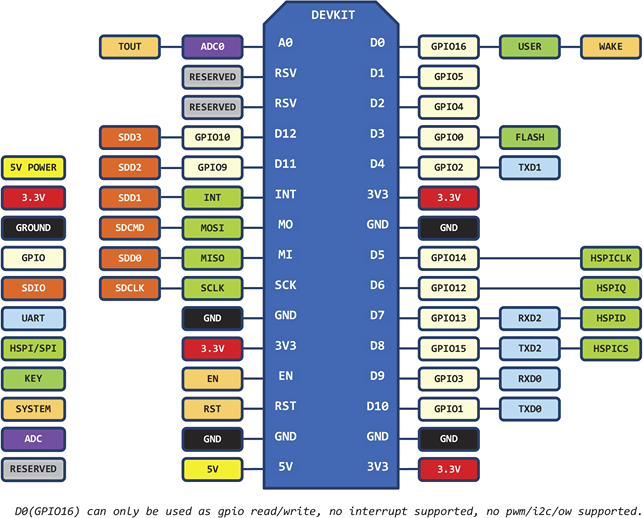
There is Version2 (V2) available for NodeMCU Dev Kit i.e. **NodeMCU Development Board v1.0 (Version2),** which usually comes in black colored PCB. NodeMCU Dev Kit has **Arduino like**Analog (i.e. A0) and Digital (D0-D8) pins on its board.

It supports serial communication protocols i.e. UART, SPI, I2C etc.

Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

**SPECIFICATIONS:**

* Wi-Fi Module – ESP-12E module similar to [ESP-12](https://www.aliexpress.com/item/new-esp8266-ESP-12-wifi-module-ESP8266-serial-WIFI-coexistence-module-AP-STA-AP-STA-WIFI/32239125397.html) module but with 6 extra GPIOs.
* USB – micro USB port for power, programming and debugging
* Headers – 2x 2.54mm 15-pin header with access to GPIOs, SPI, UART, ADC, and power pins
* Misc – Reset and Flash buttons
* Power – 5V via micro USB port
* Dimensions – 49 x 24.5 x 13mm



**GPS MODULE(UBLOX):**



This is a complete GPS module that is based on the Ublox NEO-6M. This unit uses the latest technology from Ublox to give the best possible positioning information and includes a larger built-in 25 x 25mm active GPS antenna with a UART TTL socket. A battery is also included so that you can obtain a GPS lock faster. This is an updated GPS module that can be used with ardupilot mega v2. This GPS module gives the best possible position information, allowing for better performance with your Ardupilot or other Multirotor control platform.

The Ublox NEO-6M GPS engine on this board is a quite good one, with the high precision binary output. It has also high sensitivity for indoor applications. UBLOX NEO-6M GPS Module has a battery for power backup and EEPROM for storing configuration settings. The antenna is connected to the module through a ufl cable which allows for flexibility in mounting the GPS such that the antenna will always see the sky for best performance. This makes it powerful to use with cars and other mobile applications.

The Ublox GPS module has serial TTL output, it has four pins: TX, RX, VCC, and GND. You can download the u-center software for configuring the GPS and changing the settings and much more. It is really good software.

**Features:**

1. 5Hz position update rate
2. Operating temperature range: -40 TO 85°CUART TTL socket
3. EEPROM to save configuration settings
4. Rechargeable battery for Backup
5. The cold start time of 38 s and Hot start time of 1 s
6. Supply voltage: 3.3 V
7. Configurable from 4800 Baud to 115200 Baud rates. (default 9600)
8. SuperSense Indoor GPS: -162 dBm tracking sensitivity
9. Support SBAS (WAAS, EGNOS, MSAS, GAGAN)
10. Separated 18X18mm GPS antenna

