Miniproject based on MicroController NodeMcu

A Report on Miniproject Submitted for the requirement of

University of Mumbai

The practical work done during Semester-VI in

Miniproject II
(Electronics and Telecommunication Engineering)

by

Shubham Tawade (15ET1105)

Vishal Kumar Singh(15ET1138)

Riddesh Sonawane (15ET1043)

Abhishek Singh (15ET1107)

Under the Guidance of

Ms. Jyoti Vengurlekar

Department of Electronics and Telecommunication Engineering
Ramrao Adik Institute of Technology,
Sector 7, Nerul , Navi Mumbai
(Affiliated to University of Mumbai)
April 2018

Ramrao Adik Education Society's

Ramrao Adik Institute of Technology

(Affiliated to the University of Mumbai)

Dr. D. Y. Patil Vidyanagar,Sector 7, Nerul, Navi Mumbai 400706.

Certificate

This is to certify that, the Miniproject II titled

"Railway Track Fault Detection "

is a bonafide work done by

Shubham Tawade (15ET1105)
Vishal Kumar Singh (15ET1138)
Riddesh Sonawane (15ET1043)
Abhishek Singh (15ET1107)

and is submitted in the partial fulfillment of the requirement for the degree of

Third Year of Engineering

(Electronics and Telecommunication Engineering)

to the

University of Mumbai.

Project Guide Ms. Jyoti Vengurlekar	Project Coordinator Co-ordinator Name
Head of Department Dr. M. D. Patil	Principal Dr. Ramesh Vasappanavara

Certificate of Approval by Examiners

This is to certify that the submission entitled for the project

"Railway Track Fault Detection"

is a bonafide work done by **Shubham Tawade, Vishal Kumar Singh ,Riddesh Sonawane and Abhishek Singh** under the guidance of **Ms. Jyoti Vengurlekar** . This project work has been approved for semester VI in **Miniproject-II**, University of Mumbai.

Examiners:

Internal Examiner:	External Examiner:
Examiner Name	Examiner Name

Acknowledgments

I am very glad to thank to my project guide **Ms. Jyoti Vengurlekar** and our **H.O.D Dr.M.D.Patil** for their encouragement and tremendous guidance. Here it is very special thank feelings for my colleagues for their support and help. I have been fortunate to have received many useful suggestions from my colleagues which have greatly improved the clarity of my report. At the end special thanks to our **Principal Dr. Ramesh Vasappanavara**. I would like to appreciate suggestions and criticisms about the report from the readers.

Abstract

Indian Railways is the fourth largest railway network in the world. Although there is a tremendous growth in Indian Railways, this system is still plagued by a number of problems which require immediate attention. In this project we are considering the major problems that lead to accidents. Major problems include obstacles entry on to the track and cracks on the tracks. To overcome this have proposed a testing train which uses ultrasonic sensor with a range of 100cms and delay is 30 cm.Based on the distance between obstacle and the train, the train slows down. When the train is at a distance of 20cm we increase the delay in order to slow down the train and finally when it reaches to a distance of 15cm the train automatically stops.During summer and winter seasons the tracks may expand and contract due to which cracks may occur. The ultrasonic module setup is placed to testing train to detect cracks. Here we are using arduino microcontroller. After crack detection the testing train stops and the longitudinal and latitudinal positions are sent to firebase server via internet.

List of Figures

Contents

Chapter 1

NodeMcu

1.1 Hardware

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson, and spiffs.

Figure 1.1: NodeMcu

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266.The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core,[citation needed] widely used in IoT applications (see related projects). NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9.Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project,

then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

1.1.1 Pinout Diagram

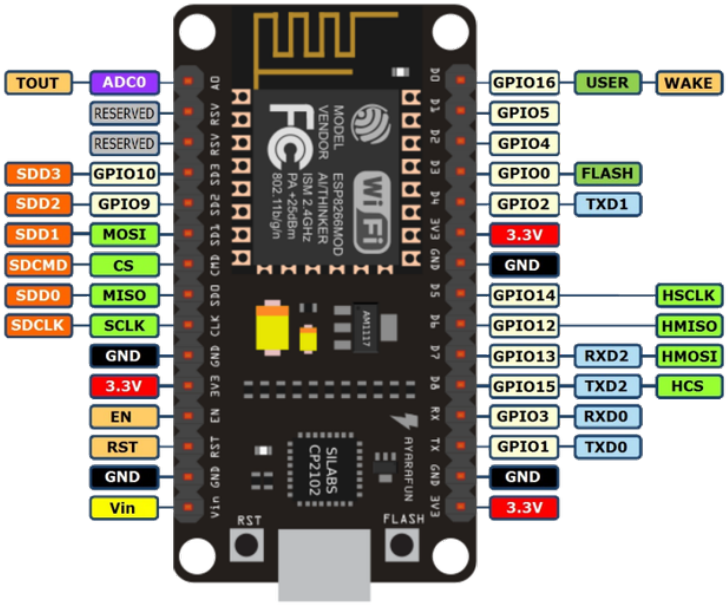

Figure 1.2: NodeMcu Pinout

Some of the features include:

