GPS GUIDED AUTONOMOUS ROBOT

A PROJECT REPORT

Presented to the Department of Electrical Engineering

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

Committee Members:

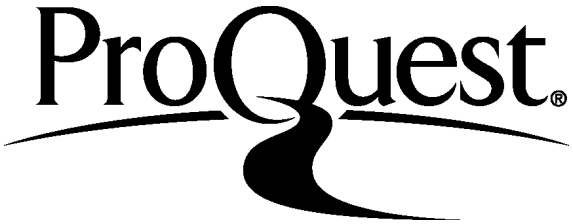
Fumio Hamano, Ph.D. (Chair)
Bahram Shahian, Ph.D.
James Ary, Ph.D.

College Designee:
Antonella Sciortino, Ph.D.

By Aniket D. Paradkar

B.E., 2013, Rashtrasant Tukadoji Maharaj Nagpur University, MH, India

May 2016



ABSTRACT

GPS GUIDED AUTONOMOUS ROBOT

By

Aniket D. Paradkar

May 2016

This project focused on building a GPS controlled 6-wheel autonomous robot. It is a self-guided autonomous robot, which can be maneuvered with the help of GPS module and compass together interfaced with the microcontroller Arduino Mega. The 6-wheels of the robot are interfaced with monster motor shield and then connected to the Arduino Mega. The speed of the robot is controlled using PWM signals sent from the Arduino board. When the robot starts, it locates its current position using the GPS module. The destination coordinates are already given in the code. Once the current location is fixed, it calculates the distance and heading between the two points. The compass module tells the current heading of the robot. The final heading is calculated by taking the difference between actual heading and current heading. With the help of final heading angle, the robot moves towards its desired location. As the robot moves close to the destination the distance reduces. The minimum distance is predefined as 5 meters in the algorithm since the precision of GPS module is within the range of 5 to 6 meters. Once the distance is less than 5 meters the robot stops, assuming it has reached to the destination location.

The purpose of building this robot was to guide the robot to multiple locations autonomously with the destination locations predefined in the algorithm. To maneuver the robot to the multiple locations it is very important to calculate the accurate distance and heading. For this project, the main task was to design an algorithm that can calculate the exact distance

between any two locations and guide the robot in the proper direction. The motors used for this project have high torque and the updating speed of the GPS module is slow. It was very important to keep the speed of the motor very low and change the speed of the motor only when there was a need to change the direction of the robot. The algorithm designed was able to fulfill these tasks and guided the robot to multiple locations and reach the final destination.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The objective of this project was to build a GPS guided 6-wheel autonomous robot that can move from one location to another using GPS module. In the previous development of this robot, there was a problem with the distance calculation and the accuracy of the GPS signals. In this project, an accurate distance formula has been implemented to calculate the distance between any two points. Also, the robot is now capable of going to multiple locations, which are predefined in the algorithm. This was one of the major accomplishments of this project as it was not implemented in the previous development of this robot. Moreover, the control system of this robot is improved. This robot can now be driven with multiple speed depending upon the GPS signals and the distance between two locations.

The operation of the robot starts by acquiring the current location and the current location is stored in variables called longitudes and latitudes. Once the current location is fixed the robot calculates the distance from its current location to the first target location. The direction of the robot is decided based on the output from the GPS module, compass and the target location. While travelling towards the target location the GPS module updates the current location and the distance is calculated on a regular basis. Once the robot reaches the first target location it stops for some time and then moves towards next location coordinates by again calculating the required parameters. The robot finally stops once it reaches its final location.

Autonomous mobile robots are self-operated vehicles that do not require any command from the operator. The movement is defined before the robot starts and it navigates according to the predefined coordinates. Autonomous mobile robots have various applications. Surveillance, military and agriculture purpose are some of the most common fields that deploy autonomous

vehicles [1]. There is a vast development going on in order to make the autonomous robots more robust and capable of doing any task. In military operations, there is a huge risk of injury while diffusing a bomb. The autonomous robot can play a very big role in saving the life of the soldiers during wars.

In this project, the code is implemented in such a way that the robot can go to various locations with the help of predefined coordinates. The GPS is not very accurate, but from the readings taken, it was found that the accuracy is about 5-6 meters. Once the robot reaches close to the destination coordinates it stops and calls next coordinates and then proceeds according to the results from the calculation. While traveling, the locations of the robot are continuously updated to keep the robot on the desired path. Once the robot reaches close to the final destination, it slows down and finally stops.

Motivation

This project is about an autonomous robot guided with the help of GPS module, compass, Arduino Mega and a 6-wheel chassis interfaced with 6 DC motors. The robot locates its current position using a GPS module. After the current location is fixed the algorithm designed for the operation of the robot calculates the distance between the current and the target location and also the required heading. These parameters guide the robot to the destination. The most important task was to calculate the accurate distance between any two locations on the surface of the earth using the distance formula. The algorithm designed for distance formula is very complex and one of the major innovative aspects of this project. There are various distance formulas available which are used in navigation, but implementing the most accurate one for this project was the most important task. Another innovative aspect of this project is that the robot can travel to multiple locations. The destination coordinates can be defined in the algorithm in the form of

latitudes and longitudes. The algorithm automatically switches to the next location coordinate once the robot reaches close to the first location coordinate. Furthermore, the algorithm is designed in such a way that the robot can change its speed depending upon the destination and the heading calculated from the formula. This makes the robot more innovative, more accurate and robust to find its way automatically without human intervention. The design of the robot, the tires, the chassis and the powerful DC motors makes the robot capable of working on any type of surface. All these new developments in this project make the robot an independently self-driven vehicle.

CHAPTER 2

LIST OF COMPONENTS AND TECHNICAL SPECIFICATIONS

This project requires many electronic components. Following is the list of the components and also their technical specifications.

1. Arduino Mega 2560 R3 (ATMega 2560)
2. Sparkfun Venus GPS with SMA Connector
3. Antenna GPS Magnetic Mount SMA
4. HMC5883L Module Electronic Compass
5. Sparkfun Monster Moto Shield
6. 6 Wheel Robot Chassis and DC motors
7. 7.4V Battery for power supply to motors.

8.9V Battery to power Arduino Mega

Arduino Mega 2560 R3

For this project Arduino Mega microcontroller, an Arduino component is used and it is shown in figure 1 [2][3]. Arduino is an open source computer tool that comes with both hardware and software platform. There are various components of Arduino in the market. The main purpose of the Arduino is to connect different electronic devices that can communicate with each other and perform various operations in the real world. The Arduino platform has a build-in integrated development environment called IDE. It helps to program any project with various other devices connected to it. It supports C and C++ programming language which are the basis of all other programming languages. I have selected this board because of the open source software that is related to the board and its flexibility for programming. It is very useful to

learn programming from the basics and advance further. The board comes with an onboard microcontroller and also various input ports to connect different components to it.

