# SMART TRANSPORTATION SYSTEM USING IBM WATSON

THIS PROJECT IS DONE BY:

* T. Sai Manasa 17251A0458
* A. Sai Ishwarya 17251A0401
* K. Bindhu Madhavi 17251A0407
* J. Madhu Priya 18255A0404

ABSTRACT

Geolocation is defined as the identification/estimation of the real-world geographic location of an object, such as a radar source, mobile phone, or Internet-connected computer terminal or any other device. Apparently, geolocation involves the generation of a set of geographic coordinates and is closely related to the use of positioning systems, it is enhanced by the use of these coordinates which determine a meaningful location, such as specific area. In this project, we will be interfacing a GPS module with NodeMCU. A simple local web server is created using NodeMCU and the location details are updated in that server webpage. NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 module and hardware which is based on the ESP-12 module. ESP8266 can be used for geolocation by firstly obtaining nearby AP properties, and then using Google geolocation API to locate the user-device. To be able to obtain a fix on the location of the device that integrates the ESP8266 chip, we assume that the host controller first could obtain data from nearby Wi-Fi networks or cellular sub-systems. The data is consolidated into a data block that must be sent to an online geolocation API or service that will estimate the device location in terms of latitude, longitude, and accuracy.

PROBLEM STATEMENT

* Integrating GPS Module with Node MCU using Arduino Programming.
* Connecting Node MCU with IBM Cloud using Arduino Programming.
* Creating a Node RED UI which displays location and vehicle data using Node Red Programming

COMPONENTS USED AND THEIR SPECIFICATIONS

Hardware

* Node MCU
* GPS module

Softwares

* Arduino IDE
* IBM Cloud

Hardware specification

1) NodeMCU

NodeMCU is an open source Lua based firmware designed for the ESP8266 WiFi SOC from Espressif and uses an on-module flash-based SPIFFS file system. It is implemented in C and is layered on the Espressif NON-OS SDK. The firmware was initially developed as a companion project to the popular ESP8266-based NodeMCU development modules, but the project is now community-supported, and the firmware can now run on any kind of ESP module.

* It consists of 13 GPIO pins(General Purpose Input andOutput)
* This board is 32 bit MCU
* Flash memory of 4MB
* SRAM of 128KB
* Crystal oscillator clock frequency of 80MHz
* ADC0 is an analog pin
* 13 GPIO pins are digital
* D2,D5,D6,D8 are analog output/PWM pin.It has reset and enable pins

2) GPS Module

The NEO-6 module series is a family of stand-alone GPS receivers featuring high-performance u-blox 6 positioning engine. These flexible and cost-effective receivers offer multiple connectivity options in a miniature 16 x 12.2 x 2.4 mm package. The compact architecture, power and memory options make the NEO-6 modules ideal for battery operated mobile devices with limited cost and space constraints. Innovative design and technology suppresses jamming sources and mitigates multipath effects which give NEO6 GPS receivers excellent navigation performance even in the most challenging environments.

* This module has an external antenna and built in EEPROM
* Interface:RS232 TTL
* Power supply:3V to 5V
* Default baudrate:9600bps
* Works with standard NMEA sentences
* It is also compatible with other micro controller boards

Basic principle of GPS:

The working/operation of Global positioning system is based on the ‘trilateration’ mathematical principle.

* The position is determined from the distance measurements to satellites. From the figure, the four satellites are used to determine the position of the receiver on the earth.
* The target location is confirmed by the 4th satellite. And three satellites are used to trace the location place. A fourth satellite is used to confirm the target location of each of those space vehicles.
* Global positioning system consists of satellite, control station and monitor station and receiver. The GPS receiver takes the information from the satellite and uses the method of triangulation to determine a user’s exact position.

