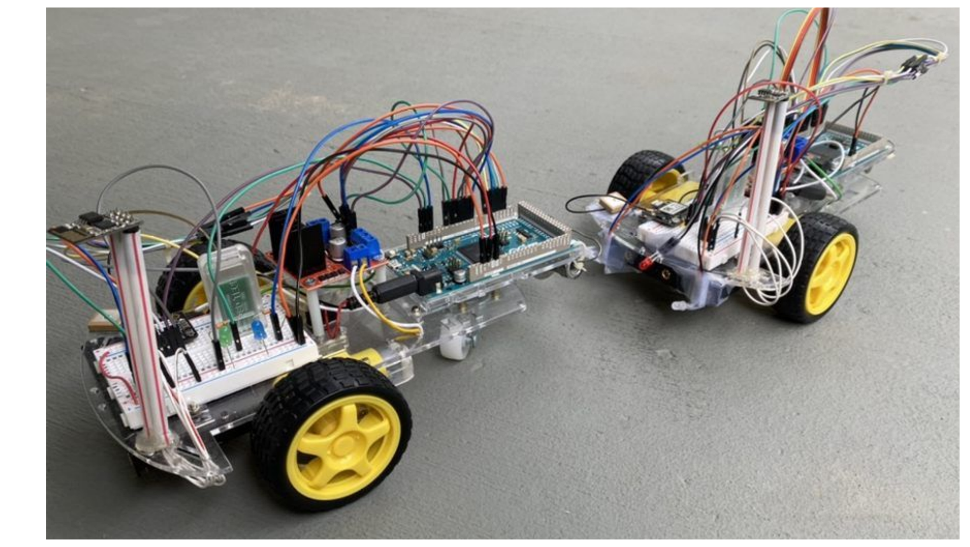
Intelligent Convoy System

Julian Freeman, Brendan Cain, Gurparmvir Singh, Rizwan Siddiqui, Sakib Kamal

Mentor: Dr. E.F. Charles LaBerge

University of Maryland, Baltimore County



ABSTRACT

An Intelligent Convoy System (ICS) permits a series of robotic vehicles to autonomously follow a lead robotic vehicle. Intelligent Convoy Systems can be implemented and utilized in many different applications, ranging from warehouse transportation to military operations. Our ICS allows for one user to control the lead vehicle via an iPhone app (ArduinoBlue™) . The lead vehicle will communicate with the follower vehicles by continuously transmitting streams of data, giving the followers enough information to process and follow the lead vehicle. For this project, our team has assembled two robotic vehicles, one lead and one follower, and developed a design that promotes scalability, allowing for up to sixteen follower vehicles. The lead vehicle will gather data including its current location, and send out that data in a packet to the follower vehicles via a WiFi link. The follower vehicles will receive the packet of data, unpackage it and process the data using algorithms to follow the lead vehicle. This communication based system explores the means of wireless communication between two vehicles and uses relevant technologies such as global positioning systems (GPS), Wi-Fi data transfers, radio links and the use of algorithms to perform calculations involving relative speeds and the relative distances between two vehicles. In the process we have assimilated external communication systems such as GPS to our own wifi access points for hosting and connecting to follower vehicles to leader as a means of internal communication within the system. We used trigonometric properties to develop a formula to calculate relative distances with respect to each vehicle, determine the the bearing of the following vehicle and created our own algorithm to use those distances and the GPS coordinates to create a checkpoint system where a lead vehicle is followed by two or more follower vehicles.

