Project Report on ”Arduino based GPS Speedometer”

Keerthana Patlolla, Rohit Balekundri, Soundarya Madhuri Royyuru
Team 7, GMU ECE508 IoT, Spring 2023
G 01397611,G 01395841,G 01374706

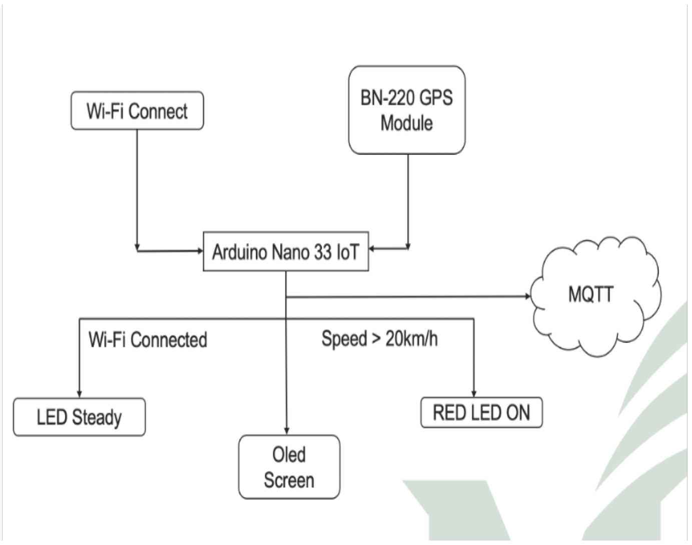
I. Abstract

The design and creation of a GPS speedometer system employing an Arduino Nano 33 IoT, a GPS module, an OLED display, and the MQTT protocol are proposed in this project. Using GPS technology, the system seeks to deliver precise speed and position information. Due of its low power requirements, small size, and wireless capabilities, the Arduino Nano 33 IoT was selected. Accurate position information is provided by the GPS module, which is in charge of receiving signals from GPS satellites. The Arduino Nano 33 IoT receives this data and analyses it, calculating the device’s speed in the process. Real-time speed and position data are shown on the OLED display, which also offers the user a simple and easy-to-read user interface and to enable remote monitoring and tracking by sending the speed and position data to a cloud or distant server. The system’s usefulness is improved by this feature, which also makes it easier to analyze data for use in a variety of applications, including fleet management, tracking, and navigation.Overall, this project offers a novel way to track position and speed data using GPS and the MQTT protocol, creating a complete system for numerous applications. Because of its small size, low power need, and wireless capabilities, the system is an efficient and practical choice for asset tracking, vehicle tracking, and navigation.In this project, the GPS-based speedometer is the major focus because it calculates longitude and latitude and is therefore more accurate and useful than a traditional speedometer.

II. Literature Review

Using cloud-based infrastructure, real-time GPS data gathering for mobility tracking was described by Du et al. The method described in this work uses a smartphone app and cloud-based infrastructure to gather and analyse GPS data. The technology has potential uses in traffic management, emergency response, and urban planning and may be used to monitor mobility trends in real-time.Alotaibi et al.’s ”GPS-based vehicle tracking system” The design and construction of a GPS-based car tracking system utilizing Arduino and GSM technologies are covered in this paper. A vehicle’s position, speed, and direction may be tracked by the system, and it can transfer this information to a distant server through the GSM network. Ghosh and Dutta’s ”Design and development of an Arduino-based weather monitoring and prediction system” This essay offers a system that gathers meteorological information including temperature, humidity, and pressure using an Arduino board and a variety of sensors. In order to transfer this data to a distant server for additional analysis and prediction, the system can also be linked to the internet. Bhandari et al.’s ”GPS-based vehicle tracking and monitoring system using Arduino” The development of a GPS-based car tracking and monitoring system utilizing Arduino and GSM technologies is covered in this paper. A vehicle’s position, speed, and direction may be tracked by the system, and it can transfer this information to a distant server through the GSM network.