FIGURE 1. Arduino Mega 2560 R3.

The Arduino board has a USB port that is used to connect it with computers very easily. It has onboard LED that becomes ON when connected to a power source. This board can be powered up using an AC to DC adapter or a battery. This board comes with 54 digital input/output pins of which 15 are used as PWM outputs, 16 analog pins, 4 USARTs hardware serial ports, a power jack, a USB connection, a 16 MHz Crystal oscillator, ICSP header and a reset button. Table 1 below shows the technical specifications of Arduino Mega [3][4].

TABLE 1. Technical Specifications of Arduino Mega 2560 R3

Microcontroller	ATMega 2560
Operating voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
DC Current per I/O Pins	20 mA
DC Current for 3.3V Pin	50 mA

Sparkfun Venus GPS with SMA Connector

Figure 2 is the Venus GPS module from Sparkfun Electronics [5]. This is the latest version of Venus GPS board which is very small, powerful and very versatile. This device allows

limited on-chip logging and one of the same version of this module comes with SPI flash memory chip. This version of GPS module is based on Venus 638FlPx [5][6]. This module has a default baud rate as 9600bps which can be adjustable to 115200bps.

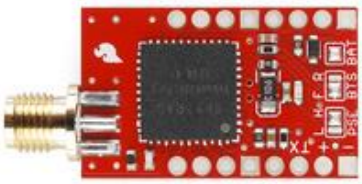


FIGURE 2. Sparkfun Venus GPS with SMA connector.

This board requires a regulated 3.3V of supply to operate. This board requires 90mA of current and at reduced power it requires 60mA of current. It also has an SMA connector to connect an external antenna, headers for 3.3V serial data, NAV (lock) indicator, pulse per second output port and supports external flash memory. It also comes with solder jumpers to easily configure the power consumption, boot memory and backup supply. It can also be connected with an external power supply in order to restart it very fast once the main supply is removed. There are pads on the bottom of the board for 0.2F supercap which helps the board hot-startable for up to 7 hrs without power. Table 2 below lists the technical specifications of the GPS module [7].

TABLE 2. Technical Specifications of Venus GPS Module

Receiver Type	L1 frequency GPS C/A code 65-channel architecture 8 million time-frequency searches per second
Accuracy	Position 2.5m CEP Velocity 0.1m/sec Timing 60ns
Update Rate	1 / 2 / 4 / 5 / 8 / 10 / 20 Hz (default 1Hz)
Baud Rate	4800 / 9600 / 38400 / 115200

TABLE 2. Continued

Main Supply Voltage	2.8V ~ 3.6V 2.8V ~ 3.6V, 1.08V ~ 1.32V
Protocol	NMEA-0183 V3.01, GGA, GLL, GSA, GSV, RMC, VTG (default GGA, GSA, GSV, RMC, VTG)
Operating Temperature	-40 ~ +85 deg-C
Storage Temperature	-40 ~ +125 deg-C

This GPS module has a connector to attach an external antenna for good reception of the signals. The antenna used to connect this GPS module is GPS Magnetic Mount SMA. It helps the GPS module to receive more signals even in high speed environment. This antenna operates at 3V and has 5-meter cable terminated by SMA connector as shown in figure 3. Table 3 lists the technical specifications of the GPS antenna [8].

FIGURE 3. Antenna GPS magnetic mount SMA.

TABLE 3. Technical Specifications of GPS Antenna

Gain 26dB
VSWR <2.0
Voltage 3.3V +/- 0.5V
Current 12mA
Weight 50g

HMC5883L electronic compass. In order to guide the robot in the desired direction, it is essential to first know which direction the robot is currently heading. To know its current heading compass is one of the options. For this project HMC5883L Module Compass is used as shown in figure 4 [9]. This compass is designed for low-field magnetic sensing with a digital interface. This compass is precise in-axis sensitivity and linearity. It is capable of measuring the direction and the magnitude of Earth's magnetic fields from milli-gauss to 8 gausses.

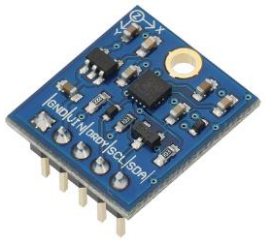


FIGURE 4. HMC5883L electronic compass.

The HMC5883L includes high resolution HMC118X series magneto-resistive sensors, automatic degaussing strap driver, offset calculation cancellation that enables the accuracy of heading in the range of 1^{o} to 2^{o} which is very good for this project [10]. It has 3-axis magnetoresistive sensors to calculate the current heading of the robot. There is no onboard regulator, so a regulated voltage of 2.16 - 3.3VDC is applied from Arduino board. The I^2C serial bus is essential for easy communication and very simple to interface with Arduino. The two logic connections SDA and SCL are used for the communication and to read the data from the compass module. The technical specifications of the compass are listed in table 4 [10].

TABLE 4. Technical Specifications of Electronic Compass [10]

Operating Voltage	2.16 to 3.6V
Supply Current	100 μA
Communication	I2C interface

TABLE 4. Continued

Maximum Output Rate	160 Hz
Resolution	5 milli-gauss

The output from the compass module is shown in figure 5. The output shows the position of the robot in 3 dimensional i.e. in X-axis, Y-axis and Z-axis. Comparing all the dimension the current heading of the robot is calculated which is also shown in the figure.

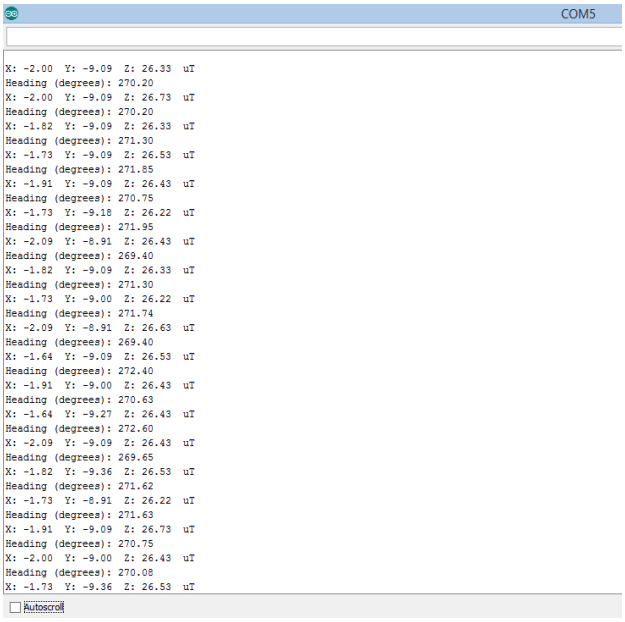


FIGURE 5. Output from the Compass module.