SOFTWARE

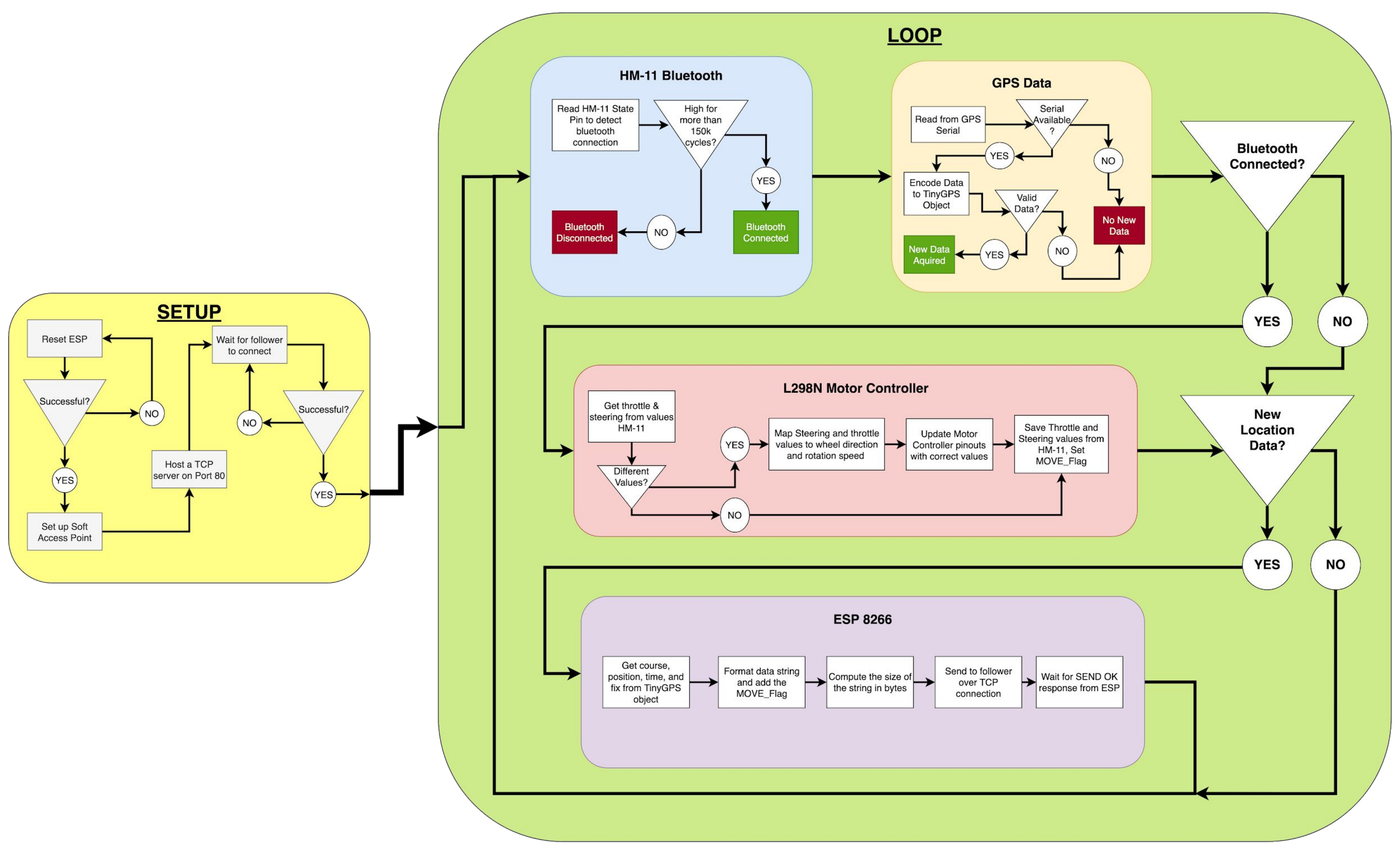


Figure 1: ICS Leader Software Architecture

The software for the intelligent convoy system was coded in the Arduino open source language which is a C++ based coding language. An Arduino program is made up of a setup function that runs once at the beginning of every program and a loop function that runs continuously until the system is powered off or reset. The follower and leader software are two separate programs that are flashed to the memory of the Arduino Dues and implement the functionality of the respective vehicles. The mode of communication between the two vehicles is a TCP link that is setup via the ESP8266 WiFi modules. This is implemented in the setup function by sending AT commands to the WiFi modules through a serial port. These AT commands setup/connect to a WiFi access point so that a TCP connection can be established. In the leader's loop function, a GPS signal is acquired and the NMEA data coming in through one of the due's serial ports is read, parsed, and updated by the TinyGPS library object. Then if a Bluetooth phone connection is established, steering and throttle values are acquired through the ArduinoBlue library and mapped to values that are read by the motor controller. Finally, all location data is formatted into a packet along with a movement flag that signifies the movement of the lead vehicle and if its Bluetooth connection was lost. This packet is sent to the follower over the ESP8266's TCP connection. In the follower's loop function, a GPS signal is acquired, and the location data is updated in a fashion similar to that of the leader. Then the follower's due attempts to read in a packet of information that was sent from the leader to the follower's ESP8266. If this packet is deemed valid, then the leader's location data is saved in a custom C++ object called a checkpoint. These checkpoint objects allow the follower to easily access the data received in past packets. Each checkpoint received is then saved in a rotating buffer. This buffer implements the follower's ability to follow the leader's path instead of just going to its most recent location as the crow flies. The follower keeps track of which checkpoint it is currently trying to reach and calculates its remaining distance and course of direction with its live location data and the data saved in the current checkpoint object. Finally, using this information, the follower can calculate its bearing (shown below) and steer towards the checkpoint using the motor driver. Once a follower has reached a checkpoint, the object is removed from the buffer and it continues along the leader's path towards the next checkpoint.

Bearing Calculation

$$\theta_2 = Course \quad \theta_1 = Heading(CurrentDirection) \quad \Delta\theta(\phi) = \begin{cases} \phi \leq -180, & \phi + 360 \\ |\phi| < 180, & \phi \\ \phi \geq 180, & \phi - 360 \end{cases}$$

$$\phi = \theta_2 - \theta_1 \qquad \Delta\theta(\phi) < 0 \rightarrow Turn\ Counterclockwise \qquad \Delta\theta(\phi) > 0 \rightarrow Turn\ Clockwise$$

