

ESE 400 Independent Study Final Report:  
Developing Autonomous Driving Algorithm  
Based on IMU Navigation

Sam Chai

Maira Feng

2018/5/4

## **1. Intro & Background**

With the increasing popularity in autonomous driving technology, self-driving cars have gained increasing attention in both the industry and the general public. For example, companies such as Google and Uber have been investing large amount of capitals in developing safe self-driving algorithms. Autonomous driving cars with robust performance are predicted to increase the traffic flow, reduce injury rate from accidents and reduce the need for parking space. Thus, the development of autonomous driving algorithms has a long-lasting importance.

Among the many considerations of successful self-driving algorithms, locating the current position of the vehicle for navigation is one of the most fundamental functionality required. In this context, the idea of our independent study project was formed to investigate autonomous driving algorithms, with a focus on calculating the current location of the car using sensor data of the inertia measurement unit (IMU).

PiCar, an open sourced autonomous car platform, is used in our research. This platform was first established by a group of WashU students and features its affordability and accessibility. It allows the freedom to add sensors and control them using Arduino or RaspberryPi which is desired in our research. Using PiCar, testing and experimenting with self-driving algorithms are made possible.

## **2. Proposal & Objectives**

There are three main objectives of our research:

- 1) Building a working PiCar to carry out experiments
- 2) Investigating IMU data
- 3) Design algorithms for self-driving navigation

First, working PiCar has to be built so that an autonomous driving system can be implemented. After the system has been implemented, IMU data then can be collected and explored through experiments. Finally, algorithms of locating the current location of the car based on IMU data can be developed.

## **3. Experimental Design**

### **a. Mechanical structure of the car**

A list of major parts used in our PiCar:

- 1) TrackStar 1/18<sup>th</sup> scale 12T Brushless Power System: motor & ESC
- 2) Turnigy servo motor
- 3) Remote controller & receiver
- 4) 11.1V LiPo battery
- 5) Power distribution board
- 6) Arduino Uno
- 7) RaspberryPi
- 8) Breadboard
- 9) SparkFun 9DoF Sensor Stick
- 10) LEDs
- 11) Assorted jumper wires
- 12) Assorted resistors

The mechanical structure of our PiCar can be viewed as three layers, as shown in Figure 1.

