An Arduino UNO application: GPS guided unmanned ground vehicle

Hasan Fatih Aci[1], Muhammed Senyurt[1], Tugay Bozik[1] and Guray Gurkan[1]

[1]Department of Electrical & Electronic Engineering, Faculty of Engineering,
Istanbul Kültür University, Atakoy Campus, 34156, Bakirkoy, Istanbul, Turkiye
hasanfatihaci@gmail.com, muhammedsenyurt@gmail.com,
tugaybozik@hotmail.com, g.gurkan@iku.edu.tr

Abstract

This paper presents an application of Arduino UNO to GPS guided unmanned vehicle. The four-wheeled small scale (55 cm x 25 cm) vehicle is capable of moving forward, rotating and consists of a GPS sensor, a magnetometer, a motor driver and four DC motors. The target coordinates (key points) are manually entered as a part of the control algorithm script (.ino file). The developed control algorithm makes a blind calculation of the distance and required rotation between consecutive key points. The initial coordinates of the vehicle, however, requires the only field-dependent calculation of the distance and target heading between the initial and the first key point. The whole target route is achieved by consecutive "rotation" and "forward movements (straight path)" between key points. The GPS sensor is used to determine the initial coordinates whereas magnetometer is used to determine the heading (and rotation) before (and during) each key point movement. When taxiing through blind-calculated route, the acquired GPS data is transmitted via Bluetooth for offline monitoring.

1. Introduction

By development of fast prototyping microprocessor boards and rich libraries of coding platforms, it has become easier to program and apply a theoretical idea to an embedded system. Being one of the fastest growing prototyping platforms, Arduino UNO is the easiest and most suitable platform even for an individual with no technical background to make technology more accessible [1]. By combining engineering knowledge and various sensor types, one can generate a complex system with high processing capabilities.

In this study, data from a GPS sensor and magnetometer that are mounted on top of a four wheel vehicle (named GUVE*) are interpreted and used for route decision by Arduino UNO (with Atmega328P [2]). Similar studies exist in the literature [3-6] about unmanned vehicle guidance, however, this study presents the application of Arduino UNO and thus Atmega328P microprocessor to this subject. The route decision algorithm that is developed under Arduino IDE and its libraries is the main frame of this study. In addition, magnetometer calibration method is also presented through this paper.

** Güve is the Turkish translation of "moth".*

2. System Design

2.1. Chassis Design

Chassis design was kept as simple and thus cheap as possible for low cost construction (Fig.1, Fig.2). 3D design of the chassis

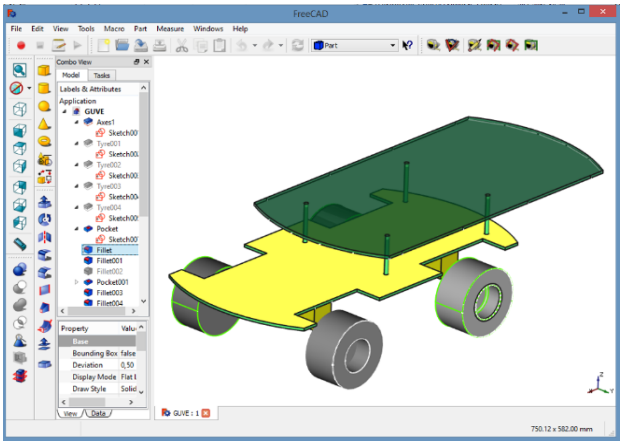

Fig. 1. 3D design of vehicle via FreeCAD

was developed via FreeCAD [7]. Poly-methyl-methacrylate (PMMA, known as *Plexiglas*®) layers were used as the main chassis material. Layer's screw holes were cut manually via Dremel® 300 multitool. Four 90 rpm DC motors were mounted to drive 120 mm radius plastic wheels [8]. The vehicle was given a tail via the top layer to shift the center of gravity through

Fig. 2. Main chassis realization of GUVE. Control panel (far top) and wheels are visible.

rear wheels. That enables the rotation of the vehicle on its axis via wheels moving in contrary directions on each side (left and right).

