International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319-6734, ISSN (Print): 2319-6726
www.ijesi.org ||Volume 10 Issue 8 Series II || August 2021 || PP 27-33

Design and Testing a Localization System on an Arduino Car

Alexander Neal, Andrew Oswalt, Lucas Swanson, and Qin Hu
School of Engineering, Eastern Michigan University, Ypsilanti, Michigan, United States of America

ABSTRACT: *The Localization System Design described in this paper as a basic framework for system design for autonomous vehicles. Using satellite data from a Global Navigation Satellite System and vehicle inertial/positional data from an Inertial Measurement Unit, we produce a program that can travel to predetermine Global Positioning System coordinates. This design aims to examine methods to collect accurate data in standardized units for use in calculations needed for the system's logic. The system is designed using a Python program supplied by a computer, provides a GNSS error of 25cm and navigates in respect to magnetic North with a manual offset. Autonomous vehicles using this design can navigate basic courses with preset GPS coordinates without obstacles. Future expansion through additional sensors is available without major adjustments to the present design.*

KEYWORDS – *Autonomous Vehicle, Localization, Global Navigation Satellite System (GNSS), Inertial Measurement Unit (IMU).*

Date of Submission: 16-08-2021 Date of Acceptance: 31-08-2021

I. INTRODUCTION

Autonomously controlled vehicles are a growing field in the automotive industry. Companies like Tesla, with their lineup of electric semi-autonomous vehicles[1], and General Motors, with their Cruise division of fully autonomous test cars[2], are working hard to make autonomous personal transportation a reality. In a future of completely autonomous transportation for people and products, accidents due to human error are practically nonexistent, creating a much safer environment on our ever-expanding network of highways and roads[3]. Another key aspect to a future with fully autonomous transportation is machine learning and its ability to increase safety and efficiency in time spent on the road, energy being wasted sitting in traffic, and pedestrian avoidance[4].

The localization system for a small Arduino robot car with similar control output to that of the Golf Cart planned for development. This is done for ease of testing and troubleshooting with minimal environmental risk. The Autonomous Golf Cart is a long-term project from the GameAbove College of Engineering & Technology at Eastern Michigan University. Our role within this project is to create an autonomous taxi service for students around campus. This project is a beginning framework to create a basic autonomous system for Global Positioning System based navigation. Due to the planned control systems of the Golf Cart, adjusting the system to the cart will require minimal effort in the future.

II. SYSTEM ARCHITECTURE

The localization system is designed entirely within Python 3.8 and Arduino IDE. Both the IMU and GNSS provide development kits and libraries compatible with Python 3.8 that allow for easy calibration and fast data collection. This requires a computer to transmit data and run the program. Our software performs independent calculations needed for pathing and communicates to the Arduino through bytes with low latency and memory usage.

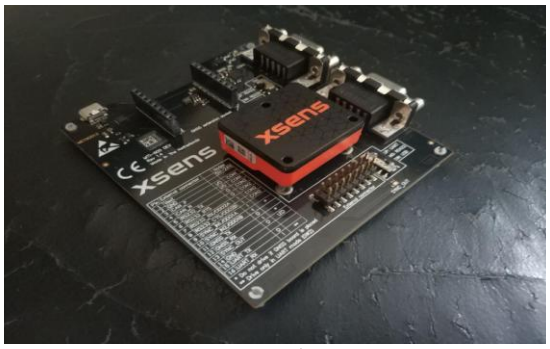


Figure 1 An Xsens Mti 630 Altitude and Heading Reference System mounted onto an Xsens MTi-600 development kit board.

The Inertial Measurement Unit or IMU is an Xsens MTi 630 Attitude and heading reference system (AHRS). This device can provide vehicle directional and acceleration data through collaboration between gyroscopes, accelerometers, and a magnetometer. Acceleration is collected in m/s^2 with accuracy of 15μg. Orientation data degrees with an accuracy of 0.2° for roll/pitch and 1.5° for yaw. Our measurements obtained are collecting in degrees and m/s^2 respectively. Using the Xsens development API for Python 3.8, we are able to initialize and communicate with the IMU at a frequency of 100Hz through an xbus USB connection to a Windows laptop.