III. Introduction

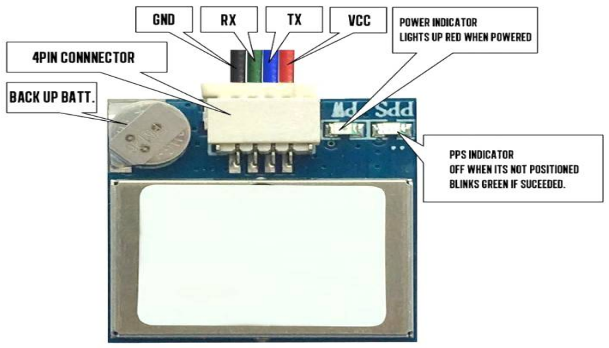


With the enormous parallel computational power, it is now feasible to link GPS technology with the internet due to the growing popularity of the IoT, enabling real-time tracking and monitoring of speed and position data. A robust microcontroller board created exclusively for Internet of Things applications is the Arduino Nano 33 IoT. The advantages of the Arduino platform are combined with the simplicity of Wi-Fi and Bluetooth connection that is already built in. This makes it the perfect option for creating GPS speedometers that can relay data to a distant server or the cloud for more research and oversight.The use of GPS technology for different purposes, including vehicle tracking, weather monitoring, and mobility monitoring, has attracted increasing interest in recent years. These investigations have demonstrated the accuracy and dependability of GPS technology as a means of gathering position and speed information. It is feasible to create cutting-edge systems that can assist to enhance traffic management, emergency response, and urban planning by merging GPS technology with the IoT. With the help of the Arduino Nano 33 IoT, GPS module, OLED display, and MQTT, we intend to construct and create a GPS speedometer for this project. To build a dependable system, we'll combine the advantages of the Arduino platform with GPS technology an accurate and dependable speedometer that can transmit data to the internet or an external server for monitoring and analysis in real time. The speed and position information will be shown on an OLED display, and the data will be sent to a cloud or distant server via the MQTT protocol. By creating a system of this kind, we intend to make a positive impact on the expanding fields of IoT as well as GPS technology.

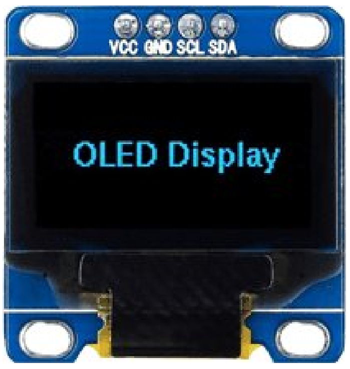
IV. Hardware Components

1) Arduino Nano 33 IoT: The SAMD21 Cortex-M0+ CPU serves as the foundation for this small and adaptable board. It is a great option for Internet of Things applications because it includes WiFi and Bluetooth connection built in. The board has a variety of digital and analog pins for connecting to sensors and other peripherals, as well as a micro-USB connection for programming and power.

2) GPS Unit:A GPS (Global Positioning System) module is a piece of equipment that can pick up satellite signals and calculate exact position and time information. The BN-220 GPS module, which is small and low-power and supports both GPS and GLONASS satellite systems, is the GPS module utilized in this project. It delivers precise and trustworthy location information, which is crucial for applications for speedometers. Also its interoperability with numerous satellite systems, high sensitivity, and quick TTFF all contribute to its dependability, while its user-friendly configuration and low power consumption give convenience and versatility. Receiving Format :GPS,GLONASS,Galileo,BeiDou,QZSS and SBAS Frequency : GPS L1,GLONASS L1,BeiDou B1,SBAS L1,Galileo E1 Channels : 72 Data Protocol : NMEA-0183 or UBX, Default NMEA-0183 Single GNSS : 1Hz-18Hz Concurrent GNSS : 1Hz-10Hz


GND
RX
TX
VCC
POWER INDICATOR LIGHTS UP RED WHEN POWERED
4PIN CONNNECTOR
BACK UP BATT.
PPS INDICATOR OFF WHEN ITS NOT POSITIONED BLINKS GREEN IF SUCEEDED.

