EC Lab Project 2023-24 II

Cycle Track Sentinel

Section: D Table: 2

Group Members:

1. Chirumamilla Satya Keerthana (210290)
2. Shobhit Sharma (210992)
3. Sumay Avi (211071)

Aim

The objective of this project is to develop an integrated system that can display the location of a cycle, as well as its running characteristics such as speed, heading and altitude above sea level on a mobile phone. Additionally, it features a buzzer, to help a person exactly locate their cycle among many others at a certain place.

This system is easy to operate. Ultimately, the project aims to both prevent robbery and display essential metrics of any cycle to all people present in our campus.

Components

- Node MCU (Microcontroller)
- Ublox Neo-6M (GPS Module)
- Buzzer
- BC 547 (NPN BJT)
- 5V Battery
- USB 2.0 Cable Type A/B
- Breadboard
- Resistors
- LED
- Jumper wires

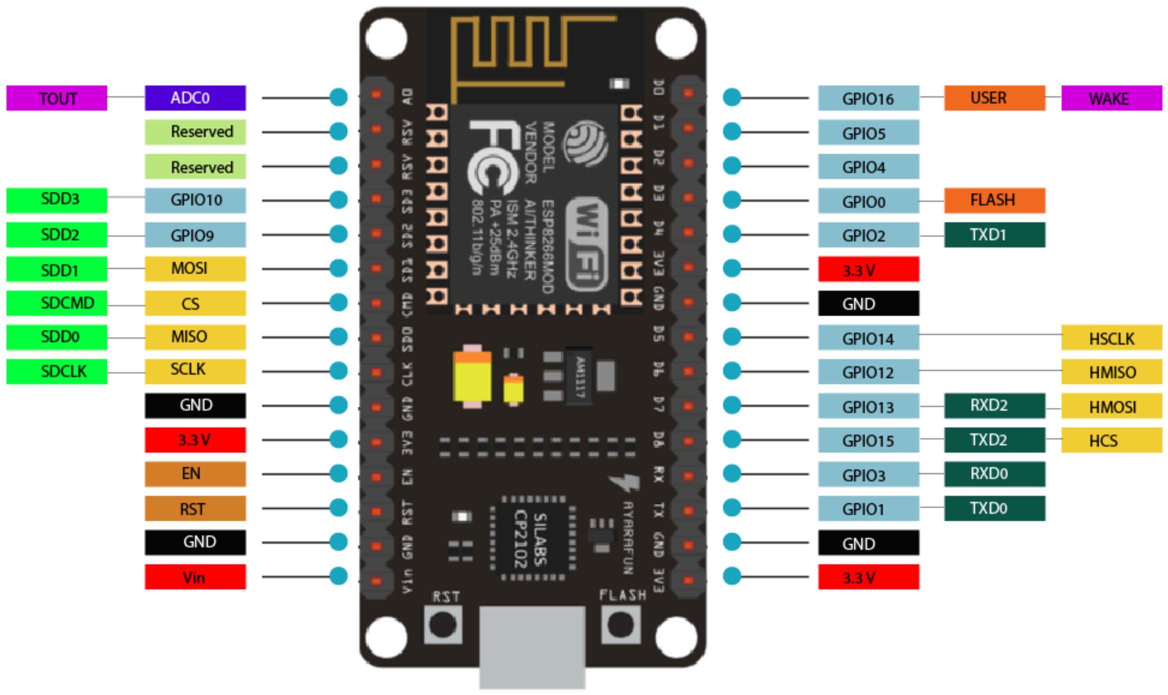
Software

Arduino Cloud

Description of Each Component

1. **Node MCU:** Node MCU is an Arduino compatible ESP8266 based Microcontroller. It consists of an in-built wifi module hence it has been used as a substitute for the arduino board. It's specifications are:

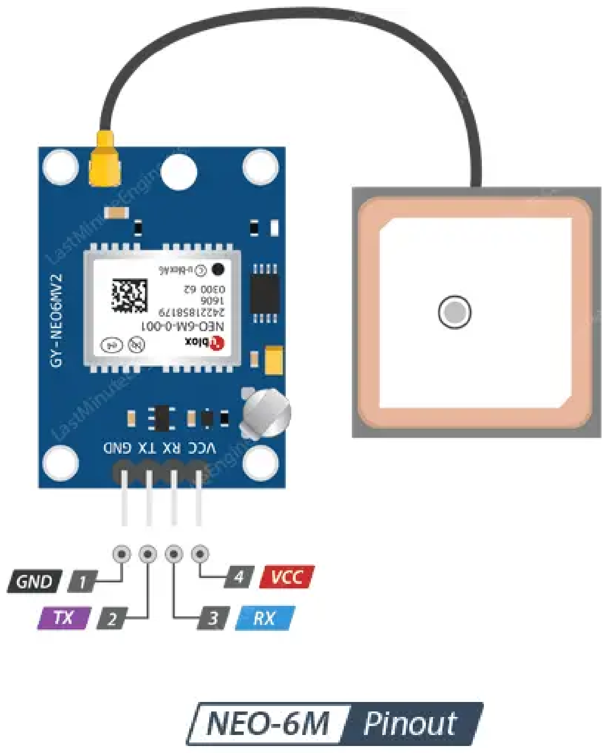
    a. Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
    b. Operating Voltage: 3.3V
    c. Input Voltage: 7-12V
    d. Digital I/O Pins (DIO): 16
    e. Analog Input Pins (ADC): 1
    f. UARTs: 1
    g. SPIs: 1
    h. I2Cs: 1



NodeMCU V3 Pinout

2. **Ublox Neo-6M**: This GPS module connects to satellites in real time to retrieve its location. It also provides information on the current date and time.
   Its specifications are:

   a. Power Supply: 3V/5V
   b. Model: GPS-NEO-6M-001
   c. Antenna: Ceramic antenna
   d. Battery: Rechargeable battery back-up
   e. Signal light: LED light
   f. Antenna size: 25*25mm
   g. Model size: 25.5mm*31.5mm
   h. Mounting Hole: 2mm
   i. The default baud rate: 38400
   j. The default output: Compatible with NMEA0183 protocol



Implementation of the circuit

A. Connections:
   1. Node MCU:
      NodeMCU1 GND to Bus GND

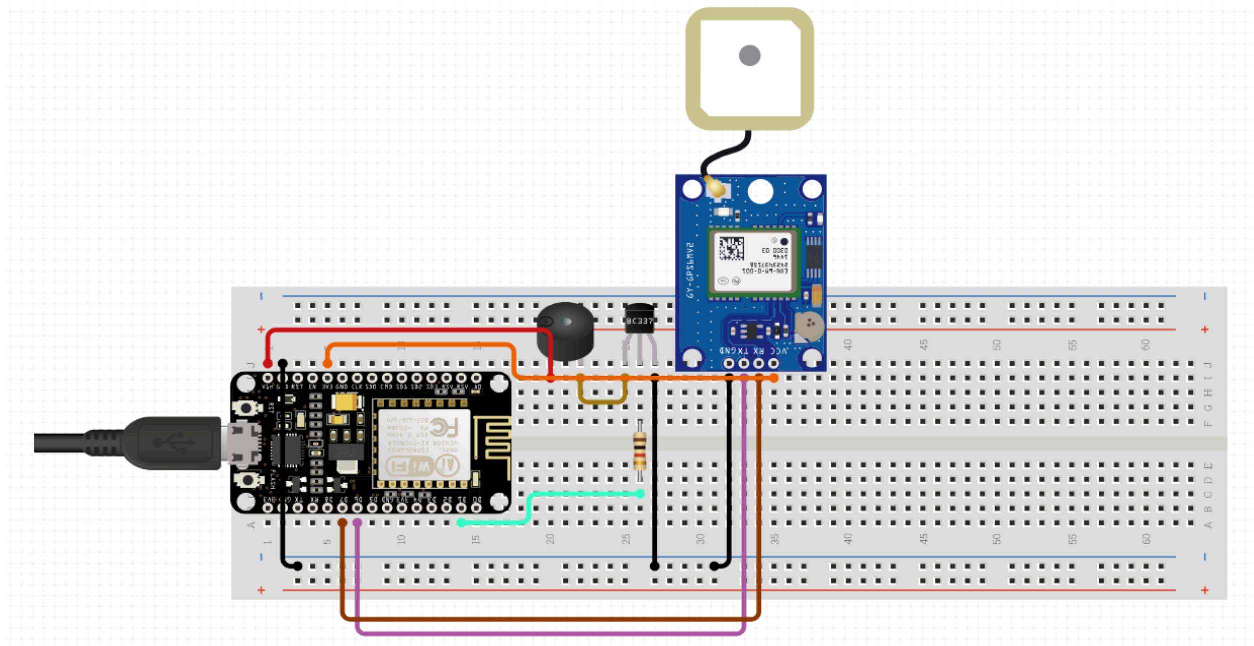
   2. Buzzer:
      Buzzer neg to BC 547 Collector
      Buzzer pos to NodeMCU1 Vin