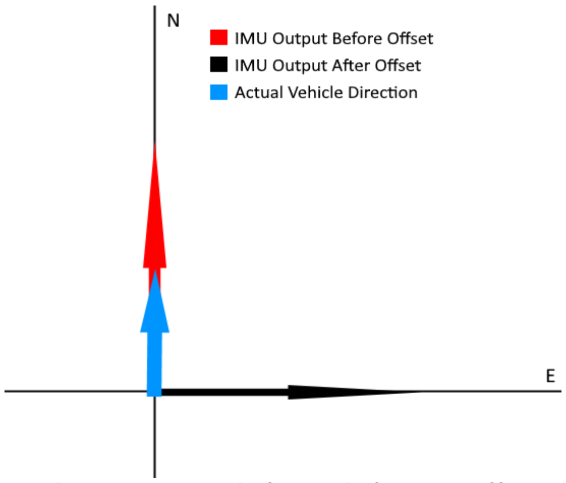

N
🟥 IMU Output Before Offset
⬛ IMU Output After Offset
🟦 Actual Vehicle Direction
E

Figure 2 Coordinate plane comparison before and after IMU offset calibration

The IMU is configured to orient its y-axis/pitch as the cardinal direction North using a +90 degree offset. This adjustment is visible in Figure 2. Our device outputs vehicle orientation to the Python program for comparison. For consistency, the IMU will be located at the center of our testing vehicle to ensure center of mass orientation to prevent improper readings due to torque. Maintaining a high frequency of checks and adjustments will allow an accurate path traveled for the cart with minimal downtime.

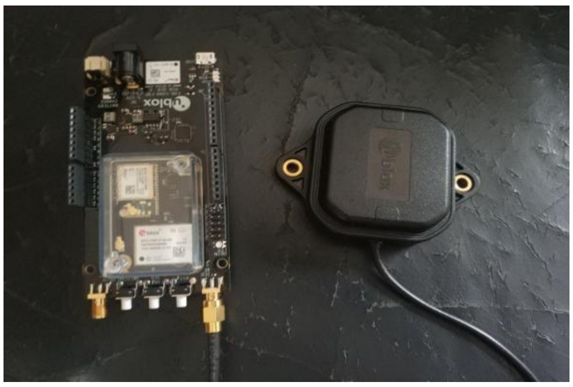


Figure 3 A u-blox CO99-F9P with ZED-F9P Global Navigation Satellite System receiver attached to its satellite.

The Global Navigation Satellite System or GNSS we selected is the u-blox ZED-F9P attached to a CO99-F9P. This device is configurable to obtain GPS coordinate signals at a rate of up to 10MHz from a cold start of 25s. We are operating the device in Satellite-based Augmentation System (SBAS) mode with an accuracy of 1.0m and a navigation update rate of 8Hz. For more accurate readings, we would use Real-time kinematic positioning (RTK) mode for an accuracy of 0.01m. However, it requires a second ZED-F9P for a base-rover system. We adjust our system accordingly for this increased inaccuracy.

The precision and accuracy of the GNSS device are very important to our project. If the device was to be too far off of the true coordinates, the car would travel to an improper location, and if implemented to the golf cart, could potentially harm a passenger. According to the device specifications, the device should be accurate to within two meters, which if accounted for and working correctly, should be accurate enough to guide the cart to the predetermined path coordinates.

To test the precision of the GNSS, twenty different locations were tested at least 100 meters apart, and twenty iterations of the coordinates being reported by the GNSS were logged. We then recorded the coordinate location reported by Google Maps by placing a pin at our location using the satellite map overlay and visual aids instead of relying on the poor accuracy of the GPS in our phone. Shown below are graphs representing the standard deviation of the coordinate sets and comparing the difference between the Google Maps and GNSS coordinates.