Monster Moto Shield: Motors are the primary part of the robot. In order to drive the robot, high power motors are used. These motors cannot be directly connected to the Arduino board since they require a large amount of voltage and also draw a high amount of current which can damage the Arduino board and other components connected with Arduino. To solve this issue for this project Monster Moto Shield is used shown in figure 6 [11]. This motor shield can be directly placed on the Arduino Mega using header pins. It is very convenient and simple for connecting the motors. With the help of motor shield the motors can be directly connected to the terminals provided and also external power supply is connected to drive the motors. Motor shield

can also be used to drive multiple motors in both the direction by varying the speed of individual motors.

FIGURE 6. Monster Moto Shield.

Monster Moto Shield is an updated version of Ardumoto motor driver shield. The biggest change made in this shield from the previous one is that the L298 H-bridge is replaced with a pair of VNH2SP30 which are full-bridge motor drivers [11]. This helps to protect the shield from high current and can drive high current motors without any problem. The motors and external power can be connected to the shield using 5mm screw terminal. It is always a good practice to use high gauge wire when high current motors are used. For high demand applications sometimes there is need for heat sink or fan but this board can hold up to 6A and barely becomes warm. This moto shield has overvoltage and under voltage protection which can shut down the motors if any problem occurs. Table 5 lists the technical specifications of the monster moto shield [11].

TABLE 5. Technical Specifications of Monster Moto Shield

Maximum Voltage	16V
Maximum Current	30A
Continuous Current Supply	14A
Maximum PWM Frequency	20 KHz
MOSFET on-resistance	19 mΩ (per leg)

TABLE 5. Continued

Under voltage and overvoltage shut down
Thermal shut down

Wild Thumper 6WD all-terrain chassis: For this project, a 6-wheel drive chassis designed by Dagu Electronics is used shown in figure 7 [12]. This robot is designed to traverse on rough terrains, a steep inclination which makes it capable of working on any kind of surface without any hassle. It has 6 powerful DC motors with brass brushes and has independent suspension for all the wheels with maximum traction which helps it to drive even on uneven areas [12]. This robot comes with 34:1 or 75:1 steel gearboxes that drive large diameter spike tires with good traction control.

FIGURE 7. Wild Thumper 6WD all-terrain chassis.

The chassis comes with 2mm thick anodized aluminum frame at the lower level and has two blocks to place batteries and other components. It also has an upper frame with 4mm holes which is also useful to mount various other components. The three motors on each side of the robot are wired in parallel and are connected to a connector on the robot. The two channel motor control is required to drive the chassis. The motors are high powered and on high power can reach to a top speed of 7 Km/hr. Table 6 lists the technical specifications for the 6-wheel robot chassis and DC motors [12].

TABLE 6. Technical Specifications of 6 Wheel Robot Chassis and DC Motors

Rated motor voltage	2 – 7.5 V
No-load Current at 7.2 V	420 mA per motor
No-load output shaft speed at7.2 V	350 RPM
Maximum payload	11lb (5 Kg)
Stall Current at 7.2 V	6.6 A per motor
Size	420 X 300 X 130 mm
Weight	6.0lb (2.7 Kg)

CHAPTER 3

INTERFACING 6 WHEEL CHASSIS WITH ARDUINO, COMPASS AND GPS MODULE

After studying all the components and their requirements it was the time to interface all the components together. Initially, it was necessary to assemble all the parts of 6-wheel robot. The motors are assembled with chassis, the connections of the motors are brought together from each side of the robot and then they are connected with the small switches given on the lower frame. Then a 7.5 V battery is installed on the lower frame case and it is connected with the motors to check if the robot is working properly. After this Arduino Mega is installed on the upper frame of the chassis and is powered by USB cable. For programming purpose, I used USB cable to power the Arduino. Further, Monster Moto shield is installed on the Arduino Mega. The motor output from both the sides of the robot are connected to the left and right screw terminals mounted on the shield. The shield is also connected to an external battery to power the motors as they require a high amount of current and voltage.

After the above part is done the GPS module along with the antenna is mounted on the robot. The GPS module is powered with 3.3V from the Arduino board and Tx Rx pin is connected to 46 and 48 pins on Arduino board respectively. Lastly, the compass module is placed on top of the robot. The compass is also powered from Arduino board and the communication pins SDA and SCL are connected to Arduino board to read the output from the compass. Figure 8 shows the assembly of the robot with all the components interfaced.

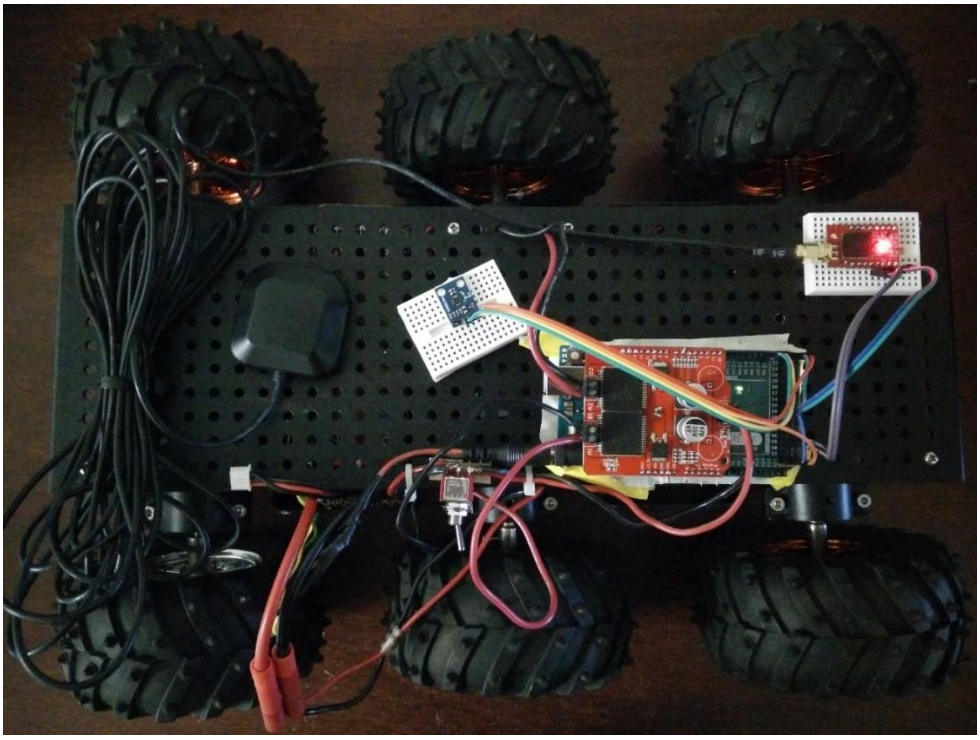


FIGURE 8. Assembly of final robot.