   3. BC 547 (NPN BJT)
      BC 547 E to Bus GND

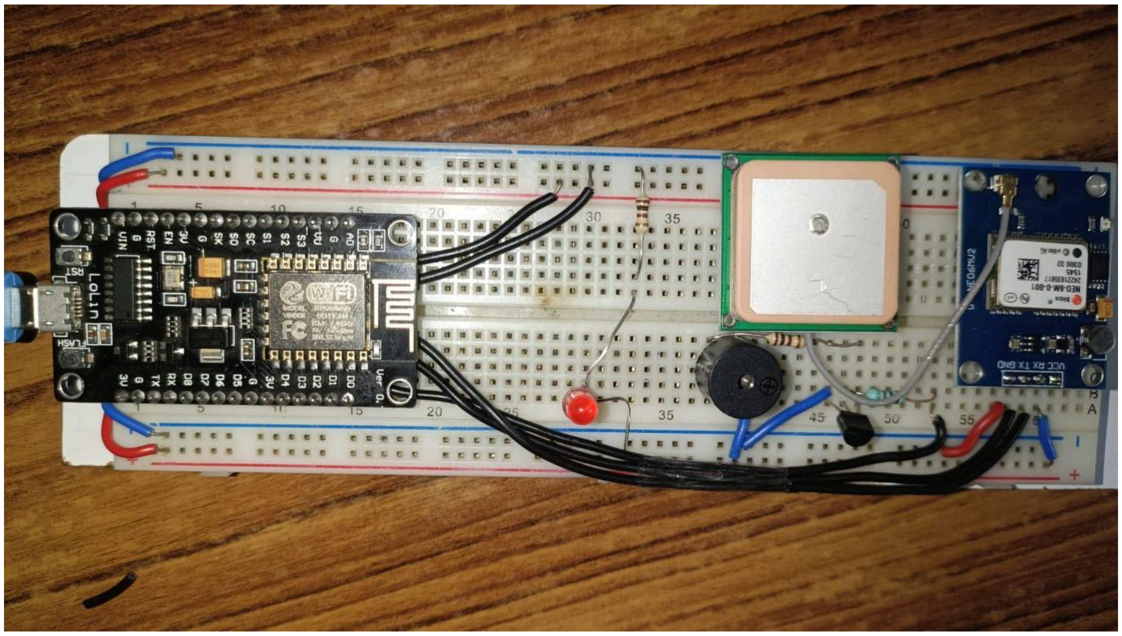
   4. Res1K Ohm
      One end of Resistor to BC 547 Base
      Other end of Resistor to NodeMCU1 D1

   5. GPSNeo6M (GPS Module)
      GPSNeo6M GND to Bus GND
      GPSNeo6M TX to NodeMCU1 D6
      GPSNeo6M RX to NodeMCU1 D7
      GPSNeo6M VCC to NodeMCU1 3V3

Circuit Diagram



Circuit Implementation



Programming the Setup

We have used the Arduino Cloud editor to write and upload the code to the Node MCU device. This is the code used for implementing our project consisting of Node MCU, Neo-6M Module:

```
#include "thingProperties.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
/*
   This sample code demonstrates the normal use of a TinyGPS++ (TinyGPSPlus) object.
   It requires the use of SoftwareSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 4(rx) and 3(tx).
*/
static const int RXPin = D4, TXPin = D3;
// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
```

```
void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(57600);

  ss.begin(9600);
  delay(1500);

  pinMode(D0,OUTPUT);
  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  // Your code here
  if (ss.available() > 0)
  {
    gps.encode(ss.read()); //read gps data
    {
      speed=gps.speed.kmph();
      altitude=gps.altitude.meters();
      if (gps.location.isValid()) //check whether gps location is valid
      locloc=Location(gps.location.lat(), gps.location.lng());
    }
  }

}


/*
  Since Led is READ_WRITE variable, onLedChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onLedChange()  {
  // Add your code here to act upon Led change
  if(led==1)
  {
    Serial.println("Buzzing");
    digitalWrite(D0,HIGH);
  }
  else
  {
    Serial.println("Not Buzzing");
    digitalWrite(D0,LOW);
  }
}
```