2.2. Control elements

Arduino UNO and motor driver boards were mounted on base layer whereas GPS sensor, magnetometer and battery were mounted under the top (transparent) layer. The control panel with two on/off buttons and 2x16 LCD was located on the top for a clear view of LCD and easy access to buttons.

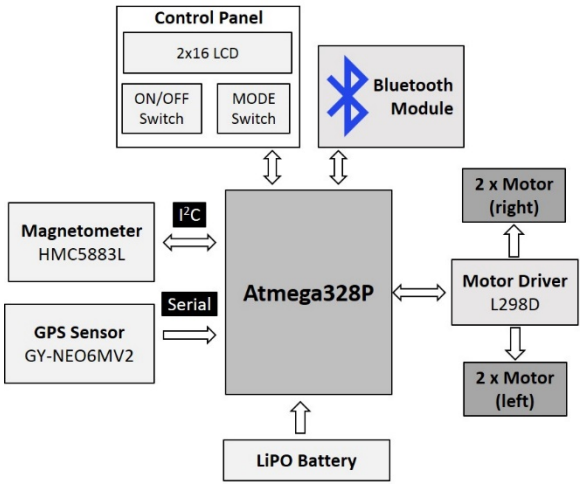

Fig. 3. Vehicle connection diagram.

Arduino UNO has Atmega328P 8-bit microprocessor [2] with 6 ADC input channels, two of which can also host I^2C connection with slave devices. In addition, it has 13 programmable digital pins, 6 of which can be used as PWM source pins.

Programmed microprocessor can also be operated in standalone mode on a breadboard/PCB with very little additional components (16 MHz crystal, two capacitors and a reset button). The board's software (like other boards in Arduino family) is programmed by Arduino IDE and uploaded via serial connection with built-in USB-serial converter. If powered from USB, the same serial connection becomes the hardware serial communication channel between microprocessor and PC. The connection ports RX and TX pins are located on board and data can be easily viewed via "Serial Monitor" of Arduino IDE.

Control panel was mounted on top layer. To display the status and outputs of the control the algorithm 2x16 LCD was used in the beginning of the project. Via built-in library in Arduino IDE, it takes really few lines of code to display messages and numbers on LCD. In later stages of the study, a Bluetooth module (HC-05) is added to enable wireless transmission of the status and outputs to a mobile device. In addition to LCD, two switches are located on control panel. The ON/OFF switch simply controls battery connection whereas MODE switch controls the connection of RX pin with outer devices. While programming, the RX pin of Arduino Board should be disconnected from loads ("Program" mode) in order to prevent the distortion of incoming program data bits from TX pin of PC port. In normal operation, the MODE switch is brought back in "Operation" mode.

The chosen GPS module GY-NEO6MV2 [9] has a (standard) maximum capacity of 55 channels in L1 band. The appropriate power supply range is from 2.7V to 3.6V. UART interface is used for serial communication with a default baud rate of 9600 bps at 1 Hz refresh rate. The external active antenna size is 25 mm x 25 mm. For reading of GPS data via serial port, TinyGPS [10] Arduino library is used. By predefined structs and arrays, the relevant library makes GPS acquisition much easier with less coding.

Vehicle rotation is tracked via 3-axis magnetometer (digital compass) HMC5883L breakout board [11] that is able to measure magnetic field in Gauss units. The sensor is connected to microprocessor's I^2C (Inter-integrated Circuit) interface and thus requires two wires, namely Serial Data (SDA) and Serial Clock (SCL). The sensor is powered via 3.3V DC voltage that is also available on Arduino board. Device parameters such as measurement range (±1.3, ±2, ±4, ±8 Gauss) and measurement mode are adjusted by coding via installed Arduino device library [12].

Four 90 rpm DC motors were driven via Arduino Motor Shield [13] that involves a L298 chip. The chip is able to control two independent motors via two H-bridge circuits. In addition to motor connection pins, direction, on-off and enable pins are also available and used in movement control of the vehicle. Pulse Width Modulation (PWM) pins of Arduino are used to control the speed of the motors.