i 10 digit Analog-Digital converter

ii On board Antenna and RF balun

iii Uses 802.11 b/g/n Wifi standards

iv Uses IPV4, HTTP, FTP, UDP TCP network protocols

v Operating voltages: 3-3.6V

vi Operating Frequency range: 2.4-2.6 GHz

vii On board power management modules,PLL, regulator, Power amplifier, Noise filters which makes it less external circuitry interface.

viii Configured in both Android iOS devices

Applications:

i Home Automation

ii Home Appliances i.e TV, Refrigerators, Light Bulbs , Fans etc. Control

iii Motor Speed control

iv Can be used in IP Cameras , Sensor Networks, Wearable electronics

v Security ID tags Baby Monitors

vi WiFi location aware devices and WiFi position systems etc

Specifications	ESP8266
MCU	Xtensa® Single-Core 32-bit L106
802.11 b/g/n Wi-Fi	Yes, HT20
Bluetooth	None
Typical Frequency	80 MHz
SRAM	160 kBytes
Flash	SPI Flash , up to 16 MBytes
GPIO	17
Hardware / Software PWM	None / 8 Channels
SPI / I2C / I2S / UART	2/1/2/2
ADC	10-bit
CAN	None
Ethernet MAC Interface	None
Touch Sensor	None
Temperature Sensor	None
Working Temperature	-40℃ ~ 125℃

Figure 1.3: NodeMcu Features

Chapter 2

Components

2.1 UltraSonic Sensor

Figure 2.1: Ultrasonic Sensor

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate

the distance between the sonar sensor and the object. Since it is known that sound travels through air at about 344 m/s (1129 ft/s), you can take the time for the sound wave to return and multiply it by 344 meters (or 1129 feet) to find the total round-trip distance of the sound wave. Round-trip means that the sound wave traveled 2 times the distance to the object before it was detected by the sensor; it includes the 'trip' from the sonar sensor to the object AND the 'trip' from the object to the Ultrasonic sensor (after the sound wave bounced off the object). To find the distance to the object, simply divide the round-trip distance in half.

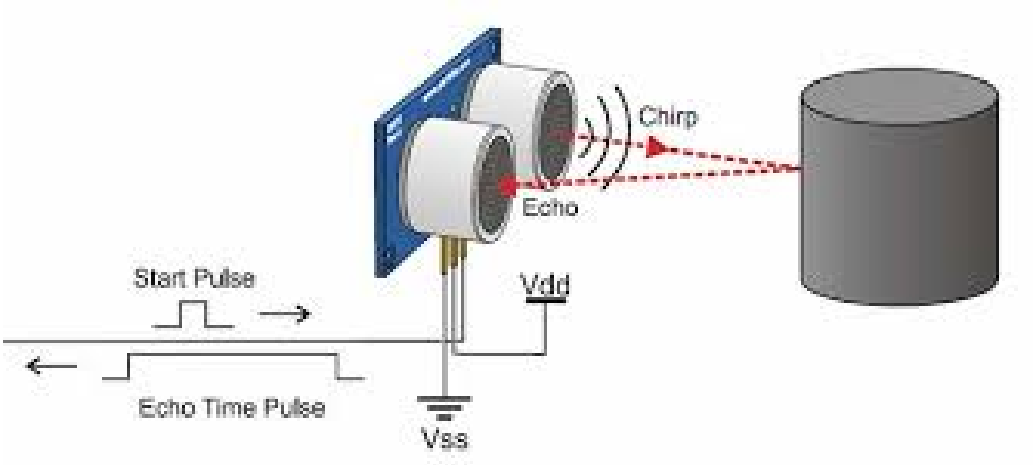

Figure 2.2: Working of Ultrasonic Sensor

The accuracy of Ultrasonic sensor can be affected by the temperature and humidity of the air it is being used in. However, for these tutorials and almost any project you will be using these sensors in, this change in accuracy will be negligible.