3) OLED Display: An OLED display, also known as an organic light-emitting diode, is a form of technology for display that use organic compounds to generate illumination whenever a current of electricity is conducted through them. The great contrary, minimal energy consumption, and broad viewing angles of OLED screens are well recognized. The SSD1306, a monochrome 128x64 pixel OLED display, is the one utilized in this project. Because it can show both text and visuals, it is appropriate for showing the position and speed data. Driver IC: SH1106 Input Voltage: 3.3V-5V DC Resolution: 128x64 Interface: I2C Current consumption: 8 mA Pixel color: Blue Viewing angle: ¿160 degree **Pin Description** :
VCC: Input power supply 3.3-5V DC
GND: Ground reference pin
SCL: Clock pin of the I2C interface
SDA: Serial Data pin of the I2C interface

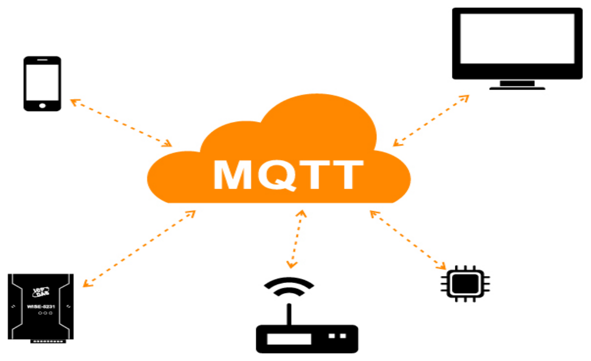


V. Software Used

1) Arduino IDE: The software tool used to program Arduino microcontroller in the project is called the

Arduino IDE (Integrated Development Environment). Open-source hardware and software make up the foundation of the electronics platform known as Arduino. An uploader, a compiler, and a code editor are all included with the Arduino IDE.Through a blend of common C++ programming tools and unique libraries created to connect with particular sensors or hardware components, the Arduino IDE supports a variety of sensors and libraries.It is simple to utilize these devices in the projects having to be familiar with all the low-level specifics since these libraries frequently include particular classes or methods that offer a high-level interface for interacting with the sensor or hardware component.

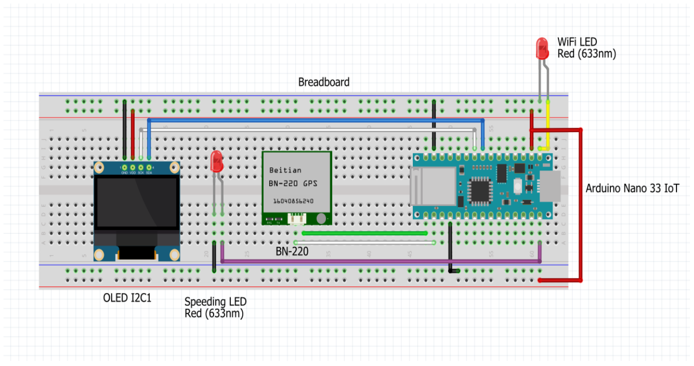
2) MQTT: The lightweight communications protocol MQTT (Message Queuing Telemetry Transport) is used in Internet of Things (IoT) applications. It supports both request/response and publish/subscribe communication patterns and is built for low-bandwidth, high-latency networks. The position and other data are sent over MQTT in this project speed up the transmission of data to a distant server for further processing or viewing.