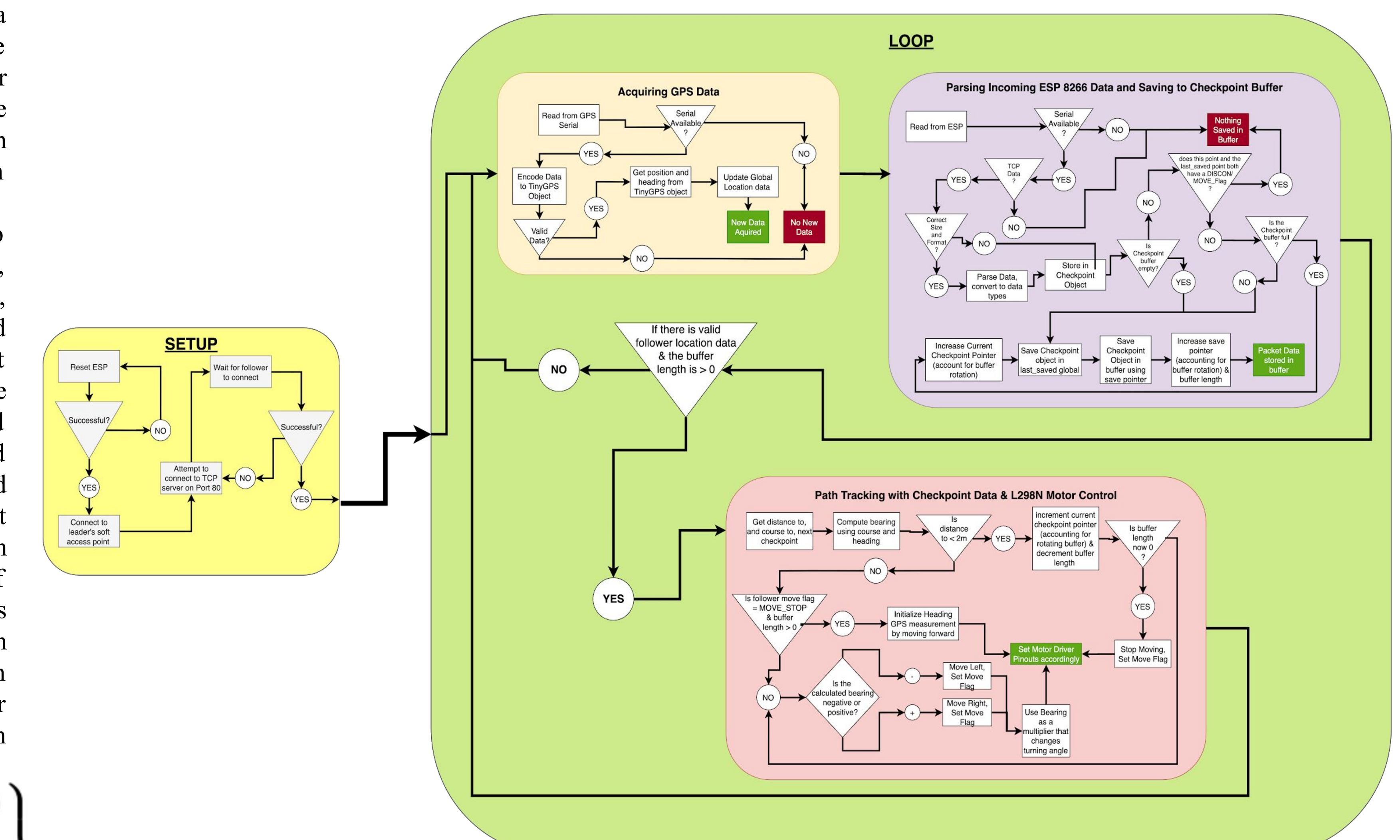


Figure 2: ICS Follower Software Architecture

HARDWARE

Each vehicle has one of the following components

- Arduino Due Microcontrollers: Processes data. Operates HM-11, Neo-6m, ESP8266 & L298N. Powers HM-11, Neo-6m & ESP8266.
- HM-11 Bluetooth Modules: Receives phone input and transmits it to the Arduino
- Neo-6m GPS Modules: Receives GPS signals and transmits them to the Arduino
- ESP8266 Wi-Fi Modules: Receives data from preceding vehicle, and transmits data to succeeding vehicle
- L298N Motor Drive Controllers: Controls two DC motors with data from the Arduino
- 18V Battery Packs: Powers Arduino and L298N
- Car Chassis with two DC Motors: The vehicle mount