**PIN DESCRIPTION:**

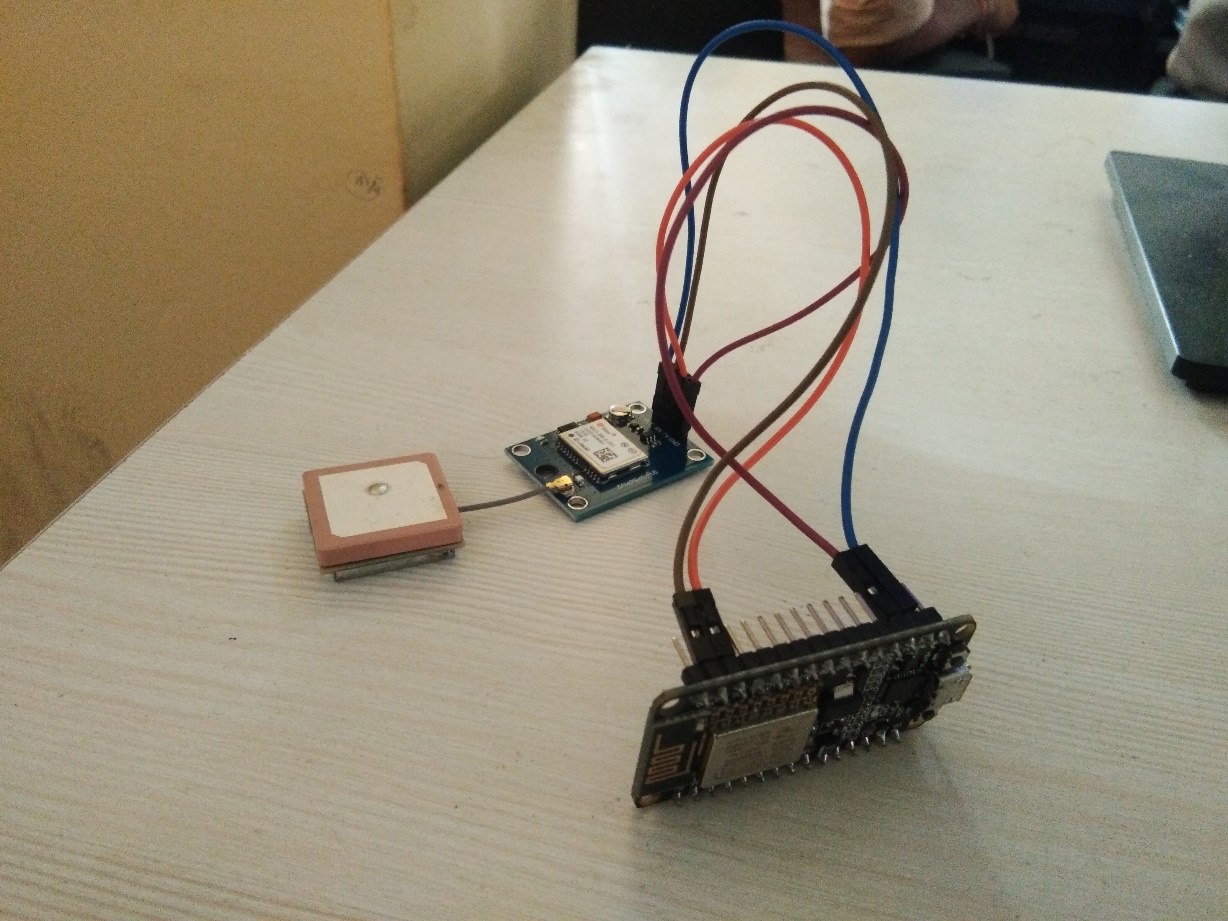
**VCC:** Power Supply 3.3 – 6 V

**GND:**Ground

**TX:**Transmit data serially which gives information about location, time etc.

**RX:**Receive Data serially. It is required when we want to configure GPS module.

**CONNECTIONS:**



**SOFTWARE PROGRAM:**

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

TinyGPSPlus gps; // The TinyGPS++ object

SoftwareSerial ss(D2,D1 ); // The serial connection to the GPS device

const char\* ssid = "vamshi";

const char\* password = "12345678";

float latitude , longitude;

int year , month , date, hour , minute , second;

String date\_str , time\_str , lat\_str, lng\_str;

int pm;

#define ORG "a41w0q"

#define DEVICE\_TYPE "chandana"

#define DEVICE\_ID "187Y1A1239"

#define TOKEN "chandana777"

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/evt/Data/fmt/json";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

PubSubClient client(server, 1883,wifiClient);

//WiFiServer server1(80);

void setup()

{

Serial.begin(115200);

ss.begin(9600);

Serial.println();

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");

// server.begin();

// Serial.println("Server started");

// Print the IP address

Serial.println(WiFi.localIP());

}

void loop()

{

while (ss.available() > 0)

if (gps.encode(ss.read()))