CHAPTER 4

SOFTWARE IMPLEMENTATION

This part is most important for the robot to function and demonstrate desired output. Software implementation include various parameters for building a sample code to test the working of each components, designing a flowchart to decide which component should operate at appropriate time, implement mathematical formulas and test whether desired results are obtained and at the end combining all the parts together and make an appropriate executable code for the desired working of the robot. As it is mentioned earlier in the introduction this project is divided into two parts. These two parts are explained below.

Interfacing Arduino Mega and GPS Module

In this part, the GPS module is first interfaced with Arduino Mega is shown in the figure 9 [13]. It is essential to know the connection pins used for the serial communication. TinyGPS library is used to test the module and check whether the coordinates are being transmitted to the Arduino board. The sample program is written in Arduino IDE software using TinyGPS library [14]. The pin connections to connect GPS module with Arduino Mega are given below.

1. The GND pin of GPS module is connected to any GND pin in Arduino board.
2. The GPS module needs 3.3V which is connected to 3.3V pin on Arduino.
3. The TX pin of GPD module is connected to pin 48 on Arduino board.
4. The Arduino board is connected to a computer using USB cable.

Figure 9 shows the interfacing of GPS module with Arduino Mega. The pins of the GPS module are connected to the respective pins of Arduino Mega. There is an antenna connected to GPS module which is used for the good reception of signals.

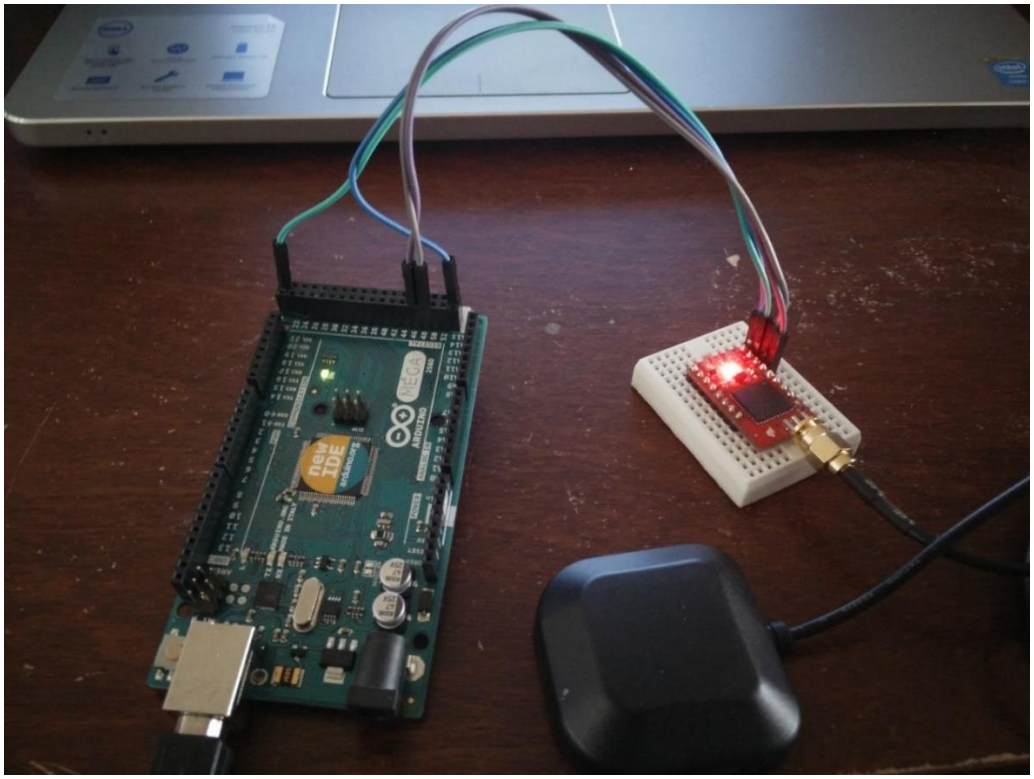


FIGURE 9. Interfacing of GPS module with Arduino board.

Once the Arduino board is connected to a computer it powers ON. The GPS module also turned ON which is indicated by an LED on the module. After the LED is turned ON it remains ON for a long time. This indicated that the GPS module is acquiring signals from the satellite. In the beginning, when the GPS module is turned ON for the first time it takes a lot of time to acquire signals from the satellite. After some time, the LED starts to blink which means it is receiving signals from the satellite. The output is shown in the form of latitudes and longitudes which can be seen on the serial output of the Arduino IDE software. Figure 10 shows the output of the GPS module. It shows the location of the robot in terms of longitudes and latitudes. The output from the GPS module is imported to an online GPS visualizer to track the coordinates [15]. Below in figure 11 is the path traced by the coordinates received. The path shown in orange color is the path traced by the GPS module and the path shown in red color is among those co-ordinates which are not traced by the module.

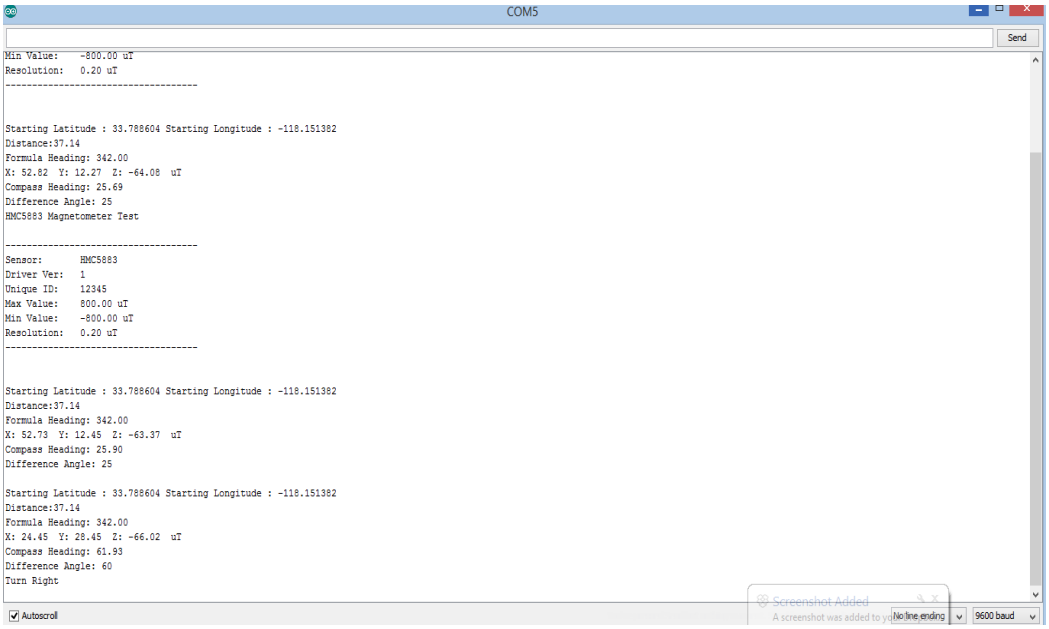


FIGURE 10. Output from the GPS module.