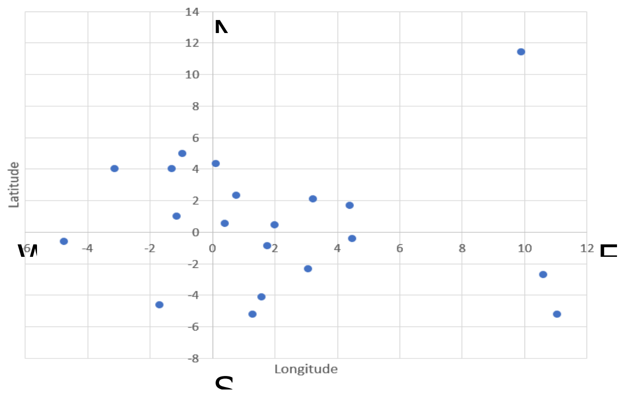


Figure 4 Difference between Google Maps and GNSS coordinates.

Shown in Figure 4 is the average difference between each set of coordinates and the coordinates given by Google Maps. The closer the value is to zero, the more accurate the coordinates. Some error occurs in the Google Map coordinate gathering due to poor accuracy of the GPS in our phone and being forced to resort to dropping pins in the map for ourselves.

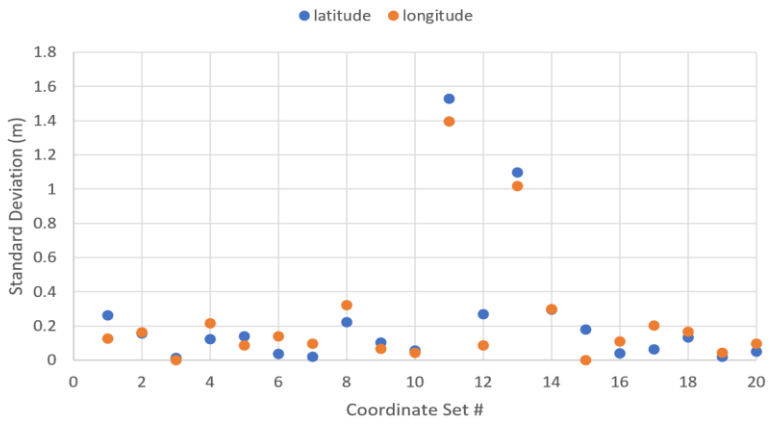


Figure 5 Graphing of the standard deviations in the coordinate set between GNSS output and expected GPS data.

Shown in Figure 5 is the standard deviations of the twenty recorded coordinates in each coordinate set. The closer the value is to zero, the more precise the coordinates. According to the values in the graph, the GNSS unit is precise to an average of 25 cm. This far exceeds the expected value of two meters.