Brief Explanation of the code

1. **GPS Module Setup**: We have utilised the TinyGPS++ library to interface with a GPS module connected through digital pins D4 (RX) and D3 (TX) using the SoftwareSerial library.
2. **Serial Communication**: A serial communication at 57600 baud rate was established for debugging purposes, and a SoftwareSerial connection at 9600 baud was initiated with the GPS module to read data.
3. **Arduino IoT Cloud Configuration**: We have implemented a connection to the Arduino IoT Cloud using predefined properties in the `thingProperties.h` file. The debug message level was set to 2 to enable detailed monitoring of the connection status.
4. **Data Processing**: The code continuously checks for available data from the GPS module in the main loop. Upon receiving data, it decodes this information to extract speed (km/h), altitude (metres), and geographic location (latitude and longitude).
5. **Variable Updating**: We have updated the Arduino IoT Cloud variables `speed`, `altitude`, and `locloc` (a `Location` object) with the respective values obtained from the GPS data, provided the location data is valid.
6. **IoT Cloud-Controlled Actuation**: The `onLedChange()` function was implemented, which gets triggered by changes in the `led` variable from the IoT Cloud. This function controls the state of a connected LED or buzzer (on pin D0), indicating the device's status ("Buzzing" or "Not Buzzing") based on the value of `led`.
7. **Hardware Interface**: We have defined the pin D0 as an output to control the connected LED or buzzer, demonstrating an integration between cloud-based commands and physical actuation.
8. **Debugging and Monitoring**: System debug information is printed to the serial monitor, providing insights into the network and IoT Cloud connection states, aiding in troubleshooting and ensuring successful operation.