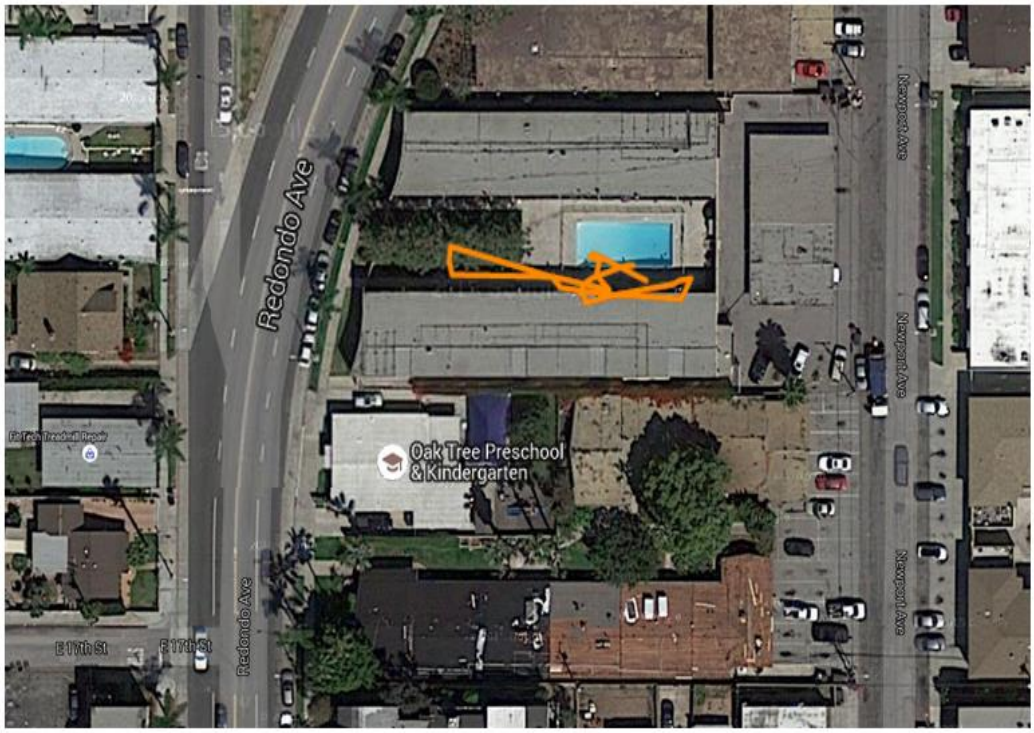


FIGURE 11. Output shown on GPS visualizer.

Implementation of Complete Logic to Drive the robot

After working on the GPS module and developing a sample code it was clear that the GPS module is acquiring signals from the satellite. The code is modified to display the latitudes and longitudes of the location. Now to drive the robot to the target it was necessary to calculate certain mathematical parameters. The target location is given in the algorithm as latitudes and longitudes. After the current location is fixed the algorithm will calculate the distance between current and target location. The algorithm will also calculate the heading needed to reach the target. The compass module will tell the current heading of the robot. A difference of the heading will be calculated by comparing the heading from the compass module and from the calculations. The resultant heading will be final heading on which robot should proceed.

After the heading is decided, the motor drives the robot to the destination location. Since the GPS module has to update the current information of the robot it was necessary to keep the speed of the robot slow. As the robot moves closer the GPS module will update the location and the distance will decrease. The formula which is calculating the distance is very accurate for shorter distances. Since the GPS module is not very good in locating the exact location there is a condition given which states that if the robot is in a radius of 5 meters then it should stop since it is very close to the destination location.

Distance formula. For a robot to how much to travel it is necessary to calculate the distance between the current position and the target. For this project, haversin distance formula is being implemented [16]. The significance of this formula is it calculates spherical distance on earth using trigonometric functions. On a spherical surface, the shortest path between two points is along an arc of a great circle. It is a circle drawn on earth with the same radius as that of the earth. Any

two points that lie on a unique great circle divide the arc into two arcs. The shortest path between the points is along the shortest arc among the two arcs. The haversin function is defined as below.

$$haversin(\theta) = sin^2\left(\theta/_2\right)$$

The haversin distance formula is given as follows.

$$haversin\left(\frac{d}{2R}\right) = haversine(\emptyset_2 - \emptyset_1) + \cos(\emptyset_1)\cos(\emptyset_2)\, haversin(\lambda_2 - \lambda_1)$$

Solving for d we get the distance formula

$$d = 2R\ \sin^{-1}\left(\sqrt{sin^2\left(\frac{\emptyset_2 - \emptyset_1}{2}\right) + \cos(\emptyset_1)\cos(\emptyset_2)\, sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Here, d is the distance between two co-ordinates,

R is the radius of earth i.e. 6371 km or 3961 miles

$\emptyset_1, \emptyset_2$ are latitudes of point 1 and latitude of point 2

λ_1, λ_2 are longitude of point 1 and longitude of point 2

Apart from the haversine formula there is also one formula to calculate the distance between any two points on the earth. The normal distance formula is given as follows.

$$d = R * \sqrt{2} * \sqrt{1 - (\cos(\emptyset_1)\cos(\emptyset_2)\cos(\lambda_2 - \lambda_1) - \sin(\emptyset_1)\sin(\emptyset_2))}$$

The haversine formula calculates the distance between any two points on earth, taking into consideration that the earth has a spherical surface. But the normal distance formula in 2 calculates the distance considering the earth as a flat surface without considering the spherical curvature of the earth. For small distances, the difference between these two formulas is very small as compared with the online results. But as the distance increases the difference between these two formulas increases and the normal distance formula gives inaccurate results. For a GPS

guided robot getting accurate results in the range of 5-6 meters is very essential. Considering the accuracy haversin formula is preferred over normal distance formula for all ranges of distances.

The comparison of the distance calculated using both haversin and normal distance formula is shown in table 7.

TABLE 7. Comparison of Haversine and Normal Distance Formula

Co-ordinates as Latitudes and Longitudes	Normal Distance Formula	Haversin Distance Formula	Actual Distance from Google
A – 39.28357, -120.51933 B – 39.61731, -120.53251	37.1 km	37.1 km	37.1 km
A – 42.69078, -119.08012 B – 44.33957, -120.28325	207.4 km	207.4 km	207.4 km
A – 39.93950, -115.60020 B – 43.05685, -119.69549	486 km	486.1 km	486.3 km
A – 42.59784, -91.21056 B – 46.90922, -118.81658	2208.4 km	2219.6 km	2220.3 km
A – 62.78254, -108.78868 B – 60.14424, 135.18732	5207.1 km	5318.2 km	5319.7 km
A – 55.413, -95.780 B – 57.031, 63.46856	7168.3 km	7372.9 km	7375.04 km

From the above table, it is clear that haversin formula gives more accurate results as compared to normal distance formula. So, haversin formula is being used for this project to calculate the distance between any two coordinates on the surface of the earth.