III. DESIGN DETAILS

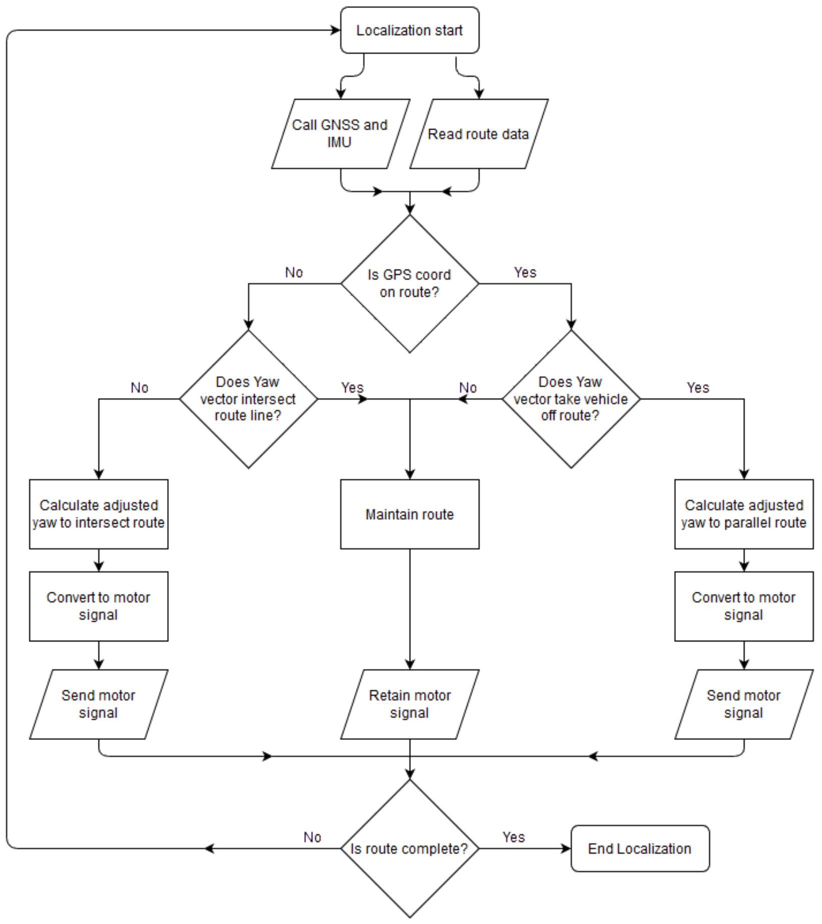


Figure 6 A Flowchart for localization using a GNSS and IMU.

The flowchart of the GNSS and IMU localization system is seen in Figure 6. When the program runs, it first calls the data from the GNSS, IMU, and the coordinates of the predetermined path. It then checks if the devices are on the correct path. If they are, it checks for the angle between the path coordinate and the devices and chooses how to respond based on that answer. It then checks for completion. If yes, it ends that portion until a new path coordinate is given. If no, it reruns the code until the answer is yes.

This concept would utilize a GNSS board and an IMU to determine where the Arduino car is and where its facing to determine how the golf cart should move in order to reach its final destination. The computer uses a GPS coordinate map to create a path for the cart to follow and the IMU would allow the computer to make minor adjustments to keep on track. The system allows for efficient and direct travel but is prone to obstacle interference due to lack of sensors.

Using the output of the GNSS and IMU along with a set of predetermined coordinates, we develop a system to determine the distance and direction of the path compared to the current location of the car. To do this, we wrote a program in Python 3.8 using the Anaconda distribution platform. For testing purposes, we ran the code on a Windows 10 computer with text files replacing the inputs from the IMU and GNSS. In the final version, we added the code for the IMU and GNSS to the pathing algorithm.