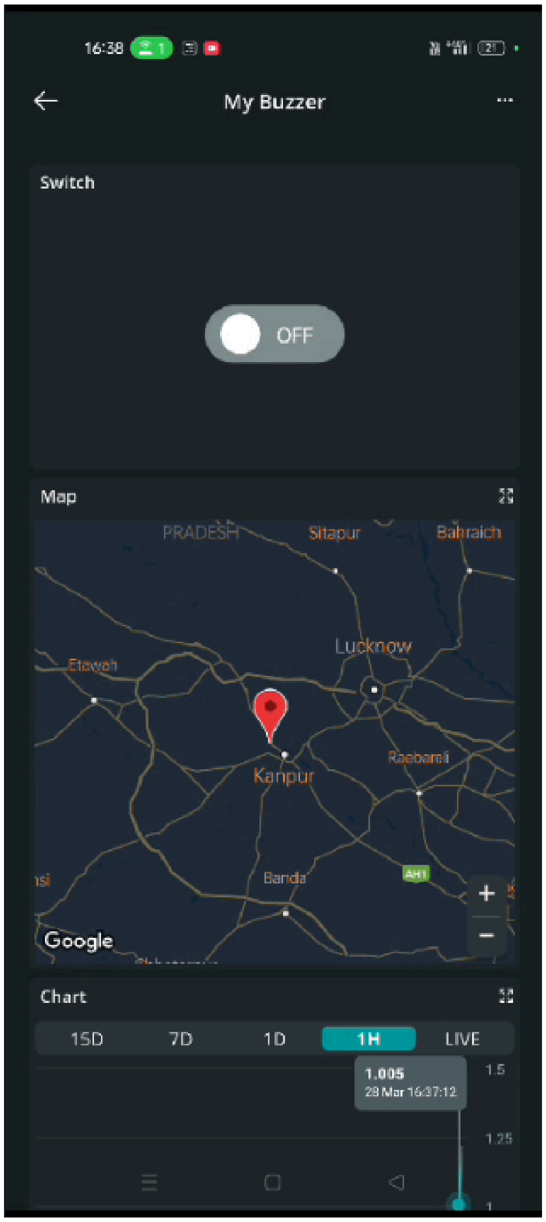
Video demonstrating the project

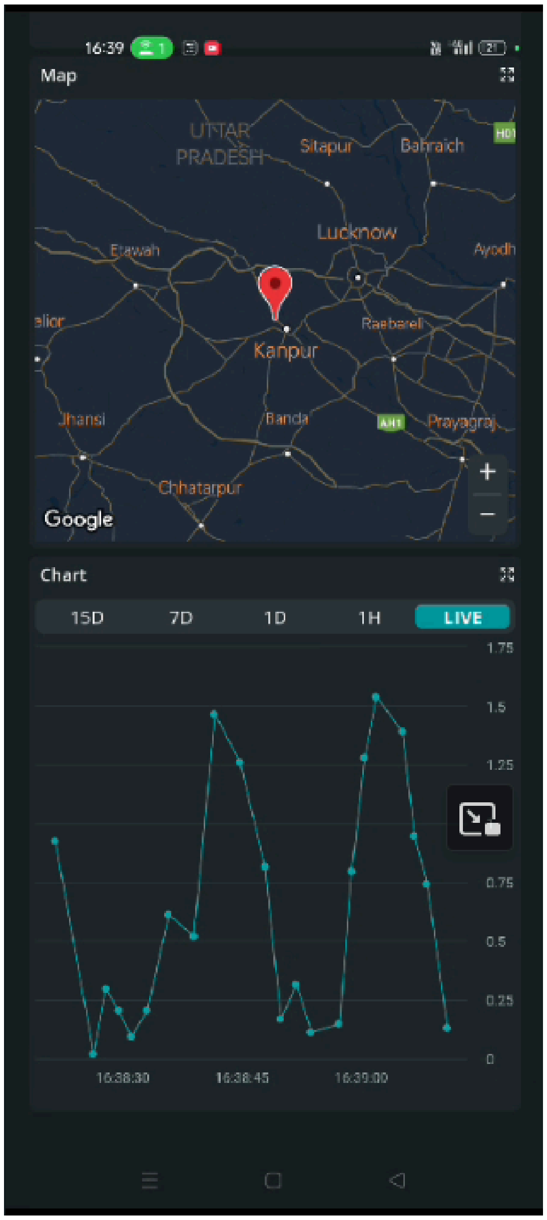
https://drive.google.com/file/d/1bdOXct5xvy45G8aUCodbGWfox1QnRILw/view

Displaying the contents on our interface

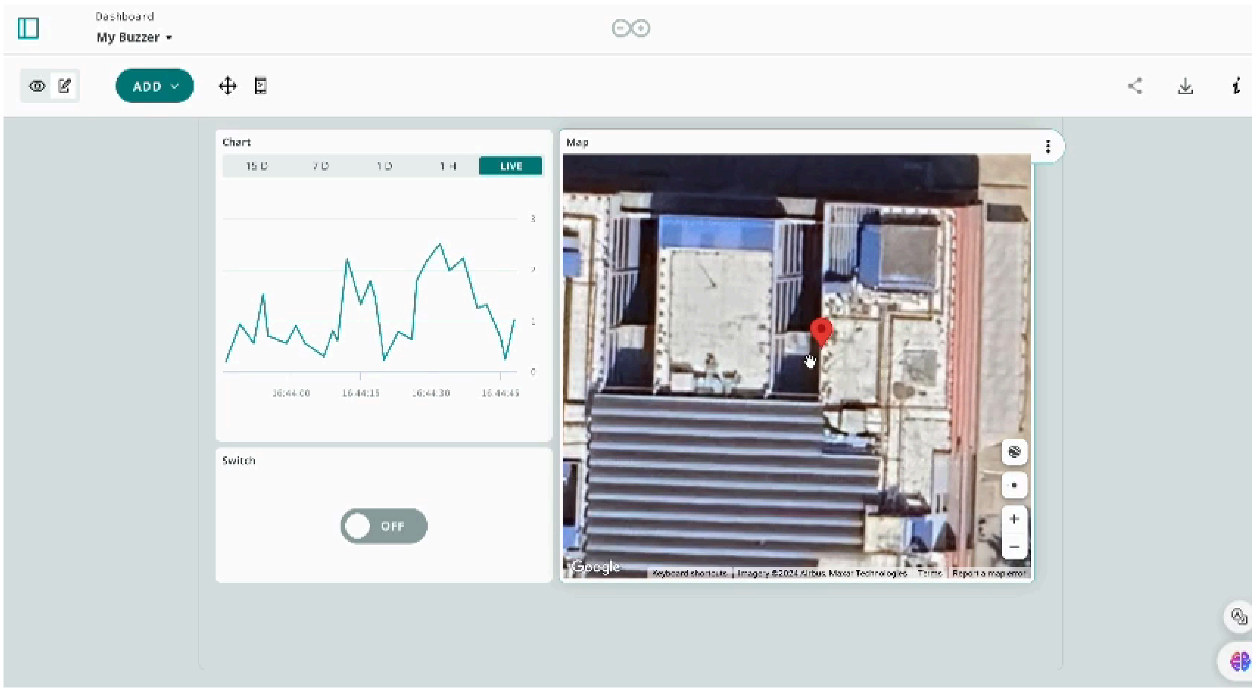
We have utilised the Arduino dashboard feature to display the contents on the mobile/laptop