We are using MQTT over other cloud networks because :
:
1. MQTT is lightweight and efficient, making it appropriate for resource-constrained devices and networks. The protocol reduces bandwidth utilization and power consumption, allowing for efficient data transport even in low-power or constrained network conditions.
2. MQTT uses a publish-subscribe messaging structure, which enables for flexible and scalable device communication. This paradigm allows devices and applications to be decoupled, allowing for real-time, asynchronous communication. Publishers and subscribers can communicate without the requirement for a direct connection, which improves scalability and flexibility.
3. Bandwidth optimization is possible because to MQTT's publish-subscribe paradigm and topic-based filtering. Messages are only sent to subscribers who have signed up for relevant topics, decreasing network traffic and conserving bandwidth.
4. MQTT provides various levels of QoS to ensure message delivery dependability. Publishers can select the appropriate QoS level (e.g., at most once, at least once, or exactly once) based on the desired balance of message delivery assurance and network overhead. This adaptability provides consistent communication in a variety of situations.
5. MQTT supports asynchronous communication, allowing devices to publish or subscribe to topics at any moment without establishing a persistent connection. When devices are not actively engaged in communication, their asynchronous nature allows them to enter low-power modes, conserving energy and prolonging battery life for battery-powered devices.
6. MQTT brokers can allow offline messaging, which means that messages can be saved and given to subscribers when they reconnect. This feature ensures that messages are not lost as a result of temporary network disconnections, making MQTT suited for scenarios with intermittent network availability.
7. MQTT is designed to handle large-scale deployments and can accommodate a high number of linked devices. Its lightweight design, fast message routing, and customizable subject structure allow for easy scalability and flexibility to varied IoT use cases. MQTT brokers can be dispersed to accommodate rising workloads.
8. MQTT can run over a variety of transport protocols, including TCP/IP, WebSocket, and others. Because of its adaptability, MQTT can work across several network types and platforms, making it compatible with a wide range of devices, operating systems, and programming languages.

VI. Working

Circuit Connection:



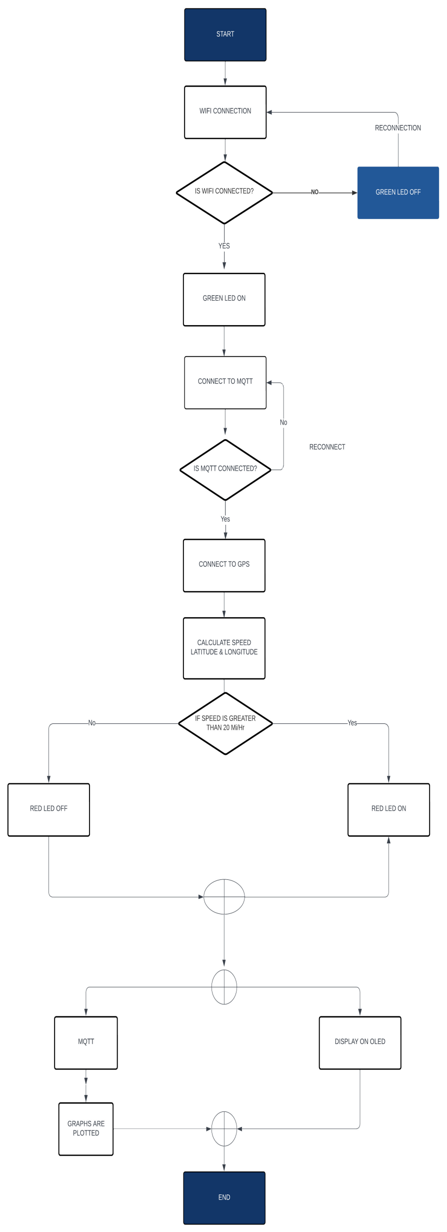
OLED Display to Arduino Nano IoT 33:
The VCC (Power) pin of the OLED display is connected to the 3.3V pin of the Arduino Nano IoT. The GND (Ground) pin of the OLED display is connected to the GND pin of the Arduino Nano IoT 33. The SDA (Serial Data) pin of the OLED display is connected to the SDA pin (usually pin A4) of the Arduino Nano IoT. SCL (Serial Clock) pin of the OLED display is connected to SCL pin (usually pin A5) of the Arduino Nano IoT 33.

GPS (BN-220) to Arduino Nano IoT 33: The GPS module's VCC (Power) pin is connected to the Arduino Nano IoT's 3.3V pin. 33. The GPS module's GND (Ground) pin is connected to the Arduino Nano IoT's GND pin 33. The GPS module's

TXD (Transmit Data) pin is connected to the Arduino Nano IoT's RX (Receive) pin (usually pin 0). The GPS module's RXD (Receive Data) pin is connected to the Arduino Nano IoT's TX (Transmit) pin (usually pin 1).

It's critical to remember that the RX and TX pins on the Arduino Nano IoT 33 are linked in opposite directions to the TX and RX pins on the GPS module. This is because the communication is serial.