{

if (gps.location.isValid())

{

latitude = gps.location.lat();

lat\_str = String(latitude , 6);

Serial.println(lat\_str);

longitude = gps.location.lng();

lng\_str = String(longitude , 6);

Serial.println(lng\_str);

}

if (gps.date.isValid())

{

date\_str = "";

date = gps.date.day();

month = gps.date.month();

year = gps.date.year();

if (date < 10)

date\_str = '0';

date\_str += String(date);

date\_str += " / ";

if (month < 10)

date\_str += '0';

date\_str += String(month);

date\_str += " / ";

if (year < 10)

date\_str += '0';

date\_str += String(year);

}

if (gps.time.isValid())

{

time\_str = "";

hour = gps.time.hour();

minute = gps.time.minute();

second = gps.time.second();

minute = (minute + 30);

if (minute > 59)

{

minute = minute - 60;

hour = hour + 1;

}

hour = (hour + 5) ;

if (hour > 23)

hour = hour - 24;

if (hour >= 12)

pm = 1;

else

pm = 0;

hour = hour % 12;

if (hour < 10)

time\_str = '0';

time\_str += String(hour);

time\_str += " : ";

if (minute < 10)

time\_str += '0';

time\_str += String(minute);

time\_str += " : ";

if (second < 10)

time\_str += '0';

time\_str += String(second);

if (pm == 1)

time\_str += " PM ";

else

time\_str += " AM ";

}

}

// Check if a client has connected

//WiFiClient client = server.available();

PublishData(lat\_str,lng\_str);

if (!client.loop())

{

mqttConnect();

return;

}

// Prepare the response

String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n <!DOCTYPE html> <html> <head> <title>GPS Interfacing with NodeMCU</title> <style>";

s += "a:link {background-color: YELLOW;text-decoration: none;}";

s += "table, th, td {border: 1px solid black;} </style> </head> <body> <h1 style=";

s += "font-size:300%;";

s += " ALIGN=CENTER> GPS Interfacing with NodeMCU</h1>";

s += "<p ALIGN=CENTER style=""font-size:150%;""";

s += "> <b>Location Details</b></p> <table ALIGN=CENTER style=";

s += "width:50%";

s += "> <tr> <th>Latitude</th>";

s += "<td ALIGN=CENTER >";

s += lat\_str;

s += "</td> </tr> <tr> <th>Longitude</th> <td ALIGN=CENTER >";

s += lng\_str;

s += "</td> </tr> <tr> <th>Date</th> <td ALIGN=CENTER >";

s += date\_str;

s += "</td></tr> <tr> <th>Time</th> <td ALIGN=CENTER >";

s += time\_str;

s += "</td> </tr> </table> ";

if (gps.location.isValid())

{

s += "<p align=center><a style=""color:RED;font-size:125%;"" href=""http://maps.google.com/maps?&z=15&mrt=yp&t=k&q=";

s += lat\_str;

s += "+";

s += lng\_str;

s += """ target=""\_top"">Click here!</a> To check the location in Google maps.</p>";

}

s += "</body> </html> \n";

// client.print(s);

delay(100);

}

void mqttConnect() {

if (!client.connected()) {

Serial.print("Reconnecting MQTT client to "); Serial.println(server);

while (!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

initManagedDevice();

Serial.println();

}

}

void initManagedDevice() {

if (client.subscribe(topic)) {

Serial.println("subscribe to cmd OK");

} else {

Serial.println("subscribe to cmd FAILED");

}

}

void PublishData(String lat\_str, String lng\_str){

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

Serial.println();

}

String payload = "{\"d\":{\"latitude\":";

payload += lat\_str;

payload+="," "\"longitude\":";

payload += lng\_str;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(topic, (char\*) payload.c\_str())) {