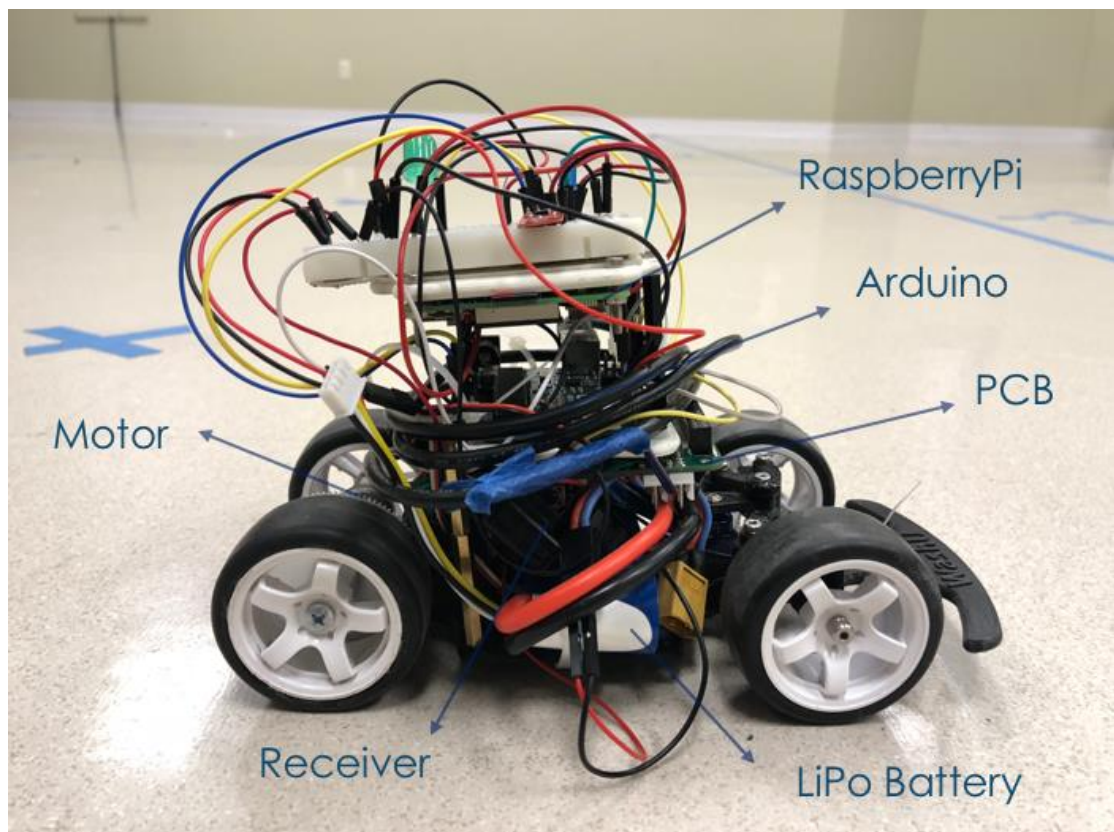


Figure 1. The side view of our PiCar

The chassis of the car constitutes the bottom layer, where the motor, servo and wheels are attached to. The battery, the receiver and the ESC were placed on top the chassis. (A more detailed list of parts and instructions to assemble the basic structure of car can be found on Instructables.) Four copper rods were screwed into the chassis to support a 3D-printed platform which started the second layer of the PiCar structure. The power distribution board was fixed to the bottom of the second layer platform upside down to save space, while the Arduino was fixed on top. Above the second layer, a third layer was built to support the RaspberryPi and a breadboard. The RaspberryPi was fixed upside down with screws. The breadboard was stuck on top of the third layer and provides connections to the IMU sensor stick, LEDs and resistors as shown in Figure 2.

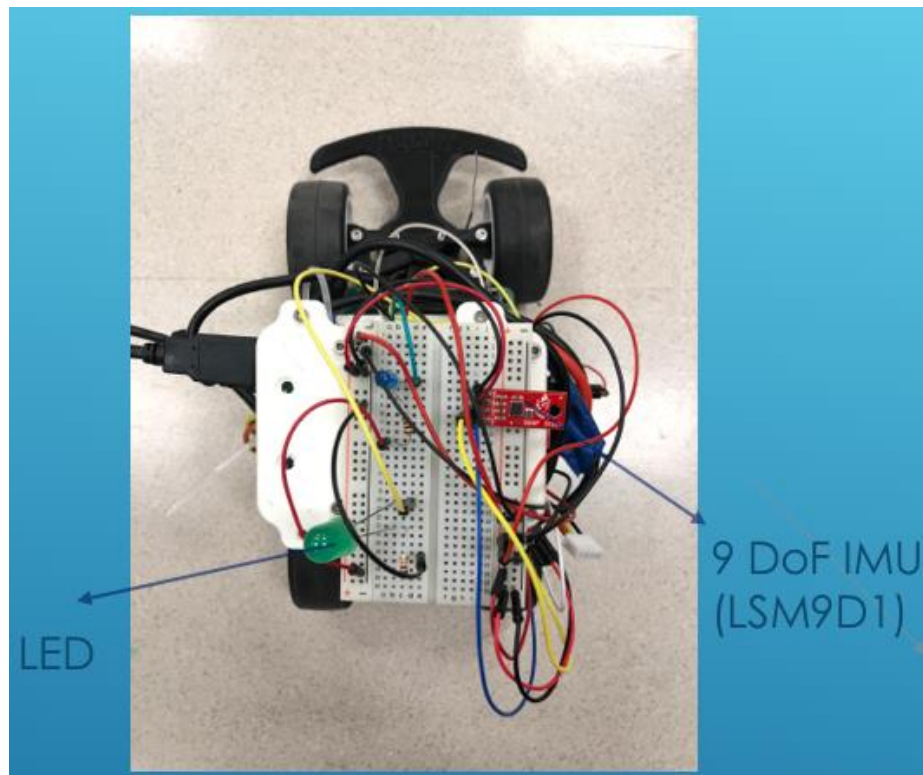


Figure 2. The top view of our PiCar

The IMU used has a LSM9DS1 chip and takes measurements with 9 degrees of freedom: The accelerometer measures the acceleration x, y, and z axis; the gyroscope measures the angular velocity of x, y, z axis; the magnetometer measures the magnetic field vector in x, y, z axis.



Figure 3. SparkFun 9DoF Sensor Stick

## b. System integration

The system of Picar involves Arduino as an information hub which collects raw data from IMU and sending `throttle` and `steer` signal to the motor-servo system. Arduino also sends current heading data, current acceleration data and current throttle value to the Raspberry Pi. The picar can run in two modes to facilitate different goals of experiment. Under driver control mode, Arduino receives commands to drive the motor-servo system from the remote control. Under self-navigation mode, the Arduino reads feedback sent from Raspberry Pi to control the car. The figure 4 below illustrates the overall system setting of Picar.