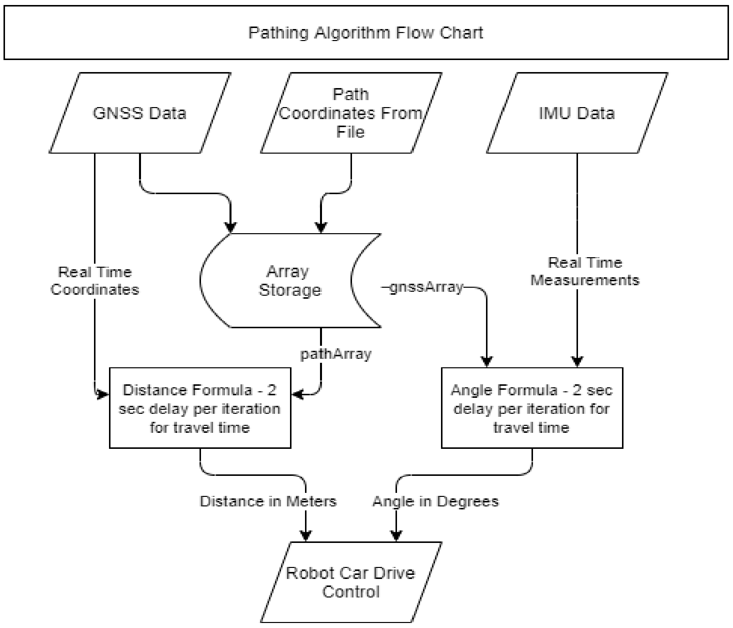

Pathing Algorithm Flow Chart
GNSS Data
Path Coordinates From File
IMU Data
Array Storage
Real Time Coordinates
gnssArray
Real Time Measurements
pathArray
Distance Formula - 2 sec delay per iteration for travel time
Angle Formula - 2 sec delay per iteration for travel time
Distance in Meters
Angle in Degrees
Robot Car Drive Control

Figure 7 Pathing algorithm flowchart using IMU and GNSS data to determine car movement.

The program calls the data from the GNSS and the IMU and reads a text file for the path coordinates as seen above. Comparing the GNSS and stored path coordinates, the program determines the distance and angle between the two points. Beyond 2 meters, the program calls the IMU data and determines which direction the car turns to parallel the GPS comparison and vehicle pitch vectors. Within a 2-meter range, the program stops the car in place, calls for the next path coordinate, and repeats the process until the path is complete.

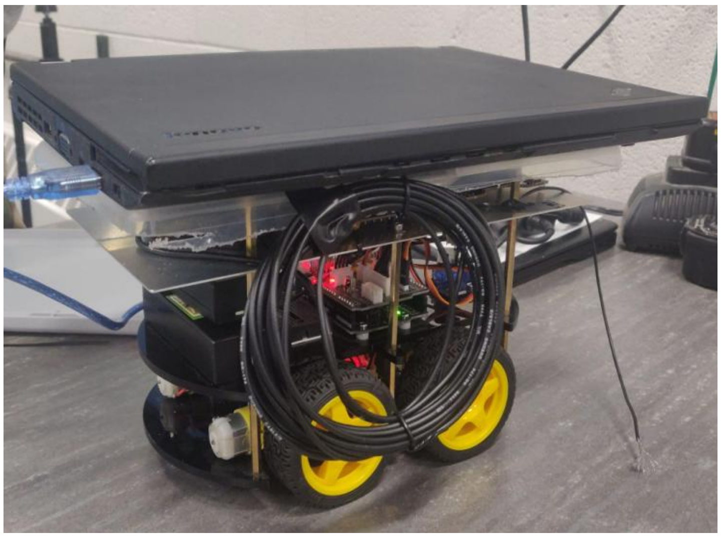
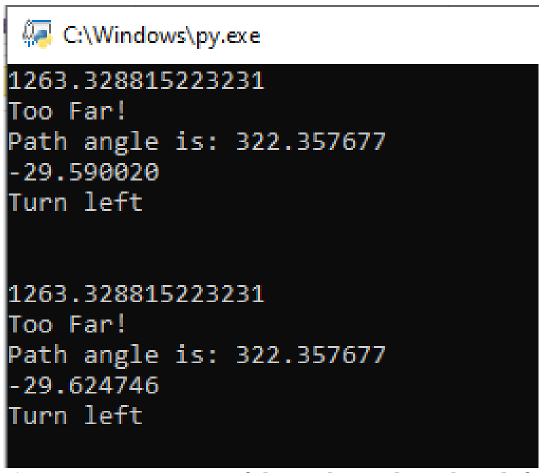


Figure 5 Final Assembly of the Arduino Car

The Arduino car is assembled using a Windows computer, the GNSS and its satellite, and the IMU connected via USB and powered by an external battery as shown above. The laptop is placed on top and the GNSS and IMU are mounted on the aluminum plate underneath the laptop. Below those two levels are the Arduino, motor drive board, battery packs for both the GNSS and the Arduino car. Mounted on the side is the extra cabling for the GNSS antenna. The wire hanging off the back of the car (right side of the image) is a grounding wire to help reduce interference.