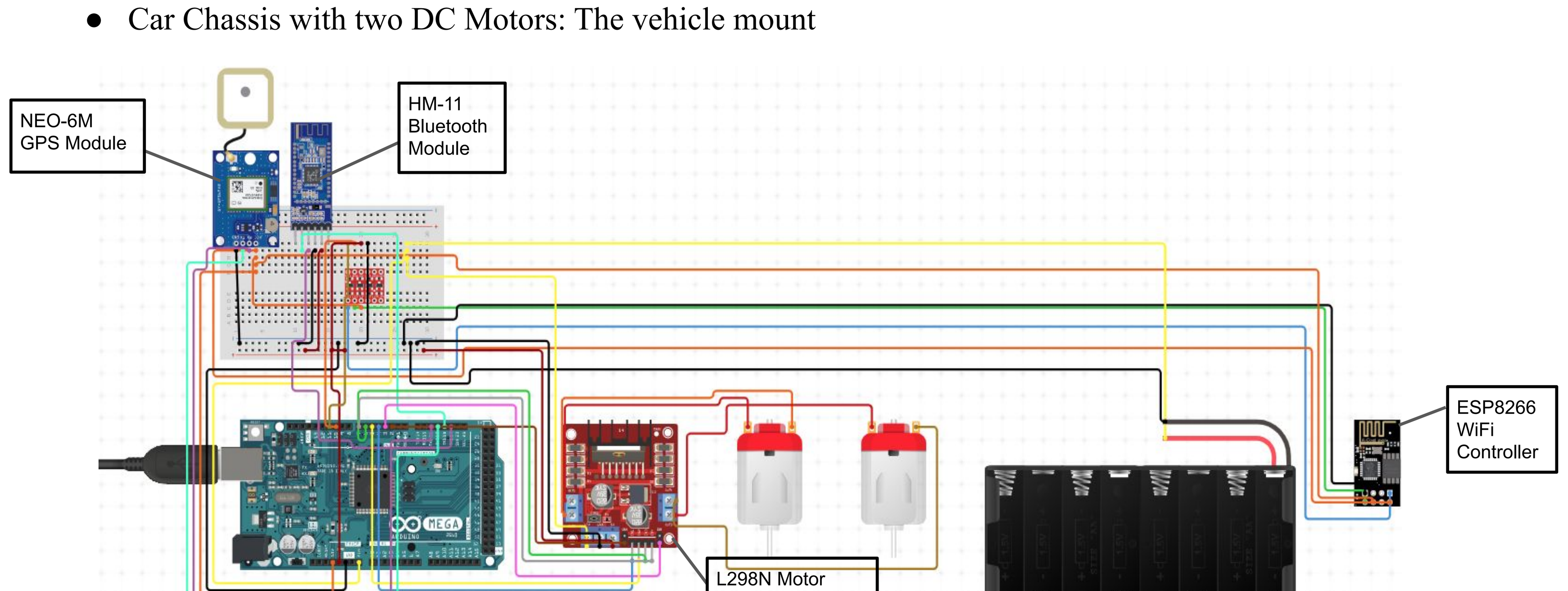


Figure 3: ICS Hardware Configuration

TESTING AND RESULTS

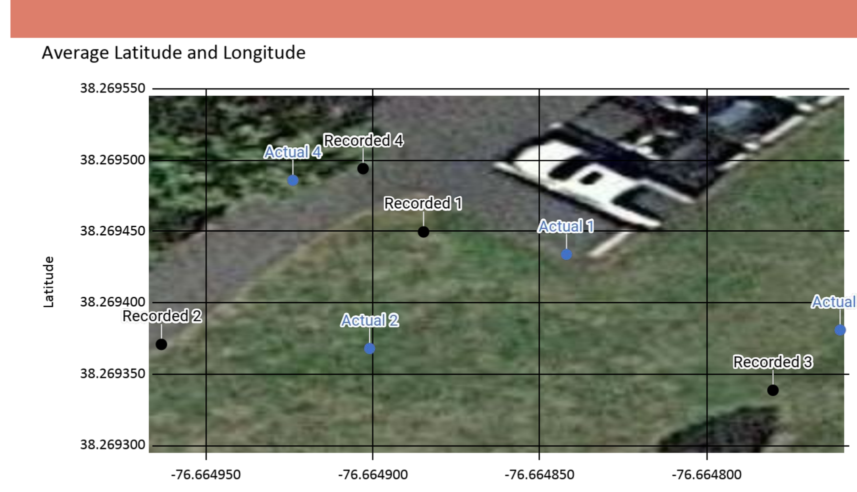


Figure 4a: GPS Discrepancy

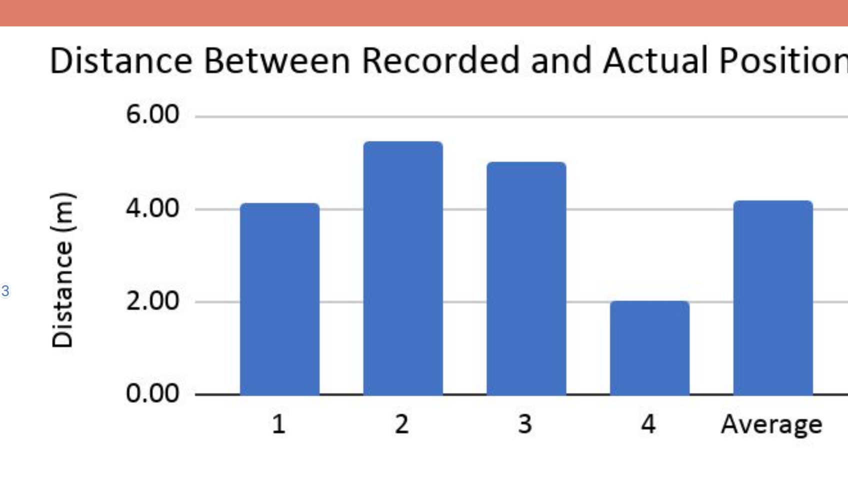


Figure 4b: GPS Discrepancy

The accuracy of the GPS modules proved to be the weakest link in this design. An inaccurate reading from either the follower or leader GPS would cause the follower to attempt to move towards a location that was not near the leaders actual path location. Another issue that proved difficult to solve was determining the heading of the following vehicle. The TinyGPS library can calculate this fairly accurately using the most recent position and the last position recorded. However, calculation is only accurate while the vehicle is moving. Since this system relies on a continuous flow of accurate information, an electronic compass would be necessary to record an accurate reading of the heading while stationary.