It is important to understand that some objects might not be detected by ultrasonic sensors. This is because some objects are shaped or positioned in such a way that the sound wave bounces off the object, but are deflected away from the Ultrasonic sensor. It is also possible for the object to be too small to reflect enough of the sound wave back to the sensor to be detected. Other objects can absorb the sound wave all together (cloth, carpeting, etc), which means that there is no way for the sensor to detect them accurately. These are important factors to consider when designing and programming a robot using an ultrasonic sensor.

2.2 Motor Driver L293D

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 7 and 10 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled.

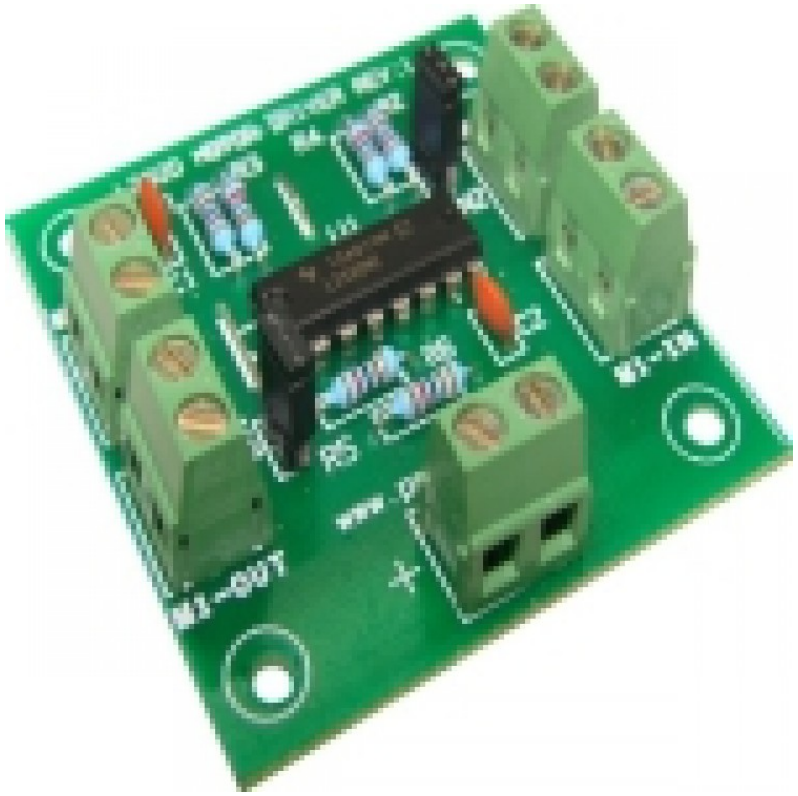

Figure 2.3: Motor Driver

As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

2.3 DC Motor

Electrical motors are everywhere around us. Almost all the electro-mechanical movements we see around us are caused either by a AC or a DC motor. Here we will be exploring DC motors. This is a device that converts DC electrical energy to a mechanical energy.

2.3.1 Principle

This DC or direct current motor works on the principal, when a current carrying conductor is placed in a magnetic field, it experiences a torque and has a tendency to move. This is known as motoring action. If the direction of current in the wire is reversed, the direction of rotation also reverses. When magnetic field and electric field interact they produce a mechanical force, and based on that the working principle of DC motor is established. Structurally and construction wise a direct current motor is exactly similar to a DC generator, but electrically it is just the opposite. Here we unlike a generator we supply electrical energy to the input port and derive mechanical energy from the output port. We can represent it by the block diagram shown below.

Figure 2.4: Ultrasonic Sensor

2.4 Google Firebase

Figure 2.5: DC Motor

Store and sync data with our NoSQL cloud database. Data is synced across all clients in realtime, and remains available when your app goes offline.

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

2.4.1 How does it work?

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

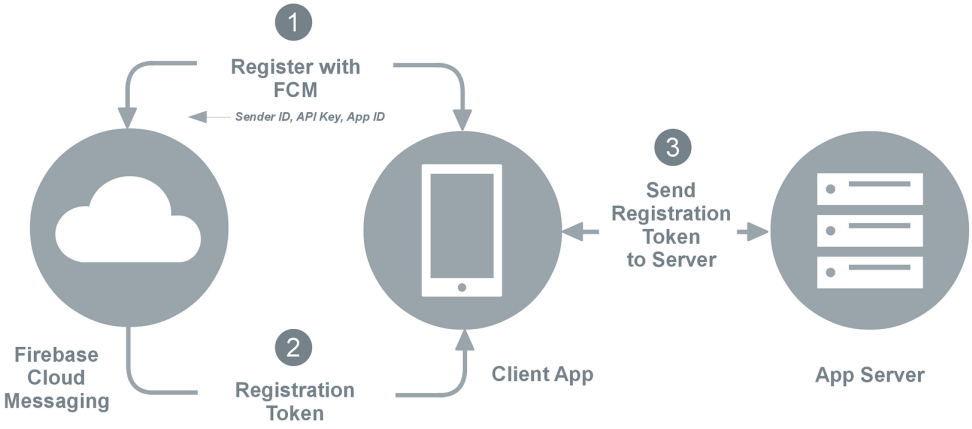

Figure 2.6: Firebase connections

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great realtime experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