Now, the distance formula is decided the latitudes and longitudes are converted to radians by multiplying them with $\pi/180$. After converting latitudes and longitudes into radians they are substituted in the haversin distance formula to get the final distance. This formula is very accurate and very helpful for navigation purpose. Since the GPS is not accurate a desired value of 5m radius is given in the algorithm. The distance calculated is continuously being checked if it

is equal or less than 5 meters. If the condition is satisfied, then the robot is assumed to be close to the target and then it stopped there.

Heading angle: Heading angle is a term used to navigate the robot in the desired direction. It is the angle between two points and the robot as to proceed in that direction in order to reach the destination. It is a common term used for navigation in the field of aircraft or marine, vehicle navigation. Heading angle is defined as the north-south line on earth or the meridian and the line connecting the two points on earth. A compass is a component that gives heading information in order to reach the destination. For this project a forward azimuth formula is being implemented which trace a path along a great circle arc [17]. Below is the heading formula which will calculate the desired heading of the robot to reach the target point.

$$h = atan2(\sin(\lambda_2 - \lambda_1)\cos(\emptyset_2), \cos(\emptyset_1)\sin(\emptyset_2) - \sin(\emptyset_1)\cos(\emptyset_2)\cos(\lambda_2 - \lambda_1))$$

Here, h is the heading,

$\emptyset_1, \emptyset_2$ are latitudes of point 1 and latitude of point 2,

λ_1, λ_2 are longitude of point 1 and longitude of point 2.

The above formula calculates the heading between the two points. The heading is calculated in radians, so it is converted into degrees by multiplying it with π/180. True heading is based on true north. Angles are measured clockwise from north. The range of angles is from 0 to 360^0. After the heading is known it is compared with the current heading of the robot given by compass. The difference between the two angles will be the final heading in which the robot should proceed to reach the destination. The process is repeated every time the GPS receives the location of the robot. After the angle is calculated it is necessary to direct the robot to north, south, east or west. This action is taken based on the Azimuth angle. Below in figure 12 is the

Azimuth angle which is implemented in the algorithm to decide which direction the robot should move to reach the destination [18].

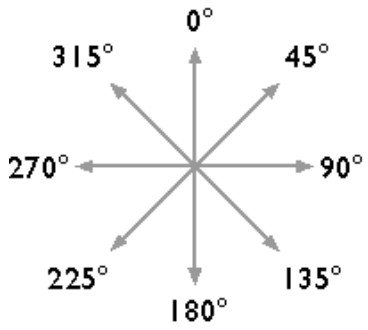


FIGURE 12. Azimuth angle.

From Azimuth angle, few conditions can be given in order to decide which direction the robot needs to turn. If the angle is between 225^0 to 315^0 then the robot has to move left towards the west. If the angle is between 45^0 to 135^0 then robot has to move right towards the east. If the angle is between 315^0 to 45^0 the robot will move forward towards the north. The reason for giving such a range of angles is because the compass is not very accurate. Such a big range will able to minimize some errors and the robot will able to move in the desired direction.

Motor control: For this project DC motors are used which are high speed motors. The motors draw a lot of current if driven at high speed. Also, is not possible to drive the robot at full speed because the GPS signals are always updated and some delays are needed to update the GPS location. Also, if the motors drive at high speed the robot will not follow the desired path and will never reach the destination location. In order to avoid this, the robot is driven to a minimum speed. The speed of the robot is controlled using PWM signals from the Arduino board. The range of PWM range is 0-255. The robot cannot be driven at very low speed because there will be a restriction of current flow to the motors. The speed of the robot is set to 55 below which the robot was unable to move.

CHAPTER 5

FLOW CHART

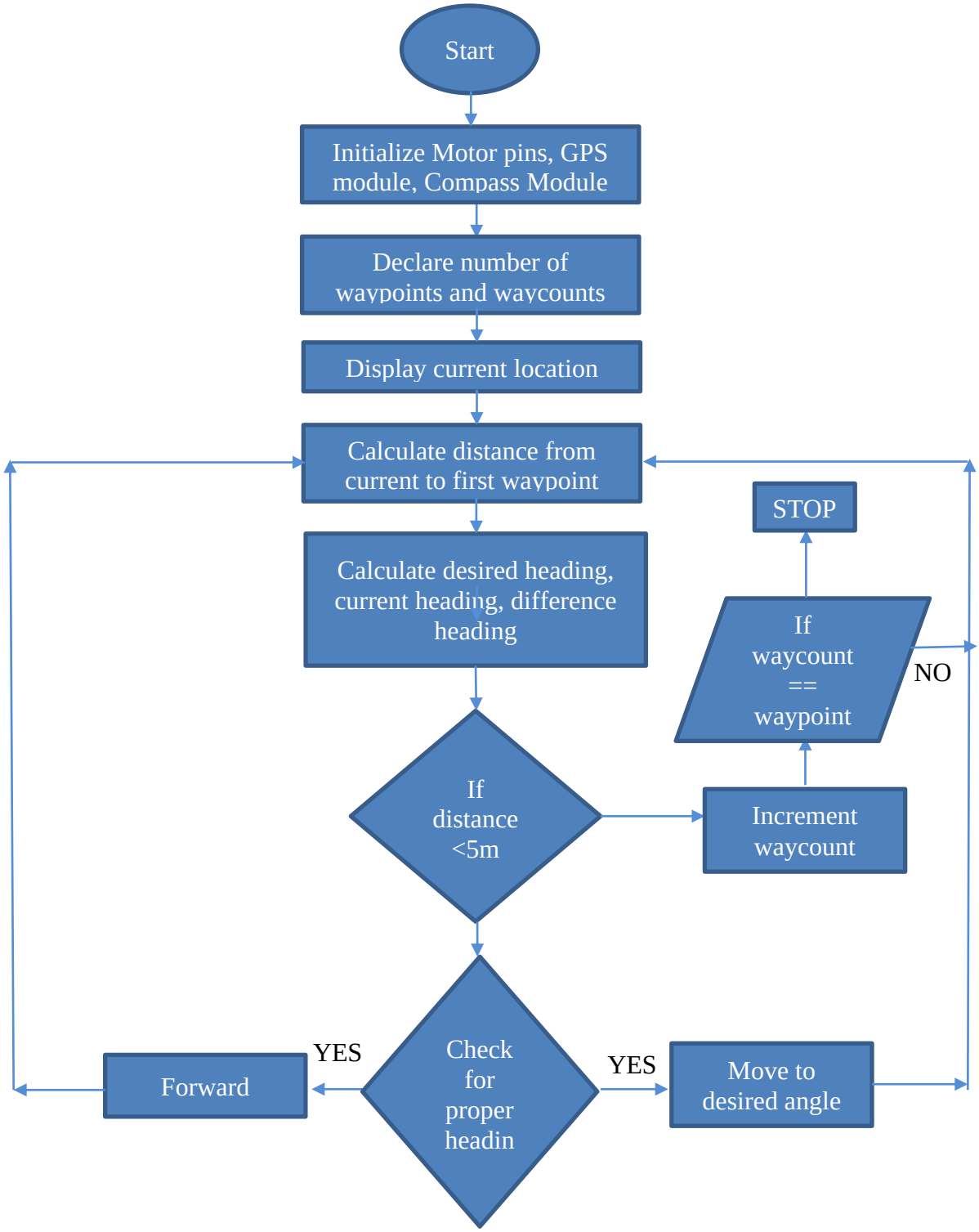


FIGURE 13. Working of robot.