All the power requirement of the vehicle is supplied via 11.1V 3700mAh Lithium-polymer (Li-Po) battery. The battery weight is used to encourage rotation movement by installation of battery on tail of the vehicle chassis.

3. Sensor data interpretation

3.1. GPS

General purpose low cost GPS receivers produce 1 coordinate pair in a second (1 Hz). Depending on the selected NMEA (National Marine Electronics Association) sentence, additional information such as satellite clock, velocity, altitude, signal quality can also be acquired. In this study, NMEA-0183 sentence (data sentence starting with $GPGGA) was used since only the position data is needed.

The positioning accuracy of these low cost receivers are above 1 meter and thus introduces an unneglectable error for small field studies. The sensor data gives the location in degrees which can also be converted to degrees, minutes and seconds. In this study, we matched the degree difference in longitude and latitude to the distance between two coordinates. Referring to our measurements, 1 second (1/3600 degrees) change in north-south direction is approximately 32 meters whereas 1 second change in east-west direction is 23 meters. These values are valid for Istanbul Kültür University, Atakoy Campus (40.99109050º north, 28.83189502º east) region. Thus, by knowing the target (key) points, we were able to calculate the approximate distance (in meters) between these points simply using Pythagorean Theorem.

3.2. Magnetometer

Magnetometer data gives instant magnetic field measured along its (x-y-z) axes. The data cannot be directly used for heading (rotation angle from North) calculation. If the sensor is rotated about one of its axis that is hold perpendicular to Earth's surface (Fig.4), the vector magnetic field measured along the other two axes should form a circular shape with a specific offset on x-y coordinates, since Earth's magnetic field has one direction (South to North) and has a constant norm in a specific

point. By detecting minimum and maximum values through two moving axes and taking the average of these, we have measured the offsets (x_0 and y_0) in a practical way. Afterwards, by zero-offset values, the heading can be calculated from raw data (x_{raw} and y_{raw}) by

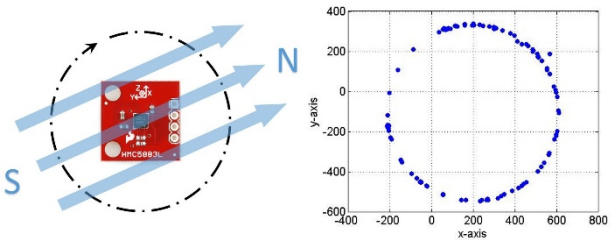

Fig. 4. 3-axis magnetometer calibration . The sensor is rotated about its z axis where x-y axes are hold parallel to earth's surface.

$$\text{heading}_{North} = \tan^{-1}\left(\frac{y_{raw} - y_0}{x_{raw} - x_0}\right) \quad (1)$$

4. Control Algorithm

4.1. Selection of key points

Control algorithm is developed under Arduino IDE (C language). For operation, target (key) points should be manually entered by the programmer in two float precision array format, one for latitude and the other for longitude components. Thus, the size of each array is the number of key points. Before each test, the key points were selected in desired order by Google Earth (Fig.5), saved as "kmz" file and converted to text file by a free online tool [14]. The text formatted coordinates were then copied easily as corresponding array elements.