2.5 Google Maps API

The Google Maps Geolocation API returns a location and accuracy radius based on information about cell towers and WiFi nodes that the mobile client can detect. This document describes the protocol used to send this data to the server and to return a response to the client.

Communication is done over HTTPS using POST. Both request and response are formatted as JSON, and the content type of both is application/json.

Before you start developing with the Geolocation API, review the authentication requirements (you need an API key) and the API usage limits.

Geolocation requests are sent using POST to the following URL:

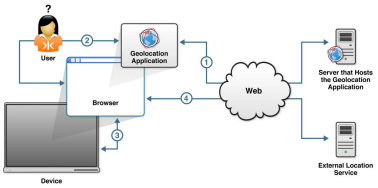

Figure 2.7: Circuit diagram of Kit

https://www.googleapis.com/geolocation/v1/geolocate?key=YourApiKey You must specify a key in your request, included as the value of a key parameter. A key is your application's API key. This key identifies your application for purposes of quota management.

2.6 Circuit diagram

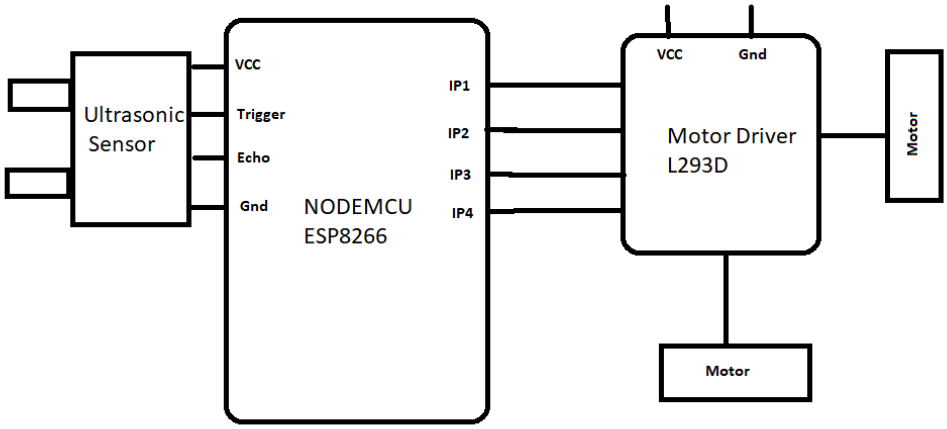

Figure 2.8: Block Diagram

1 The Vcc required for ultraspnic is connected to the VU of Node Mcu for 5V supply.

2 Trig pin of sensor is connected to the digital output pin D5 of Node Mcu [3] Echo pin is connected to the Node Mcu pin D6.

4 Pin G of Node Mcu is used as a common ground for both ultrasonic and motor driver.

5 Motor driver is biased using an external 5V battery supply.

6 The four input pins IN1,IN2,IN3 and IN4 are connected to the digital pins D0,D1,D2 and D3 respectively.

7 The four output pins of motor driver will be used for driving two motors where each motor will get output connections from motor driver.

2.7 List of components and applications

It consists of various components named below:

- Nodemcu esp8266
- Motor Driver L293D
- Ultrasonic Sensor
- Google Firebase
- Google Maps API
- Jumpers
- DC Motors

Chapter 3

Geolocating with nodemcu

3.1 Introduction

The ESP8266 is capable of actively scanning for nearby Wi-Fi access points operating in the 2.4GHz band. The scanning only takes a couple of seconds and the ESP8266 may then be put in a power-saving mode, thus saving power in battery-powered systems. This is achieved by issuing a single AT command that lists all APs available in the area, along with their corresponding SSID, RSSI and MAC address:

1 . Firstly, make sure that ESP8266 is powered up and functional by issuing command: AT The response must be: OK

2 . Then, the host controller can obtain a list of all available Wi-Fi APs by issuing command: AT+CWLAP In response to this command, the ESP8266 will return a list of APs available, and their corresponding SSID (the second parameter), RSSI value (the third parameter) and MAC address (the fourth parameter), which are important dening parameters that also are required for the geolocation.