The above flow chart explains the operation of the algorithm designed for the robot. Starting from the top is the initialization section for motor pins, GPS module pins and the declaration of some variables that are used in the later part of the algorithm. Initially, when the Arduino starts the GPS module, compass and monster moto shield is powered ON. The GPS module locates the current position of the robot. The current location is stored in two variables as latitudes and longitudes. The destination coordinates are defined in the algorithm. Now the algorithm calculates the distance between the current location and destination using the distance formula. Further, the current heading of the robot is located using the compass module and also the heading from a current location to the destination is calculated using heading formula. In order to drive the robot in the desired direction, the difference of the current heading and the formula heading is calculated. Once these calculations are done the robot starts to move towards the first location given in the algorithm. While traveling towards the destination location the GPS module continuously updates the location and also the overall process is done till the robot reaches close to the destination.

In the algorithm, the minimum distance condition of 5 meters is defined which is checked every time a new location is found. If this condition is satisfied it means that the robot has reached its first destination location. Now the counter is incremented and again the same process is repeated. Once the counter reaches the predefined value it means that the robot has reached to its final destination and it slowly stops. The advantage of this algorithm is that it calculates the accurate distance between ant two coordinates. As per the working of the robot, it is observed that the accuracy of the robot is 5 to 6 meters which is acceptable for this project.

CHAPTER 6

CONCLUSION

The primary goal of this project was to build an autonomous robot that can move from one location to another with the help of GPS coordinates. The important task of this project was to calculate the accurate distance between any two points on the surface of the earth. Further implementing an algorithm that can store multiple coordinates which will guide the robot to reach the final location. These major tasks were not accomplished in the previous development of this project, hence, it became very important to design a new algorithm to make the robot more robust and innovative from the previous one. Before implementing a new algorithm, it was very important to understand the working of all the components. At first, the GPS module was studied and interfaced with Arduino Mega. A sample code was written to check whether the GPS module is acquiring signals from the satellite. From the reading of the GPS module it was observed that the accuracy of the GPS module is about 5 to 6 meters which is acceptable for this project. Next, the GPS module is interfaced with compass and the 6-wheel robot chassis and DC motors. The compass was used to know the current heading of the robot. The DC motors are high power motors so it was necessary to keep the speed of the motors very low for the proper working of the robot. After studying all the components individually, they are assembled together to make a complete robot.

To calculate the distance haversin distance formula is used, which calculate accurate distance with the help of great circle property around the spherical shape of the earth. After the distance is calculated heading between the current and the destination coordinates is calculated. A forward Azimuth heading formula is used to calculate the heading. The heading calculated is compared with the current heading given by the compass module. The difference of the two is

the final heading, which the robot has to follow. At each location, the coordinates are updated and all the parameters are calculated again. Due to continuously updating the location, the robot moves in the desired direction and stops after it reaches close to the target. The algorithm is designed in such a way that multiple coordinates can be given with the help of which robot reaches to the final destination.

The output shows that the robot is able to gather all the data needed to reach destination location. The final output of this project is shown in a video. A link of the video is attached here: GPS Guided Autonomous Robot