Working:



1. Following the setup, the Arduino checks to see if the wifi is connected or not, according to the algorithm shown in the image. Because wifi is unstable when we travel, we connect to it via ethernet or a Wifi hotspot.
2. When the wifi is connected, the green led glows; if the wifi is not connected, the procedure stops. Once the wifi is established, it checks for the MQTT connection; if the MQTT connection is lost, it attempts to reconnect. After connecting to the MQTT, the Arduino attempts to connect to the GPS Module.

3. The GPS module and display to the Arduino board are connected. The GPS module typically talks with the Arduino via serial communication (UART), whereas the display may use a variety of interfaces such as I2C or SPI.
4. The GPS module collects signals from many satellites and calculates location and time information. It sends the latitude longitude to the Arduino.
5. The Arduino board continuously gets GPS information from the module. It compares the current position to the previous position it obtained from the GPS module in order to determine the speed.
6. Depending on the GPS module, the calculated speed is displayed in miles per hour.

7.The speed, latitude, and longitude are determined after everything is connected. Because this is real-time, the values change every second. Here, we've added a led to see if the speed is greater than 20 miles per hour. So, if the speed exceeds 20 miles per hour, the RED Led illuminates. Whether or not the speed exceeds 20 miles per hour, the data of speed, latitude, and longitude is continuously supplied to the MQTT broker, and a MQTT graph is generated.
8. Display Output: Once the speed has been determined, the Arduino delivers the data to the display module. The Arduino connects with the display using the I2C appropriate protocol and sends the speed value to be displayed on the screen.
6. Real-Time Updates: The Arduino continuously updates the speed reading depending on new GPS data received. It repeats the calculation and display process to provide real-time speed information.

MQTT Connection

1. Setting up the MQTT library: The Arduino Sketch includes the MQTT library. Our use of the ArduinoMqttClient.h library facilitates the creation of a MQTT connection.
2. Publish data to MQTT: Once the connection is established, the MQTT library function publishes the data to a specific topic on the MQTT broker. The MQTT publish function is used to convey the data payload.
3. Handle MQTT events: To guarantee dependable communication, it's critical to handle MQTT events while the data is being sent. To handle events like connection status, message acknowledgments, and incoming messages, MQTT libraries typically offer callback functions.
4. Disconnect from the MQTT broker: After completing the project, the MQTT libraries assist us in disconnecting from the broker and cutting the connection.

Libraries Used:

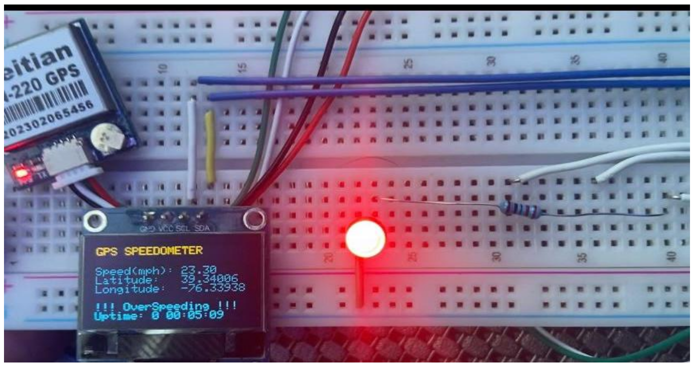
1. Scheduler.h – Allows many tasks to run simultaneously without interfering with one another. Only the Arduino sam and samd architectures (Due, Zero...) are supported. The Arduino can perform many functions at the same time thanks to the Scheduler library. This enables work to be completed