An example of the output data is: +CWLAP:(2,"HotelFlower",-76,"c8:3a:35:b3:16:48",11,0,0) +CWLAP:(3,"IoTBits",-93,"08:bd:43:66:6a:86",6,-27,0)OK

3.2 Obtaining Geolocation

To prepare the ESP8266 for establishing a connection with the online geolocation API or service, the model must be connected to a router with internet access. This can be done by setting the ESP8266 to station mode (so that it can be connected to a router).

3.3 Establishing a secure connection

1 . Set ESP8266 to station mode: AT+CWMODE=1 An access point can now be connected to ESP8266 by using: AT+CWJAP="SSID","password" If the connection is successful and the AP assigns the ESP8266 with a valid local IP address, the command will return: WIFI CONNECTED WIFI GOT IP The ESP8266 can now be made to establish connection with a geolocation API or service over a secure connection (https). 2. Initialize the SSL buffer as follows: AT+CIPSSLSIZE=2048

2 . Connect securely to Google API by issuing command: AT+CIPSTART="SSL","www.googlea

3 If everything works as expected, this command will return: CONNECT OK Otherwise, the ESP8266 will respond with ERROR.

3.4 Obtaining a location x by using Google API

The Google geolocation API requires the user to send in dening parameters via a POST request, formatted as a JSON string, to obtain a location x. The URL must contain the user API key for quota management purposes by Google.

3.5 Working

i When our wagon is moving, ultrasonic sensor keeps on continously sensing the distance between wagon and the track.

ii As soon as a crack appears on the track,ultrsonic sensor detects it by comparing its distance with the reference distance between wagon and track.

iii .Once the crack is detected ultrasonic sensor gives a low input signal to ESP8266 micro controller

iv Now our micro controller ESP8266 switchs the motor driving input to low, thereby stoping the wagon from moving any further.

v .Inorder to get the location of crack and where the wagon is ,we make use of ESP8266's on-chip wi-fi module.

vi .We setup a network on the wagon using a dongle which is connected to the ESP8266 wifi. By using this we continously track the wagon.

3.6 Power Consumption

A Brief Note on Power Consumption The approximate current consumption in some low-power modes is as follows: Modem sleep mode: 15 mA Light sleep mode: 3 mA Deep sleep mode: 20 uA (please account for ash standby mode current) Assuming that the sleep mode is set to Modem-sleep (i.e. AT+SLEEP=2) and the DTIM interval is 100ms, when connected to an AP, the power consumption trends will be similar to what is outlined below:

For applications that stay connected to APs: Assuming that the ESP8266 is connected to an AP with light sleep between beacons activated, and DTIM beacons every 100 ms, the average current consumption when maintaining connection with an AP is less than 20 mA. For applications that simply scan for APs: The typical scan time for active SSID scanning can range from 2 sec to 2.5 sec.

Chapter 4

Result

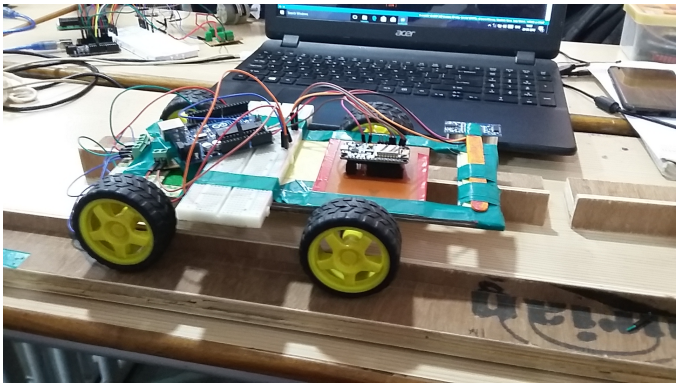

Figure 4.1: Working model

Chapter 5

Conclusion

It can thus be concluded that the ESP8266 may be used for geolocation in multiple ways: by obtaining AP properties and then using Google geolocation API to locate a user-device, or by locating the end-users geolocation directly, when the ESP8266 has access to the internet. Availability of the cellular network parameters is not a necessity for implementing geolocation with ESP8266. However, the accuracy of results may vary depending on how many access points are available and connected to the internet. The power consumption of the ESP8266 module, when used as a geolocating device, may be kept extremely low with Deep-sleep mode enabled between scans.

Bibliography

[1] Google images (http://www.google.com/images)

[2] Google Maps Api form www.googlemapsapi.com

[3] Technical terms from www.learnabout-electronic.org And from Wikipedia

[4] "Book Name ",Nodemcu dev kit using Arduino IDE: Get started with ESP8266 Kindle Edition by Maagesh Jayakumar