IV. RESULTS



```
C:\Windows\py.exe
1263.328815223231
Too Far!
Path angle is: 322.357677
-29.590020
Turn left


1263.328815223231
Too Far!
Path angle is: 322.357677
-29.624746
Turn left
```

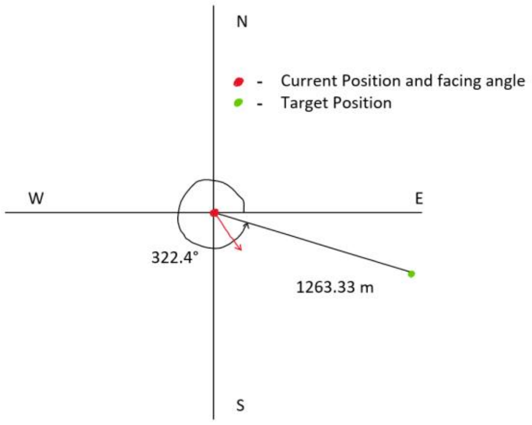


Figure 7 Test output of the pathing algorithm (left) and a coordinate plane representation of the systems decision making.

The system can navigate through multiple GPS coordinates stored onto the computer's memory without user input beyond powering the machine. No obstacles were present for the car to navigate around. Our localization estimates the distance between the car and the next point successfully and compares the vehicle's pitch to the angle needed for the quickest straight-line path. Figure 7 shows a debugging window of the system listing the distance from the next point in mm, the needed angle, and the vehicle angle in degrees.

Detailed testing revealed the system is effective and able to communicate efficiently with the Arduino for effective control. The computer in testing was held due to improper weight calculations. Further revisions should expect to implement a smaller form factor computer if expected to test with the Arduino car. However, the system is lightweight and effective enough for future expansion onto the project with addition for object and obstacle avoidance for more practical implementation.

V. CONCLUSION

Our localization system using GNSS and IMU data allows for basic, linear navigation with predetermined paths. This system operates with inaccuracies of +/- 25 cm and a consistent offset for the IMU. The project can be scaled to a more developed project for the overall golf cart project. Future developments may include a full pathing algorithm paired with additional data collection devices such as the LIDAR Puck or vehicle prediction using acceleration data from the IMU. It is recommended that this system is only used in controlled, testing environments in its current state.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Mr. John Earl and the School of Engineering for their support and dedication to the golf cart project.

REFERENCES

[1]. "Autopilot." *Tesla, Inc*, www.tesla.com/autopilot.
[2]. "We're Cruise, a Self-Driving Car Service Designed for the Cities We Love." *Cruise*, www.getcruise.com/.
[3]. Kaplan, Scott, et al. "The Future of Autonomous Vehicles: Lessons from the Literature on Technology Adoption." Wiley Online Library, John Wiley & Sons, Ltd, 4 Dec. 2019, onlinelibrary.wiley.com/doi/full/10.1093/aepp/ppz005?casa_token=5x2HZtZPqX8AAAAA%3ALXhHfHBeapk4cEZXIsvtnPWQtkeBNIUeFFKsM2mOOrERFn3xGEvie9OnaCOuy91-VWAKEhwDRS6UAIlO.
[4]. Navarro, Pedro J., et al. "A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data." MDPI, Multidisciplinary Digital Publishing Institute, 23 Dec. 2016, www.mdpi.com/1424-8220/17/1/18/htm.
[5]. Ami Woo, Baris Fidan, and William W. Melek. "Localization for Autonomous Driving." *Handbook of Position Location: Theory, Practice, and Advances, Second Edition,* 2019. *University of Waterloo.*
[6]. A. Leon-Garcia. "Probability and Random Processes for Electrical Engineering, 2nd Ed., Addison Wesley"
[7]. Roy. D. Yates and David. J. Goodman. "Probability and Stochastic Processes, A Friendly Introduction for Electrical and Computer Engineers, 2nd Ed., Wiley"
[8]. Peter Lehnér. "Get position from GPS using Python." 9 Oct. 2016, https://www.youtube.com/watch?v=_DuMjcl52BU.
[9]. Jan V. Sickler. "GLOSNASS Satellites and Signals." PennState, College of Earth and Mineral Sciences, Department of Geography, https://www.e-education.psu.edu/geog862/node/1874.
[10]. Katie Spence. "Where to Sell Scrap Circuit Boards: 8 Scrap Buyers Listed." *First Quarter Finance*, 29 Sep. 2018, https://firstquarterfinance.com/where-to-sell-scrap-circuit-boards/.