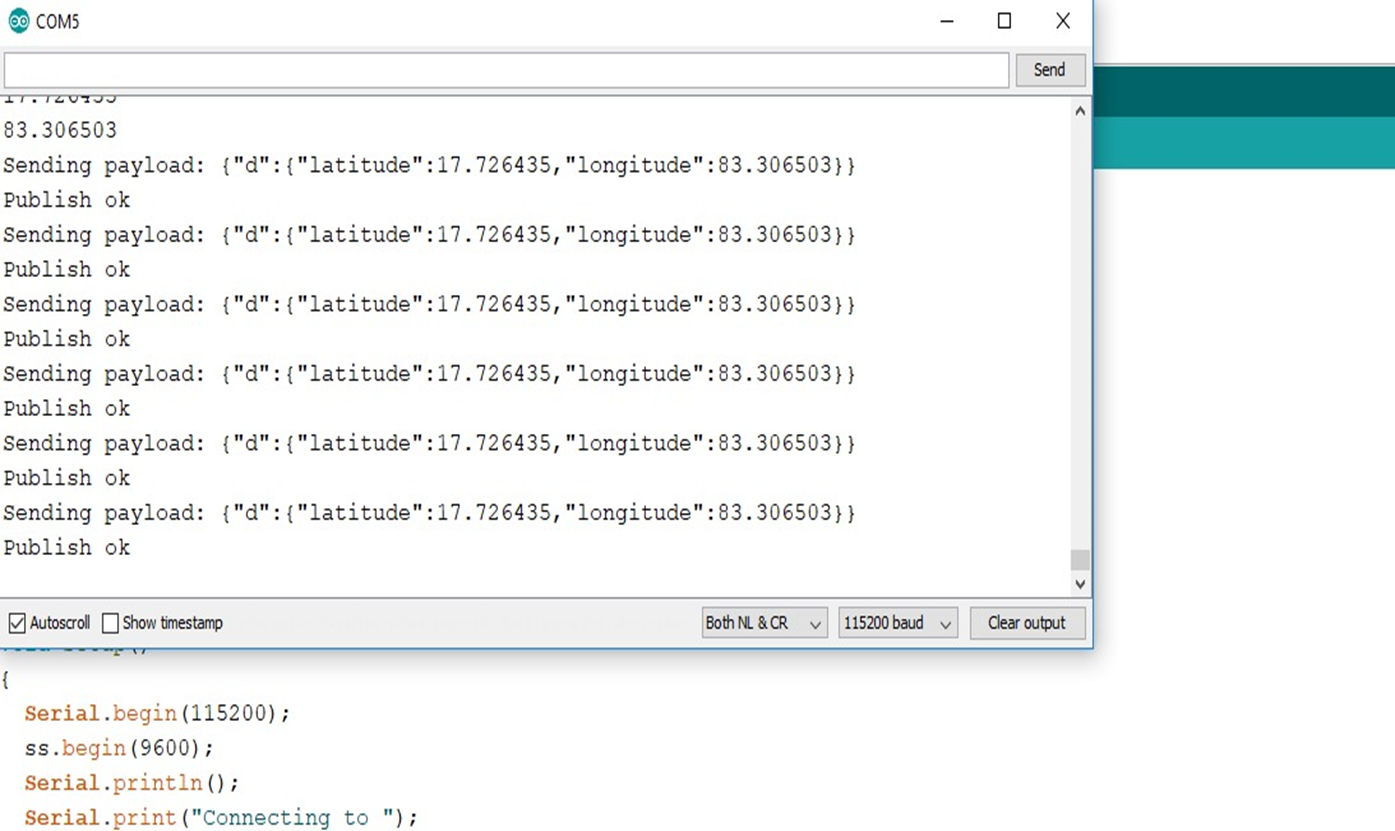
Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