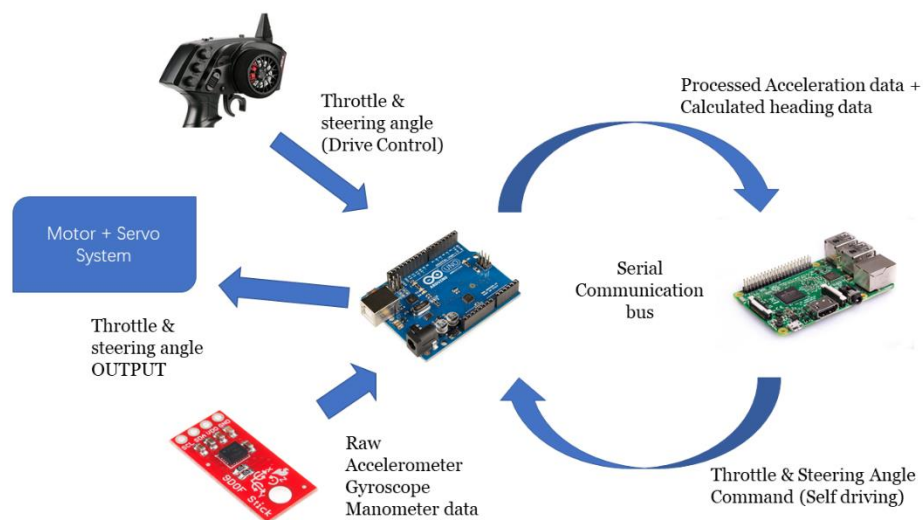


figure 4: Illustration of system setup of Picar

### I. Serial communication between Raspberry Pi and Arduino

The communication between Pi and Arduino is built on the serial communication. The package `pyserial` is used in python program on Raspberry pi to receive byte sent from Arduino. Arduino uses its built-in

`Serial` package to send data. All data transferred through serial are bytes. On Arduino side, it's necessary to convert the read in data from char to integer based on ASCII table. The `readFromPi` function in `runningArduinoWithoutCom.ino` is a good example to illustrate the conversion. On Raspberry Pi, it's easier to save one line of data into a buffer of bytes and convert it to String with built-in function `bytes.decode()`. The python program stores the input data sent from Arduino into String `str_line`. This enables the Raspberry Pi to easily extract the information if the format of data is specified.

All data sent between Arduino and Raspberry Pi, including current heading, acceleration, or even the steering angle feedback from Raspberry Pi, are initialized with a specific index. The indexes are specified for each data type so that the receiver understands how to deal with it properly. Either Raspberry Pi or Arduino only sends one type of data per line. For example, if the heading data with the value of 23.4 is sent from Arduino, it will put "H23.4" byte by byte onto the communications stream. The first character 'H' is the specified initial index for heading data. The `readline` function in `pyserial` package puts one line of data into bytes buffer. Then the buffer is converted to a string. The Python program extracts the data out as a float and continues its computation and communication. Information about other data types are included and specified in the Table 1 and 2 below.

Table 1: initial index explanation for Arduino-to-Pi communication

From Arduino to Raspberry Pi		
Initial Index	Data type	How receiver processes it
'T'	Current Throttle Value	Extract the integer following the 'H'

'A'	Current acceleration data	The acceleration is a vector with two dimensions— $A_x$ and $A_y$ —separated by a comma when sent. The program first find position of comma and extracts two floats before and after comma out.
'H'	Current Heading calculation	Extract the float following the 'H'; output all files and shutdown Raspberry Pi whenever 70 is the integer after 'H.'

Table 2: initial index explanation for Pi-to-Arduino communication

From Raspberry Pi to Arduino		
Initial Index	Data type	How receiver processes it
'S'	Next steering angle the Arduino needs to write to Arduino	Convert the following two bytes (the appropriate signal ranges below 100) to integer

## II. Reading Values from IMU

Most IMUs are communicated with Arduino with I2C, a protocol designed for communication between one master and multiple slaves. Installing the [library](#) for LSM9DS1 will automatically setup the protocol of communication. The raw data of  $A_x$ ,  $A_y$ ,  $M_x$ ,  $M_y$  (acceleration along x-axis, acceleration along y-axis, magnetic reading from x-axis, magnetic y-axis) are read from IMU. The `calcAccel` function in the IMU library is used to calculate the acceleration under the unit g ( $9.8\text{m/s}^2$ ). The `printAttitude` function is overridden to produce the ideal heading which will be specified later in the report.