WIFI

ARDUINO

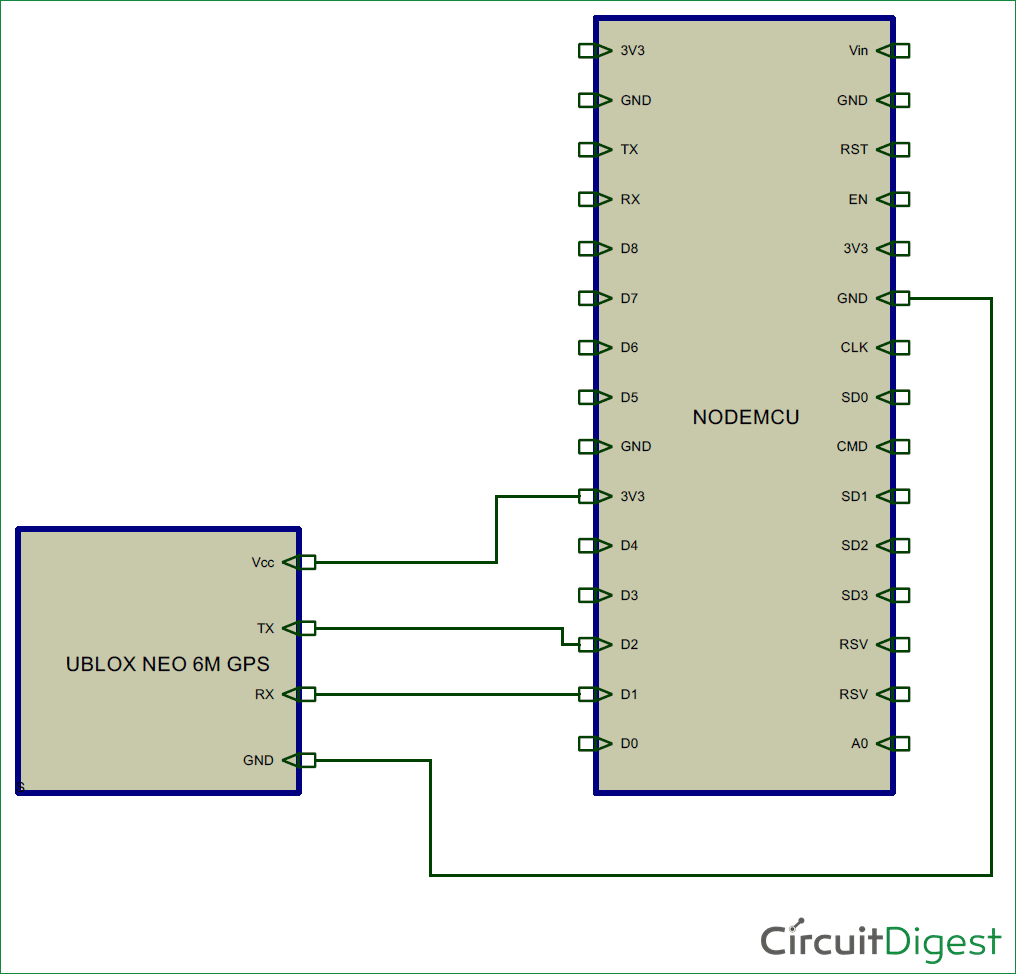
GPS MODULE

IBM CLOUD

NODEMCU

WORLD MAP

NODERED FLOW



Pin connections of GPS module to Node Mcu

FINAL CODE

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

#include<PubSubClient.h>

TinyGPSPlus gps; // The TinyGPS++ object

SoftwareSerial ss(D4, D5); // The serial connection to the GPS device

const char\* ssid = "dlink-gnits2g";

const char\* password ="dlink@gnits";

#define ORG "x9899"

#define DEVICE\_TYPE "mcfhbb"

#define DEVICE\_ID "1990"

#define TOKEN "980765343"

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);

float latitude , longitude;

String lat\_str,lng\_str;

void setup() {

Serial.begin(9600);

ss.begin(9600);

Serial.println();

Serial.print("Connecting to ");

Serial.print(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected, 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);

longitude = gps.location.lng();

lng\_str = String(longitude , 6);

}

}

PublishData(lat\_str,lng\_str);

delay(100);

}

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\":{\"lat\_str\":";

payload += lat\_str;

payload+="," "\"lng\_str\":";

payload +=lng\_str ;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

delay(500);

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

Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

}

}

PROJECT HIGHLIGHTS

* Working with IBM Watson cloud services
* Accessing Sensor data from anywhere in the world.
* Notifications and SMS are triggered if vehicle has reached particular location
* Message to the respective authority as soon as the package reaches specific location.
* Tracking of Vehicle Location is made much easier for the customer as well as the delivery authority.
* Improves Client Satisfaction, Driver safety .

INSTALLATIONS REQUIRED

#include <TinyGPS++.h>

#Include <SoftwareSerial.h>

#Include <ESP8266WiFi.h>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*