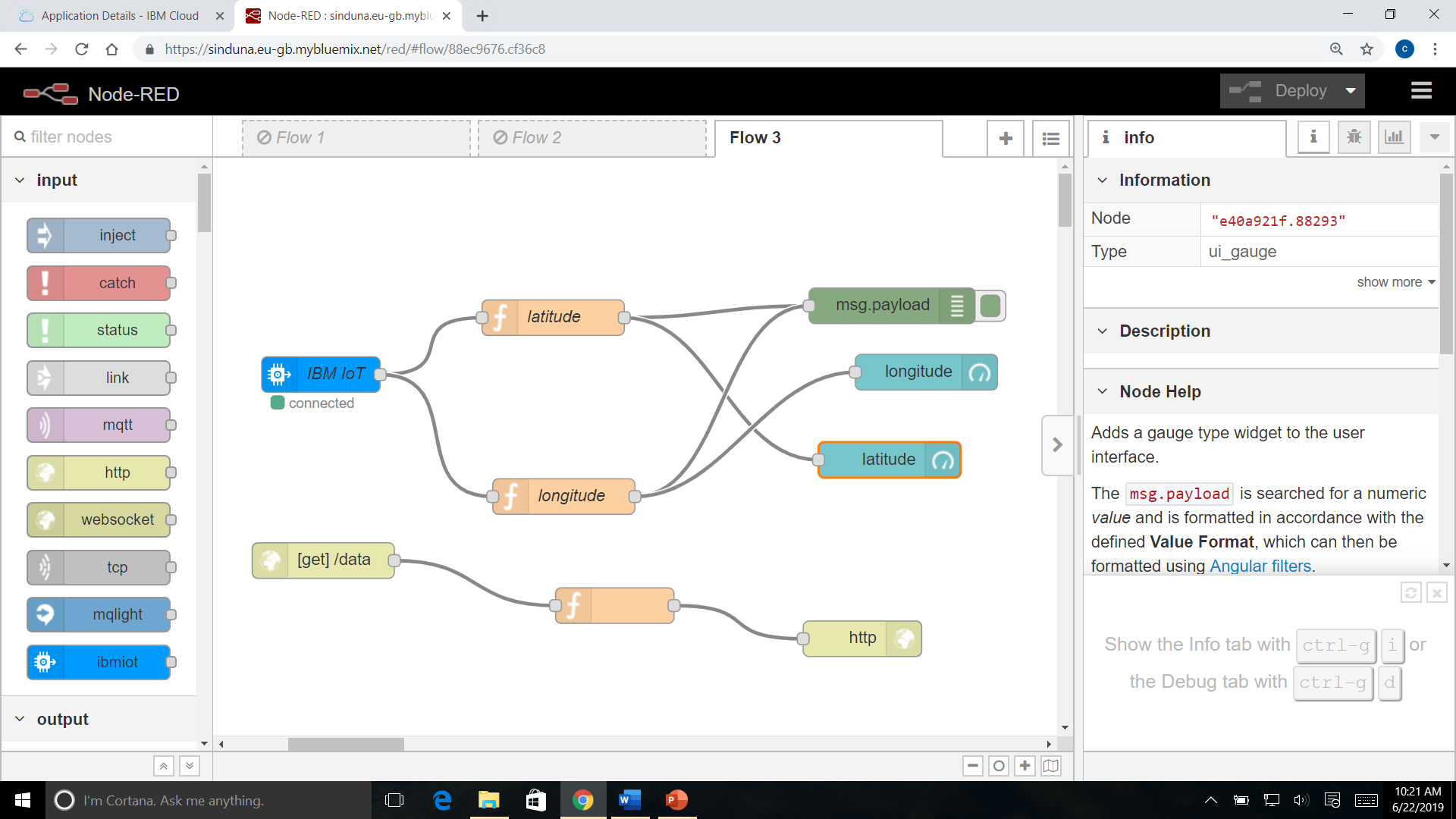
}

}

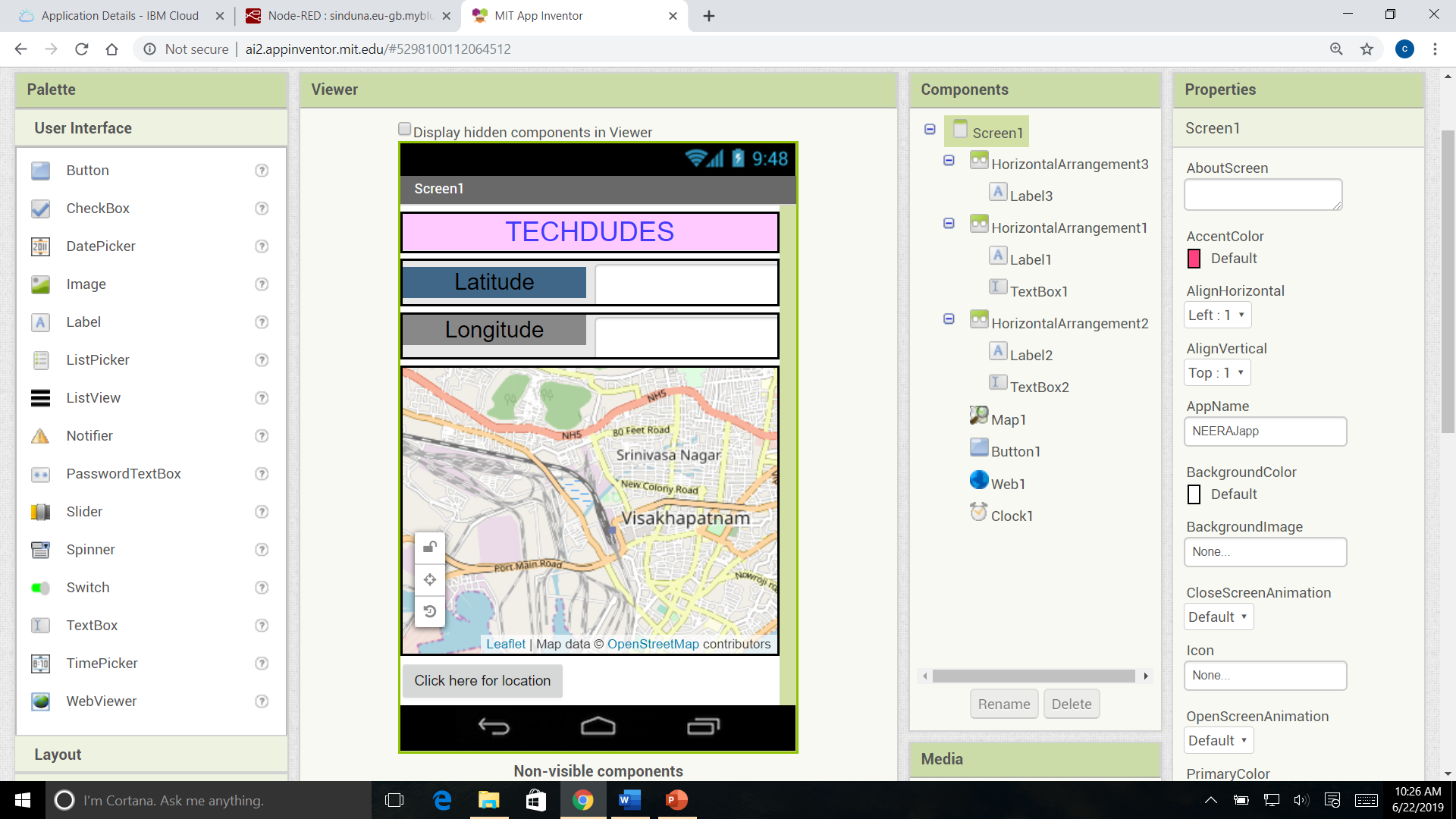


**FIGURE: SERIAL MONITOR OUTPUT**

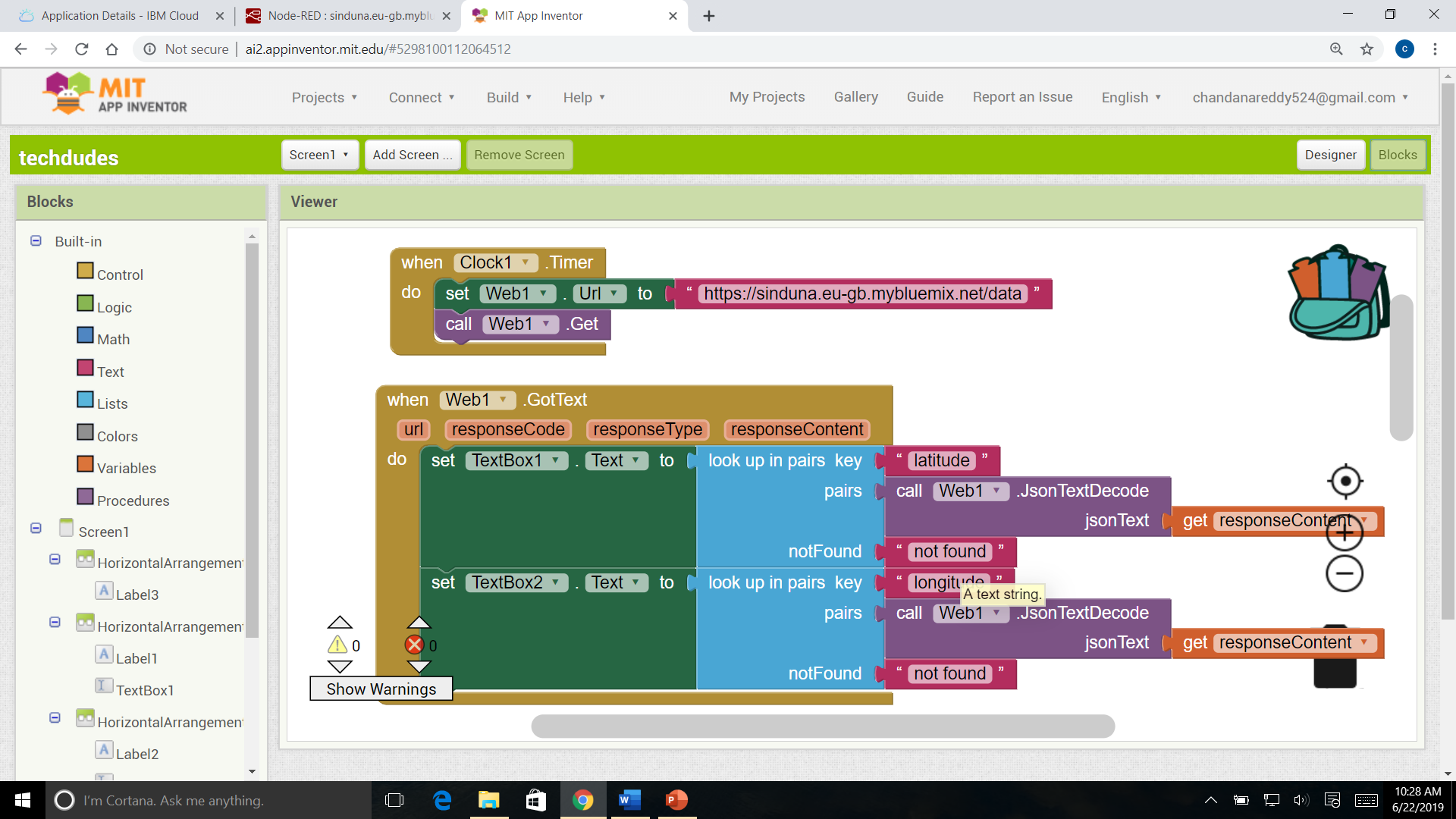
**NODE RED:**

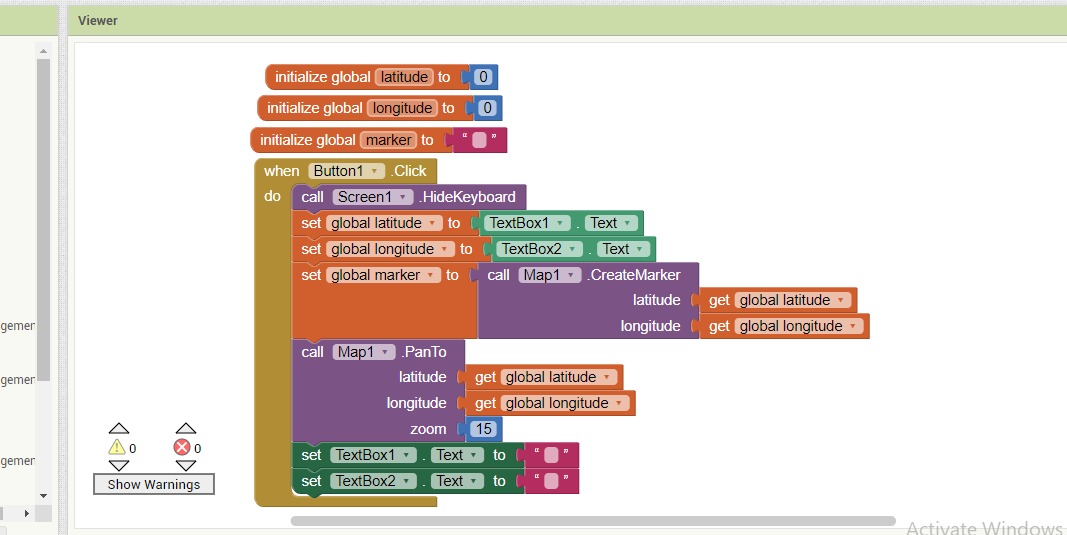