without being interrupted. Because the CPU switches from one task to another, this is a cooperative scheduler. Control is passed between jobs via methods in the library.
2. NTPClient.h – this is a popular Arduino library for acquiring precise time from NTP (Network Time Protocol) servers. It enables the Arduino board's internal clock to be synchronized with the precise time provided by NTP servers via the internet. This library makes it easier to query NTP servers and get time data.
3. WiFiUdp.h – It is an Arduino library that offers capabilities for UDP (User Datagram Protocol) communication over Wi-Fi networks. It enables the Arduino board to transmit and receive UDP packets, allowing us to communicate with other devices or services that use the UDP protocol.
4. stdio.h – The library is used in Arduino for standard input/output operations. It supports printing and reading data through the serial port, which serves as the principal communication link between the Arduino board and your computer.
5. modTinyGPS.h – It is a popular Arduino library that parses and extracts GPS data from NMEA (National Marine Electronics Association) phrases. It provides an easy-to-use interface for interacting with GPS modules and retrieving data such as latitude, longitude, altitude, speed, and other parameters.
6. WiFiNINA.h – It is library is an official Arduino library that is created exclusively for Arduino boards that use the Arduino WiFi Nina firmware. It includes classes and functions for connecting to Wi-Fi networks, maintaining Wi-Fi connections, and conducting network activities through Wi-Fi.
7. ArduinoMqttClient.h – The ArduinoMqttClient.h library is an official Arduino library that provides MQTT (Message Queuing Telemetry Transport) client functionality for Arduino boards. MQTT is a lightweight messaging protocol commonly used for IoT (Internet of Things) applications to enable communication between devices and services.
8. ArduinoJSON.h – The library is an Arduino library that contains functions for processing and producing JSON (JavaScript Object Notation) data. JSON is a popular data interchange format that is both human-readable and simple to deal with. The ArduinoJSON.h library enables Arduino boards to handle JSON data, making it easier to interface with other devices or services that use JSON for data exchange.

VII. Challenges and Limitations of

While Arduino-based GPS speedometers provide a low-cost and versatile alternative for tracking speed using GPS technology, there are certain limitations of using Arduino based GPS Speedometer:
1. GPS Signal Reception: The availability and quality of the GPS signal determine the accuracy and reliability of GPS-based speedometers. GPS signal reception may be hindered or sporadic in particular environments, such as congested urban areas or enclosed places, resulting in mistakes or gaps in speed data.
2. When the GPS module is powered on or has lost its satellite lock, it may take some time to collect satellite signals and establish a stable GPS fix. The Time to First Fix (TTFF) might vary based on factors such as signal strength, visibility of the sky, and the sensitivity of the GPS module. As a result, during the initialization process, the speedometer may provide inaccurate speed readings.
3. GPS units typically send updates at a set pace, such as once per second. While this update rate is appropriate for many applications, it may not accurately reflect quick changes in speed. Furthermore, GPS measurement accuracy might vary, particularly in situations when satellite view is limited or obscured, potentially leading to speed reading discrepancies.
4. Positional Accuracy: While GPS is generally precise at determining location, its positional accuracy can vary based on factors such as satellite geometry, atmospheric conditions, and multipath interference. This variation in accuracy can have an impact on the precision of speed computations, especially for short distances or when traveling at low speeds.
5. Power Consumption: GPS modules can consume a large amount of power, which may impair the overall power efficiency of the Arduino-based speedometer. To save energy, battery-powered applications should consider power-saving measures such as turning down the GPS module when not in use or improving sleep modes.
6. Size and Portability: When compared to dedicated commercial GPS speedometers, Arduino boards and GPS modules, coupled with extra components like as displays and power supply, may result in a considerably bigger and less portable speedometer gadget. This size limitation may limit certain use cases where compactness and portability are critical.
7. Environmental variables: GPS signals can be impacted by environmental variables such as large buildings, mountains, and dense flora, resulting in signal attenuation or signal blockage. In such cases, the speedometer may exhibit decreased accuracy or intermittent speed readings.

VIII. Results

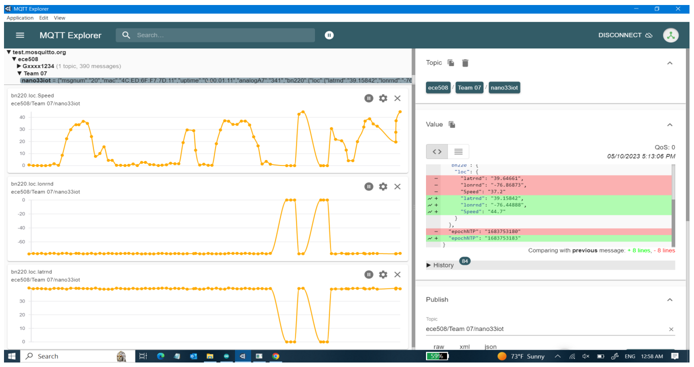


```
GPS SPEEDOMETER

Speed(mph):  23.30
Latitude:    39.34006
Longitude:   -76.33938

!!! OverSpeeding !!!
Uptime: 0 00:05:09
```