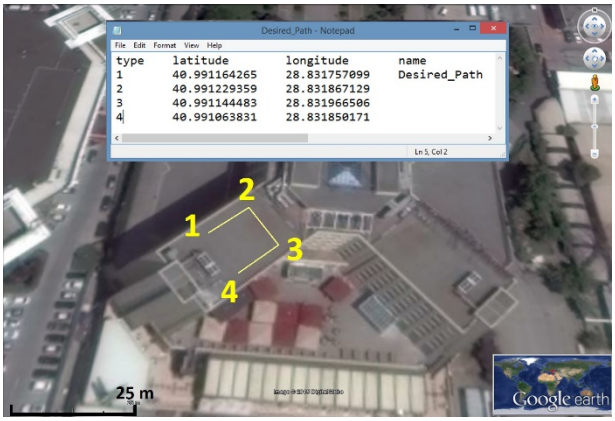

Fig. 5. Ordered key points used in terrace test

4.2. Magnetometer calibration

As mentioned earlier, the magnetometer data cannot be directly used for heading calculation with a single measurement. The control algorithm adjusts motor directions and inputs such that the vehicle rotates around its axis for 10 seconds during which magnetometer data is also acquired. During acquisition, minimum and maximum values of data from each axis (x and y for this study) are updated continuously. After end of rotation, we calculated the offsets in each axes by finding the mean of the final minimum and maximum (x and y axes) data values. The calculated offset values were stored and used for the rest of the run for heading calculations.

4.3. Finding initial location

Since the vehicle is randomly placed in the test field, the first duty is to find this unknown point. It is simply achieved by 60 seconds of static acquisition of GPS data. After 60 measurements, the average of the latitude and longitude components are assigned as initial point.

4.4. Movement between key points: Heading and distance calculations

After determination of initial vehicle coordinate, the next step is to calculate the target heading and target distance to the next (and thus the first) key point. For calculation of these parameters, we have developed two main functions that require two coordinates (current and target) and thus four parameters as input.

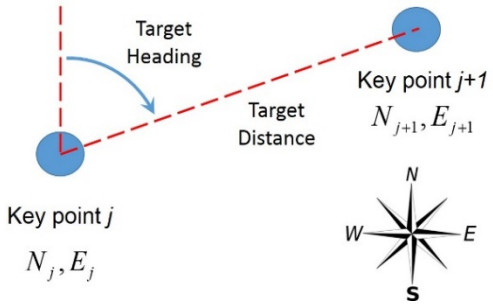

Fig. 6. Target heading and distance illustration between two adjacent key points. N denotes north (latitude) component and E denotes east (longitude) component of coordinates.

For target heading calculation, the ratio of the differences between target (N_{j+1}, E_{j+1}) and current (N_j, E_j) points are used. The target is defined as the angle between north and target point with origin considered as current point (Fig.6). That is, with ΔN and ΔE denoting the differences between north and east components, the target heading is calculated by

$$\text{heading}_{\text{Target}} = \frac{\Delta E}{\Delta N} \quad (2)$$

Depending on relative target position, target angle can take values from 0 to 359 degrees. Thus, the proper angle value is further achieved by considering the relative position in four quadrants principle. After calculation of target angle, the vehicle starts a clockwise rotation on its axis with half of its maximum speed. At the same time, the heading angle is updated by continuous acquisition of magnetometer data. As soon as the difference between the (calculated) target heading and (updated) current heading gets below 1 degree, the rotation stops.

After heading adjustment, the next step of the control algorithm is to determine the forward movement duration, which requires the distance and velocity parameters. For this, we first calculated the speed of the vehicle in forward movement mode, which was found to be 40 cm/s (1.45 km/s). The distance is the target distance and calculated by a second function that has the

same inputs as target heading calculation function. As mentioned earlier, the 1 second changes in north-south and east-west direction were found to be 32m and 23 m, respectively. Thus by pre-calculated parameters ΔN and ΔE, the distance (in meters) is given by

$$\text{distance}_{\text{Target}} = \sqrt{(32 \cdot \Delta N)^2 + (23 \cdot \Delta E)^2} \quad (3)$$

Following the calculation of the target distance, the duration of forward movement is simply found as

$$\Delta t_{\text{Forward}} = \frac{\text{distance}_{\text{Target}}}{0.4} \quad (4)$$

As the final key point is reached, the vehicle stops and "Finished" word is displayed on 2x16 LCD screen and transmitted via Bluetooth module to mobile receiver.