A. Mobile View





B. Laptop View



Power Calculation

$V_{\text{across chip}}$ = 4.5 V

$I_{\text{through chip}}$ = 48 mA

$P_{\text{consumed by circuit}} = V_{\text{across chip}} * I_{\text{through chip}}$ = 216 mWatts

If powered by a 5V 1800mAh battery, the setup would run for approximately 8.3hrs
To make the setup be used without recharging time and again, other sources of energy such as Dynamo or Solar panel maybe be incorporated into the circuit (Future Work)

Conclusion

This project successfully demonstrates the integration of a GPS module with the Arduino platform, enabling real-time tracking of geographic location, speed, and altitude. By connecting the device to the Arduino IoT Cloud, we have allowed for remote monitoring of these parameters and the control of connected actuators based on cloud commands, illustrating the potential of IoT applications in real-world scenarios.

Additional Questions

Q. What problem are you trying to solve, and why is it important/interesting?

A. Our group project aims to solve multiple problems plaguing the residents of any college campus, such as IITK. One of the biggest problems faced here is cycle theft. With this GPS module, we can locate our cycle on Google Maps just by a simple click of a button on our phone.

   Another issue faced is during any special events or gatherings, it becomes quite difficult to find our cycle among scores of others. The above location can take us only to a broad area where our cycle is located. To exactly narrow down to our cycle, the buzzer module will emit noises when we wish for it to do so. Then we can easily pinpoint our cycle. Health enthusiasts would also benefit from this, as it would display the speed, distance travelled, altitude above sea level (helpful in hilly regions) on our phone.

   Our project was interesting in the sense that with the help of a few simple electronic components, we are solving a huge practical problem faced by everyone who owns a cycle at any college campus.

Q. What are the existing solutions? Describe a few of them and list any shortcoming in them. Is your solution approach unique in some way?

A. The existing solutions to find a lost/stolen cycle are almost impractical, such as manually hunting for it at each and every nook and corner of the campus, or reporting the matter to the CIS guards. Another solution is to buy an expensive tracker from the market and attach it to the cycle. This option becomes infeasible for some people, especially students. Our solution provides accurate information on the location of the cycle, helping to retrieve it in case of theft. Additionally, the components we have used are relatively inexpensive, bringing the overall cost much lower than any market based solution. Hence this module can be used by the entire campus community, even students.

Q. What resources do you require to complete the project? Give a breakdown of the tasks that you need to accomplish week by week to complete the project.

A. The complete set of components and tools used in this project have been listed above

   Timeline followed:
   Week 1: Getting familiar with all the components, such as the Node MCU and Neo-6M module
   Week 2: Assembling the components and making the entire circuit.
   Week 3: Writing the code in the Arduino Cloud and integrating it with the hardware.

Acknowledgements

On behalf of the entire team, we would like to thank Prof. Mazhari for providing us this wonderful opportunity to create and implement a project of our choice. This project gave us insights into both microcontrollers and circuitry. It even exposed us to challenges that arise during a practical implementation of a module, both in terms of software and hardware.

I would also like to thank the lab staff and TA’s of the EE 381 Lab for their constant support and assistance they provided whenever necessary.