Thus the calculated Speed, latitude and longitude values are stored in arduino's memory unit. This data is displayed on oled module in real time . In case of over speeding an caution pop up appears on the oled screen there by glowing Red led.Also all the computed values are updated onto the cloud server using MQTT. The MQTT generates the following Speed, Latitude and Longitude graphs with continuous monitoring.The system can offer important insights on the performance and behavior of cars and drivers by utilizing the power of cloud computing and IoT technology. Among other things, this information may be utilized to improve safety, cut down on fuel use, and optimize routes.



```
nano33iot = {"msgnum":"20","mac":"4C:ED:6F:F7:7D:11","uptime":"0 00:01:11","analogA7":"341","bn220":{"loc":{"latrnd":"39.15842","lonrnd":"-76
```

```
  "bn220": {
    "loc": {
-     "latrnd": "39.64661",
-     "lonrnd": "-76.86873",
-     "Speed": "37.2"
+     "latrnd": "39.15842",
+     "lonrnd": "-76.44888",
+     "Speed": "44.7"
    }
  },
- "epochNTP": "1683753180"
+ "epochNTP": "1683753183"
}
```

Comparing with previous message: + 8 lines, - 8 lines

Topic
ece508/Team 07/nano33iot

IX. Conclusion

Down the lines, Arduino-based GPS speedometers give a versatile and cost-effective method for monitoring speed correctly utilizing GPS technology. These speedometers offer advantages such as easy integration, portability, and real-time speed tracking by exploiting the capabilities of Arduino boards and GPS modules.Arduino-based GPS speedometers deliver accurate speed readings in a variety of locations due to their ability to retrieve latitude, longitude, and velocity data from GPS satellites. They enable the presentation of speed information on a variety of output devices, such as LCD displays or OLED screens, thereby improving the user experience.However, it's vital to examine the constraints and limits of Arduino-based GPS speedometers, such as GPS signal strength, TTFF, power consumption, and potential speed reading mistakes. Addressing these constraints through optimization approaches, alternate positioning systems, and correct setup can assist enhance the speedometer's overall performance and dependability. Despite these challenges, Arduino-based GPS speedometers are nevertheless commonly utilized in applications such as car tracking, sports monitoring, and navigation systems. With the proper implementation and understanding of their limits, Arduino-based GPS speedometers can be a valuable tool for properly measuring and tracking speed in a number of situations.

X. References

1] DIY Arduino GPS Speedometer - Instructables https://www.instructables.com/DIY-Arduino-GPS-Speedometer/ 2] DIY Arduino GPS Speedometer - Instructables https://www.instructables.com/DIY-Arduino-GPS-Speedometer/ 3] Code Reference:Code lecture 15 TaskScheduler https://github.com/arkhipenko/TaskScheduler 4] Arduino GPS Speedometer for Cars - Electronics Hub https://www.electronicshub.org/arduino-gps-speedometer-for-cars/ 5] Du, N., Zhang, J., Yan, Z., Lin, W. (2018). Real-time GPS data collection for mobility monitoring using smartphone and cloud-based infrastructure. Transportation Research Part C: Emerging Technologies, 97, 271-285. 6] Alotaibi, M., Almutairi, N., Alqahtani, M., Alsaleh, M., Alharthi, M. (2020). GPS-based vehicle tracking system. International Journal of Advanced Computer Science and Applications, 11(8), 250-255. 7] Ghosh, S., Dutta, P. (2017). Design and development of an Arduino based weather monitoring and prediction system. International Journal of Computer Applications, 168(1), 1-5. 8] Bhandari, S., Adhikari, R., Ojha, P., Sharma, R. (2018). GPS based vehicle tracking and monitoring system using Arduino. International Journal of Advanced Research in Computer Science, 9(2), 19-22.