### III. Reading from Remote control

To facilitate experiment for different with different purposes, Arduino is enabled to read from remote control. In this system, the remote control is regarded as a transmitter, and a receiver is attached to Arduino. Connecting the receiver pins to Arduino pins and using `PulseIn` makes the Arduino store its reading into int variables. The reading from the remote control is linearly mapped to appropriate ranges of integers to control motor and steering angle. For example, if the trigger is not pulled, the reading oscillates from around 1450, corresponding to throttle value of 90—motor stops running. If the trigger is pulled down, the car runs forward and vice versa.

The kill switch function is achieved by sacrificing the function of running motor reversely, which has little utility under current experimental setting. Whenever Arduino detects that the trigger is pulled up (reading far less than 1400), it shifts to the state of not functioning and sending stop signal to raspberry Pi. Details of remote control filtering and kill switch functionality will be discussed later in Challenge Section.

### IV. Servo-motor system

Arduino controls the servo-motor system, motion of the entire car, with the built-in `Servo` package. Arduino wave signals to pins that control motor or servo. To control a motor, the Arduino needs to first output its waveform to ESC (electrical speed controller) which translates the signal to control the motor running or not. To control a servo, Arduino just needs to output waveform with specific magnitude to the servo data pin. For motor control, signal values range from 0 to 180 where 0 stands for full throttle in reverse direction, 180 stands for full throttle in positive direction, and 90 is the not running signal. The signal for servo is theoretically same as motor's; however, due to the unbalanced mechanical structure, it's detected that the signal 83 can correctly forces the front wheels to face forward. The upper and lower limit of servo signals are selected so that the rotation of servo doesn't physically interfere other parts of the car.



	Motor	Servo
Full throttle forward/left	180	55
Full throttle backward/right	0	111
Middle	90	83
Normal forward/left for loop running	108	69
Normal backward/right	70	97

In figure 5 below, the wiring schematics of Arduino and Raspberry Pi are shown. A customized PCB (designed Meizhi) is used to lower down the 11V voltage input from battery to steady 5V output to power Raspberry Pi. The SCL and SDA lines IMU are connected to #4 and #5 of analog, constructing a I2C communication. There are three wires connecting ESC and motor; connection is only needed for wires with the same color. There are also two LED connects to Raspberry Pi and Arduino for debugging purpose. The LED connected to Arduino will be on when Arduino starts communication with Raspberry Pi. The LED connected to Raspberry Pi will blink at 1Hz whenever the python program has done outputting its csv file and ready to shutdown. The LED will be continuously on for 6 seconds before the Raspberry Pi really shutdown itself.