4. Field Test

The developed control algorithm is tested on terrace floor of Istanbul Kültür University Atakoy Campus building with key points chosen in a rectangular shape (Fig.5). The vehicle was placed randomly on terrace. During movement through key points, acquired GPS coordinates are transmitted to mobile receiver through Bluetooth. These coordinates are shown in

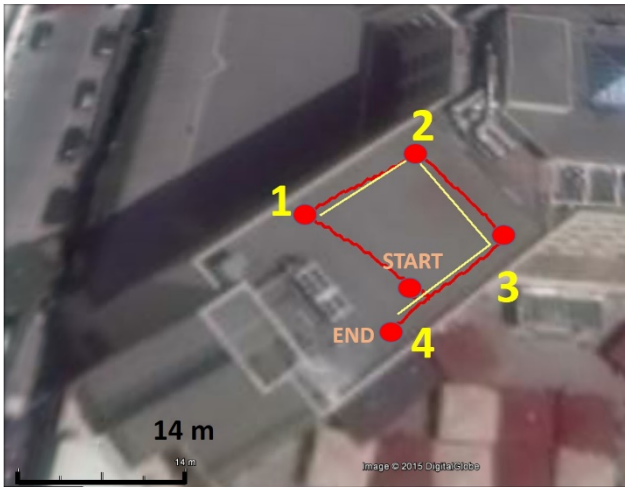

Fig. 7. The overlapped form of ordered key points (red dots) with acquired GPS coordinates (red line) through terrace test. The initial and final coordinates are also shown.

Fig.7.

5. Conclusion

Unmanned vehicle control algorithm was developed via Arduino IDE and embedded to Arduino UNO (Atmega328P). The GPS sensor and magnetometer are the key elements for detecting the location and orientation of the vehicle. The target heading and distance calculations within desired key points are independent of initial coordinate (Fig.5, Fig.7). However, the accuracy of the detected initial coordinate directly affects the consistency of these calculations since inaccurate detection yields wrong target heading and target distance parameters to the first key point. The vehicle's velocity determines the forward movement durations between key points and thus the algorithm should be updated (Eq. 4) for a different velocity. The performance of the control algorithm was satisfactory for future autonomous driving applications. Further studies may include obstacle detection and path update algorithms to be developed via Arduino IDE.

6. Acknowledgement

This project is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) with project number 1919B011400287.

7. References

[1] D. Kushner. (2011). *The Making of Arduino*. Available: http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino

[2] ATMEL. (2010). *Atmega328P Datasheet*. Available: http://www.atmel.com/images/doc8161.pdf

[3] C. Wuthishuwong, C. Silawatchananai, and M. Parnichkun, "Navigation of an intelligent vehicle by using stand-alone GPS, compass and laser range finder," in *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, 2009, pp. 2121-2126.

[4] R.-S. Run, Y. Jui-Cheng, and T. Cheng-Yu, "A low cost implementation of GPS guided driverless cars," in *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, 2010, pp. 1610-1614.

[5] C. Shun Hung, J. Jyh Ching, and L. Yu Chun, "Applying fuzzy gain scheduling technique for GPS-based unmanned vehicle navigation and control," in *SICE Annual Conference 2010, Proceedings of*, 2010, pp. 336-341.

[6] S. J. O. Corpe, T. Liqiong, and P. Abplanalp, "GPS-guided modular design mobile robot platform for agricultural applications," in *Sensing Technology (ICST), 2013 Seventh International Conference on*, 2013, pp. 806-810.

[7] *FreeCAD, An Open Source parametric 3D CAD modeler*. Available: http://www.freecadweb.org/

[8] *Dagu Wild Tumper Wheels*. Available: https://www.pololu.com/product/1558

[9] *u-blox NEO 6 GPS Series datasheet*. Available: http://www.rlocman.ru/i/File/2011/04/22/1.pdf

[10] *TinyGPS Arduino Library*. Available: http://arduiniana.org/libraries/tinygps/

[11] *HMC5883L Breakout Board*. Available: https://www.sparkfun.com/products/10530

[12] *HMC5883L Arduino Library*. Available: https://github.com/jarzebski/Arduino-HMC5883L

[13] *Arduino Motor Shield datasheet*. Available: http://www.farnell.com/datasheets/1625170.pdf

[14] *GPS Visualizer webpage*. Available: http://www.gpsvisualizer.com/