**MIT APP INVENTOR:**



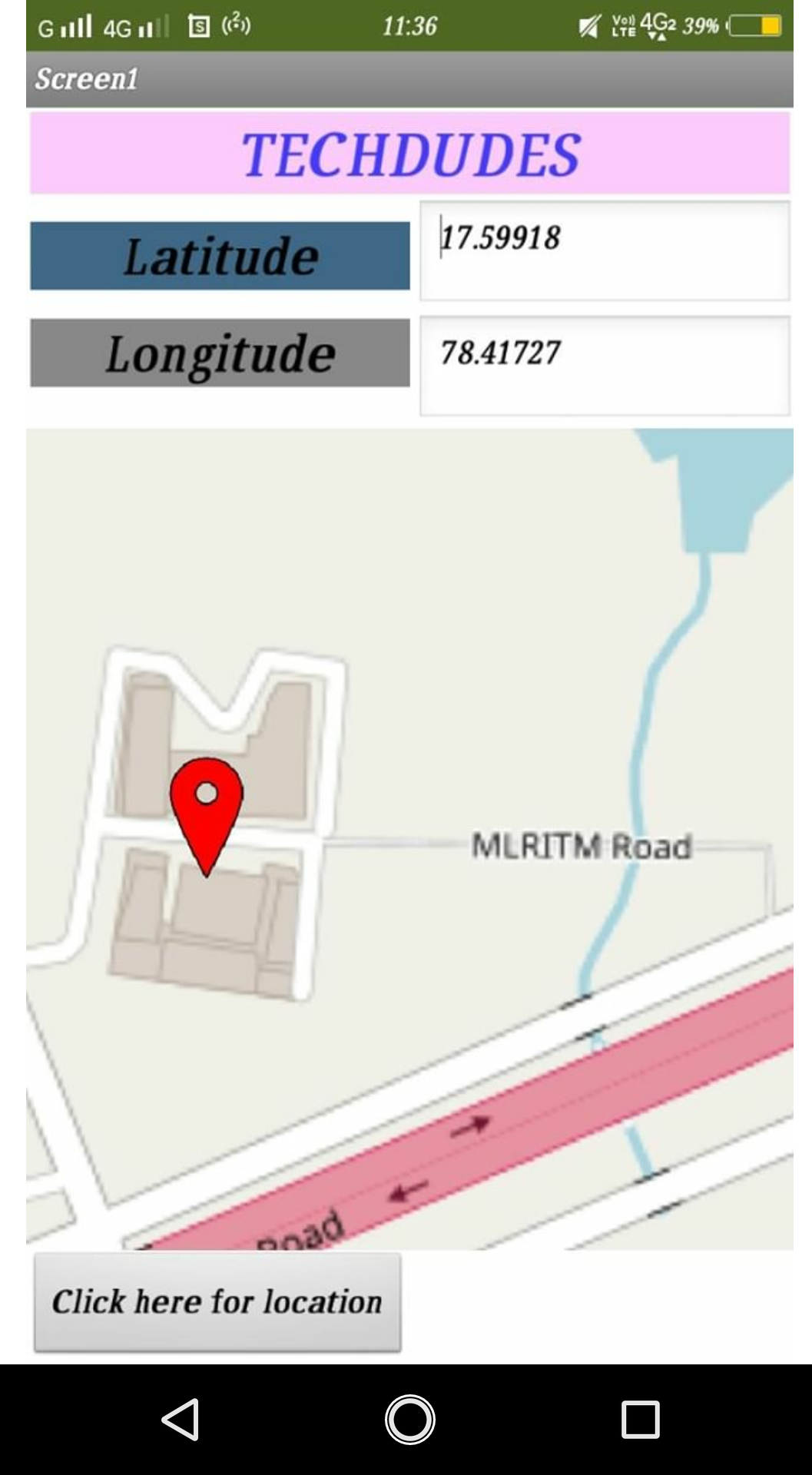
**FIGURE: MIT DESIGNER SECTION**

****



**FIGURE: MIT BLOCK SECTION**

**OUTPUT:**



**Conclusion :-**

In this project we are getting GPS location through app . Here GPS module send signals through transmitter to GPS satellite and get the location through receiver and it communicates with Node MCU through UART protocol and this received data is loaded into the cloud and app . By this are getting the live updates of our location and we can manipulate this data and used for many purposes.

**Future Scope :-**

This project has a huge future scope that we can implement Multimodal dynamic routing , Scheduling, Inteligent mobility information services for citizens, Innovative transportation solutions many.