[11]. International Committee on Electromagnetic Safety. "IEEE C95.3-2002 - IEEE Recommended Practice for Measurements and Computations of Radio Frequency Electromagnetic Fields With Respect to Human Exposure to Such Fields, 100 kHz-300 GHz." *IEEE Standards Association*, 11 January 2003, https://standards.ieee.org/standard/C95_3-2002.html, updated 12 June 2008.

[12]. SMC/SC Standards Committee. "IEEE 7010-2020 - IEEE Recommended Practice for Assessing the Impact of Autonomous and Intelligent Systems on Human Well-Being." *IEEE Standards Association,* 1 May 2020, https://standards.ieee.org/standard/7010-2020.html.

[13]. RS/SC IEEE Reliability Committee. "IEEE 1413-2010 - IEEE Standard Framework for Reliability Prediction of Hardware." *IEEE Standards Association*, 9 April 2010, https://standards.ieee.org/standard/1413-2010.html.

[14]. Melanie Zanona. "How driverless cars can reduce pollution." *The Hill*, 24 Oct. 2016, https://thehill.com/policy/transportation/302550-how-driverless-cars-can-reduce-pollution#:~:text=Autonomous%20vehicles%20also%20have%20the%20potential%20to%20reduce,at%20high%20speeds%20in%20order%20to%20mitigate%20traffic.

[15]. "Avoiding crashes with self-driving cars: Today's crash-avoidance systems are the mile markers to tomorrow's autonomous vehicles." *Consumer Reports*, Feb. 2014, https://www.consumerreports.org/cro/magazine/2014/04/the-road-to-self-driving-cars/index.htm#:~:text=Autonomous%20vehicles%20could%20help%20reduce%20crashes%20caused%20by,greater%20mobility.%20But%20are%20we%20ready%20for%20them%3F

[16]. "SBAS Fundamentals" *European Space Agency Navpedia*, Jul. 2018, SBAS Fundamentals - Navipedia (esa.int)

[17]. "ZED-F9P u-blox F9 high precision GNSS module data sheet", *u-blox*, Jun. 2020, ZED-F9P Data sheet (u-blox.com)

[18]. "GPS Sentences | NMEA Sentences | GPGGA GPGLL GPVTG GPRMC", *RF Wireless World*, 2012 https://www.rfwireless-world.com/Terminology/GPS-sentences-or-NMEA-sentences.html

[19]. Jacobson, Brandon. "Get USB GPS Coordinates with Python/NMEA to Decimal Conversion | #110 (Iron Man Suit #1)." 13 Nov. 2020, https://www.youtube.com/watch?v=I4NPqFqf-Po.

[20]. New Xsens MTi Software Provides Open-source API to Support Popular Embedded Development Platforms." *Xsens*, 22 Feb. 2019. https://www.xsens.com/press-releases/new-xsens-mti-software-provides-open-source-api-to-support-popular-embedded-development-platforms

Alexander Neal, et. al. "Design and Testing a Localization System on an Arduino Car." *International Journal of Engineering Science Invention (IJESI)*, Vol. 10(08), 2021, PP 27-33. Journal DOI- 10.35629/6734