To fully understand the accuracy of the NEO-6M a test was done that compares recorded latitude and longitude coordinates to the known coordinates a specific location. The average of difference in meters of these two recordings are shown in Figure:-4a. It can be observed from Figure:- 4b that the discrepancy between test coordinates and actual coordinates differ by a maximum of about 5m. While this is to be expected from a low priced GPS module, it is nowhere near the actual performance capability of US satellites according to gps.gov.

SYSTEM DESIGN DIAGRAMS

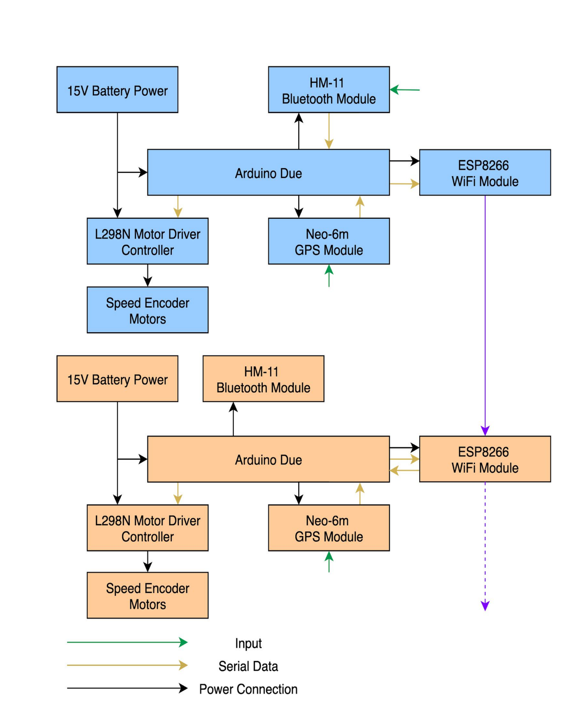


Figure 5: ICS System Architecture

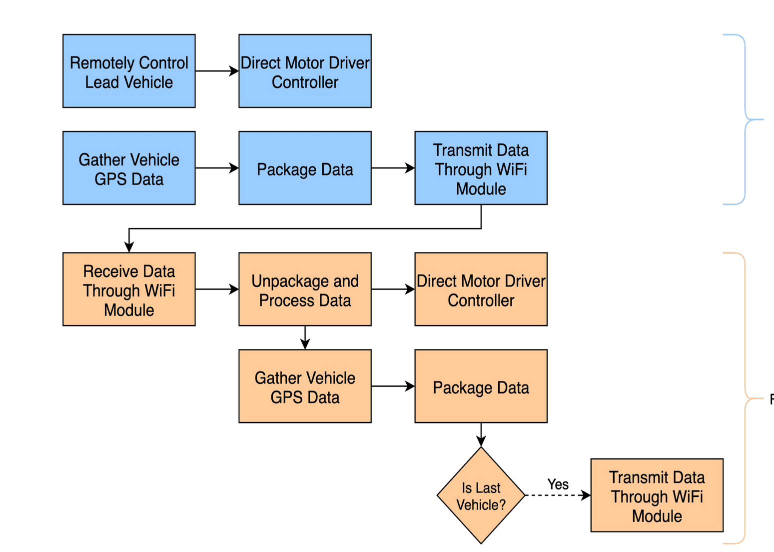


Figure 6: ICS Functional Flow Diagram

REFERENCES

ArduinoBlue™ Guide

AT Command Reference (github.com)

ESP8266 Datasheet (sparkfun.com)

US GPS Satellite Accuracy(GPS.gov)

Arduino IDE Reference

AT Instruction Set (espressif.com)

TinyGPS Library Reference (arduiniana.org)