APPENDIX

ALGORITHM

```
#include <AltSoftSerial.h>
#include <TinyGPS.h>
#include <Wire.h>
#include <HMC5883_U.h>
float i, headingValue,diff;
float flat, flon;
float heading = 0;
int headinggps = 0;

AltSoftSerial gpsSerial(48,46);
TinyGPS gps;

int radius, x4=0;
#define waypoints 2
int waycont=1;
float x2lat;
float x2lon;

float flat2 = 33.788640;
float flon2 = -118.151381;

float flat3 = 33.788620;
float flon3 = -118.151340;

/*  VNH2SP30 pin definitions
 xxx[0] controls '1' outputs
 xxx[1] controls '2' outputs */
int inApin[2] = {7, 4};  // INA: Clockwise input
int inBpin[2] = {8, 9}; // INB: Counter-clockwise input
int pwmpin[2] = {5, 6}; // PWM input
int cspin[2] = {2, 3}; // CS: Current sense ANALOG input
int enpin[2] = {0, 1}; // EN: Status of switches output (Analog pin)

/* Assign a unique ID to this sensor at the same time */
HMC5883_Unified mag = HMC5883_Unified(12345);

//=====================================
void Forward()
{
  motorGo(0, CW, 55);
  motorGo(1, CW, 55);
  }

void Backward()
{
  motorGo(0, CCW, 55);
```

```
  motorGo(1, CCW, 55);
  }

void Right()
{
  motorGo(0, CCW, 255);
  motorGo(1, CW, 255);
}

void Left()
{
  motorGo(0, CW, 255);
  motorGo(1, CCW, 255);
}

void Stop()
{
  motorGo(0, CW, 0);
  motorGo(1, CW, 0);
  }
//======================================
void setup(void)
{
  Serial.begin(9600);
  gpsSerial.begin(9600);
  Serial.println("HMC5883 Magnetometer Test");
  Serial.println("");

    // Initialize digital pins as outputs
  for (int i=0; i<2; i++)
  {
    pinMode(inApin[i], OUTPUT);
    pinMode(inBpin[i], OUTPUT);
    pinMode(pwmpin[i], OUTPUT);
  }
  // Initialize braked
  for (int i=0; i<2; i++)
  {
    digitalWrite(inApin[i], LOW);
    digitalWrite(inBpin[i], LOW);
  }

  /* Initialise the sensor */
  if(!mag.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections */
```

```
    Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
    while(1);
  }

  }
//================================================
bool newdata()
{
  while (gpsSerial.available())
  {
    if (gps.encode(gpsSerial.read()))
      return true;
  }
  return false;
}
//================================================

void loop(void)
{

   if(waycont==1)
  {
    x2lat = flat2; // First waypoint
    x2lon = flon2;
  }
    if(waycont==2)
    {
      x2lat = flat3; // second
      x2lon = flon3;
    }

//=============Distance================================

float diflat=0;
float diflon=0;
float dist_calc=0;
float dist_calc2=0;
float Int_Dist=0;
float Final_Dist=0;

diflat=radians(x2lat-flat);
flat=radians(flat);

diflon=radians(x2lon-flon);
x2lat=radians(x2lat);
```

```
dist_calc = (sin(diflat/2)*sin(diflat/2));
dist_calc2 = (cos(flat)*cos(x2lat)*sin(diflon/2)*sin(diflon/2));
Int_Dist = dist_calc + dist_calc2;

Final_Dist = (2*atan2(sqrt(Int_Dist),sqrt(1-Int_Dist)));
Final_Dist = Final_Dist*6371000; // distance is in meters
distance.meter = Final_Dist;

Serial.print("Distance:");
Serial.println(distance.meter);
if(distance.meter<5)
{
  if(waycont==waypoints)
  {
 Stop();
  }
  waycont+=1;
}

//==================Heading Formula Calculation================

flon = radians(flon);
x2lat = radians(x2lat);

heading = atan2(sin(x2lon-flon)*cos(x2lat), cos(flat)*(sin(x2lat)-sin(flat))*cos(x2lat)*cos(x2lon-
       flon)); //
heading = heading*180/3.1415926536; //convert radians to degrees
int head = heading;

if (head<0)
{
  heading = ((head+360)); // If heading is negative make it positive
}
 Serial.print("Formula Heading: ");
 Serial.println(heading);

//================================Compass====================
  sensors_event_t event;
  mag.getEvent(&event);

  Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print("  ");Serial.println("uT");
```

```
  // Correct for when signs are reversed.
  if(heading < 0)
    heading += 2*PI;

  // Check for wrap due to addition of declination.
  if(heading > 2*PI)
    heading -= 2*PI;

  // Convert radians to degrees for readability.
  float headingDegrees = heading * 180/M_PI;

  Serial.print("Compass Heading: ");
  Serial.println(headingDegrees);

//============Desired Heading================

x4 = headingDegrees - heading; //difference
if(x4<2)
{
  x4 = x4+360; //if difference is negative
}
else
{
  x4 = x4;
}

Serial.print("Difference Angle: ");
Serial.println(x4);

delay(500);

if(x4!=0)
{
   if(x4>225&&x4<315) //Left Turn
  {
    Left();
    Serial.println("Turn Left");
    delay(300);

    Stop();
    delay(500);
    }

   else if(x4>=45&&x4<135) //Right Turn
  {
    Right();
```

```
    Serial.println("Turn Right");
    delay(5000);

    Stop();
    delay(500);
      }

  else
  if(headingDegrees>315&&headingDegrees<45)// Forward
  {
    Forward();
    //Serial.print("Turn Left");
    delay(1000);

    Stop();
    delay(600);

   }
}

else
{
  Forward();
  delay(300);
}
if(distance.meter<1)
{
  Stop();
  Serial.println("Destination has reached");
}
}
}
void motorOff(int motor)
{
  // Initialize braked
  for (int i=0; i<2; i++)
  {
    digitalWrite(inApin[i], LOW);
    digitalWrite(inBpin[i], LOW);
  }
  analogWrite(pwmpin[motor], 0);
}

void motorGo(uint8_t motor, uint8_t direct, uint8_t pwm)
{
  if (motor <= 1)
```

```
{
  if (direct <=4)
  {
    // Set inA[motor]
    if (direct =6)
      digitalWrite(inApin[motor], HIGH);
    else
      digitalWrite(inApin[motor], LOW);

    // Set inB[motor]
    if ((direct>0)||(direct<2))
      digitalWrite(inBpin[motor], HIGH);
    else
      digitalWrite(inBpin[motor], LOW);

    analogWrite(pwmpin[motor], pwm);
  }
}
```

REFERENCES

REFERENCES

[1] Georgia Tech Research Institute. (n.d.). On Their Own: Research on Autonomous Technology is Developing Increasingly Sophisticated Capabilities in Air, Marine and Ground Robotic Vehicles. [Online]. Available: http://gtri.gatech.edu/casestudy/ autonomous-technology-research-developing-increasi

[2] Amazon. (n.d.). Arduino A0000067 DEV BRD, ATMEGA2560 R3. [Online]. Available: http://www.amazon.com/Arduino-Mega-2560-MEGA R3/dp/B006H 0DWZW/ref= sr_1_1?ie=UTF8&qid=1447734901&sr=8-1& keywords = arduino+mega+2560+r3

[3] Arduino. (n.d.). Arduino Mega 2560 and Genuino Mega 2560. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardMega2560

[4] Arduino. (n.d.). What is Arduino. [Online]. Available: https://www.arduino.cc/en/ Guide/Introduction

[5] Sparkfun. (n.d.). Sparkfun Venus GPS with SMA Connector. [Online]. Available: https://www.sparkfun.com/products/11058

[6] SK Pang Electronics. (n.d.). Venus GPS with SMA Connector. [Online]. Available: http://skpang.co.uk/catalog/venus-gps-with-sma-connector-p-1093.html

[7] SkyTraq Technologies, Inc. (2016). Venus638FLPx GPS Receiver Data Sheet. [Online] Available: http://cdn.sparkfun.com/datasheets/Sensors/ GPS/Venus/ 638/doc /Venus638FLPx_DS_v07.pdf

[8] Sparkfun. (n.d.). Antenna GPS 3V Magnetic Mount SMA. [Online]. Available: https://www.sparkfun. com/products/464

[9] Parallax. (n.d.). Gyroscope Module 3-Axis L3G4200D. [Online]. Available: https://www. parallax.com/product/27911

[10] Honeywell. (2013). 3-Axis Digital Compass IC HMC5883L. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_ IC.pdf

[11] Sparkfun. (n.d.). Sparkfun Monster Moto Shield. [Online]. Available: https://www. sparkfun.com/products/10182

[12] Pololu. (n.d.). Dagu Wild Thumper 6WD All-Terrian Chassis, Black, 34:1. [Online]. Available: https://www.pololu.com/product/1562

[13] Simon Monk. (n.d.). Sparkfun Venus GPS and Arduino. [Online]. Available: http:// www.doctormonk.com/2012/05/sparkfun-venus-gps-and-arduino.html

[14] Mikal Hart. (2013). TinyGPS. [Online]. Available: http://arduiniana.org/libraries/tinygps/

[15] GPS Visualizer (2016). GPS Visualizer. [Online]. Available: http://www.gpsvisualizer.com/draw/

[16] Plus Magazine. (2014). Lost but lovely: The Haversin. [Online] Available: https://plus.maths.org/content/lost-lovely-haversine

[17] Open Site Mobile. (2011). Calculate distance, bearing and more between Latitude/Longitude points. [Online]. Available: http://opensitemobile.com/latlon/

[18] PENNSTATE University. (2014). The Nature of Geographical Information. [Online]. Available: https://www.e-education.psu.edu/natureofgeoinfo/c5_p7.html