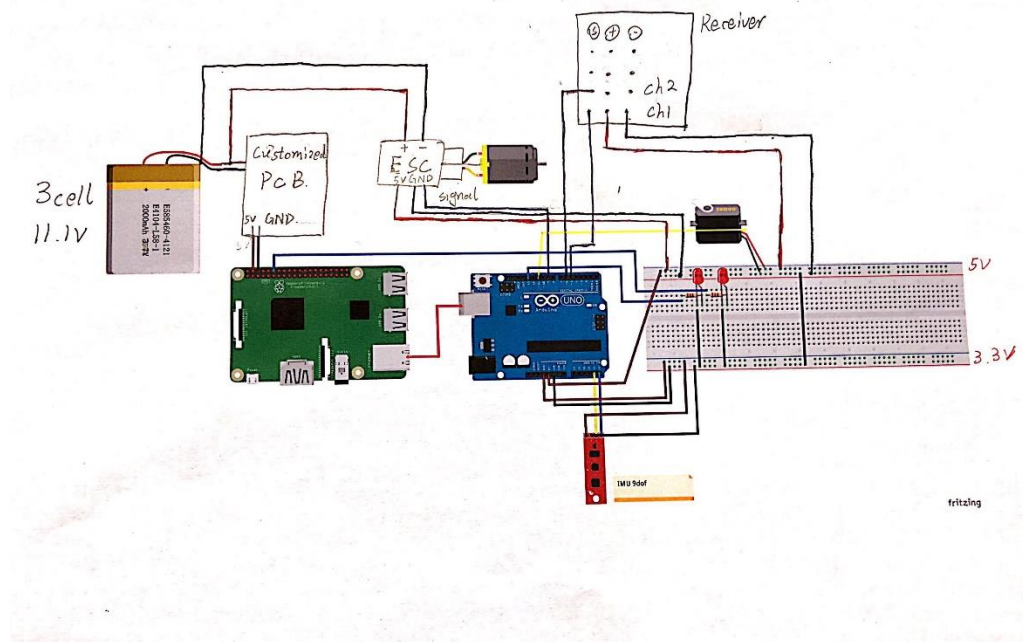


Figure 5 : Detailed wiring of Picar system

### c. Program flow

#### I. Heading experiment

Usually the reading magnetic sensor of IMU is significantly different from expectation. The first step is to find the correct offset for  $M_x$ , and  $M_y$  reading to generate a more robust heading calculation. Theoretically speaking, both  $M_x$  and  $M_y$  should have the behavior as a sine wave oscillating around 0 if the sensor is rotated 360 degrees. The appropriate offset could be calculated after the maximum and minimum of  $M_x$  and  $M_y$  are found. The program `imuCalibrationv2.ino` is the Arduino program that is being used to help through the calibration process. The program flow is illustrated in the diagram below. The Arduino will read from IMU for each 50ms (This is almost the upper limit of sampling rate; it is explained later in detail in later section). The Arduino program calculates the mean of the most recent five readings from IMU and uses the average to calculate headings. The pattern is printed so that Serial plotter can generate the diagram of  $M_x$ ,  $M_y$  against time. Observing the plot and finding the maximum and minimum of magnetic reading with the help of Serial monitor gives a correctly offset  $M_x$  and  $M_y$ . If the offset is

picked correctly, Mx and My behave like sinusoidal wave and the heading ranges from 360 to 0.

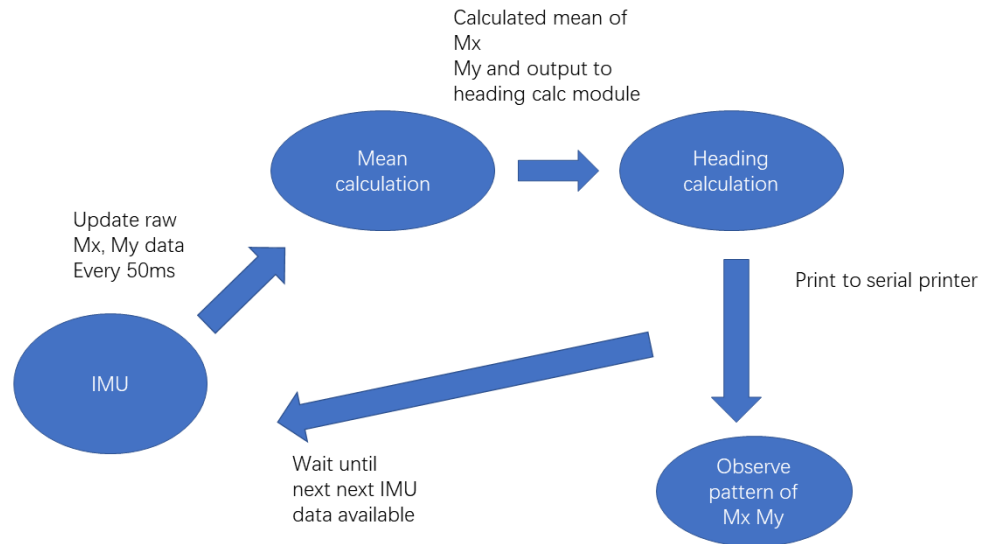


Figure 6: Illustration of heading experiment program flow

## II. Uniform circular experiment

Under the experiment setup, the picar is running in a circular loop in counterclockwise direction with initial position facing north. Arduino collects data and sends them to Raspberry Pi; Raspberry Pi stores the data whenever the kill switch signal is asserted for future analysis purpose.

Under experimental setup, the Arduino is reading from IMU every 50ms. It also read signals for throttle and servo angle and map them to appropriate constant (the experimental setup is to investigate circular uniform motion).

The running throttle is set to 108 and servo angle is mapped to 69/97.

Whenever Arduino realizes the trigger is pulled up (reading significantly less than 1400), it maps the throttle value to 70 and sent to Raspberry Pi.

Raspberry Pi will jump out of the reading data loop and exports the csv file.

The Figure 7 below illustrates the program flow for uniform circular motion experiment.

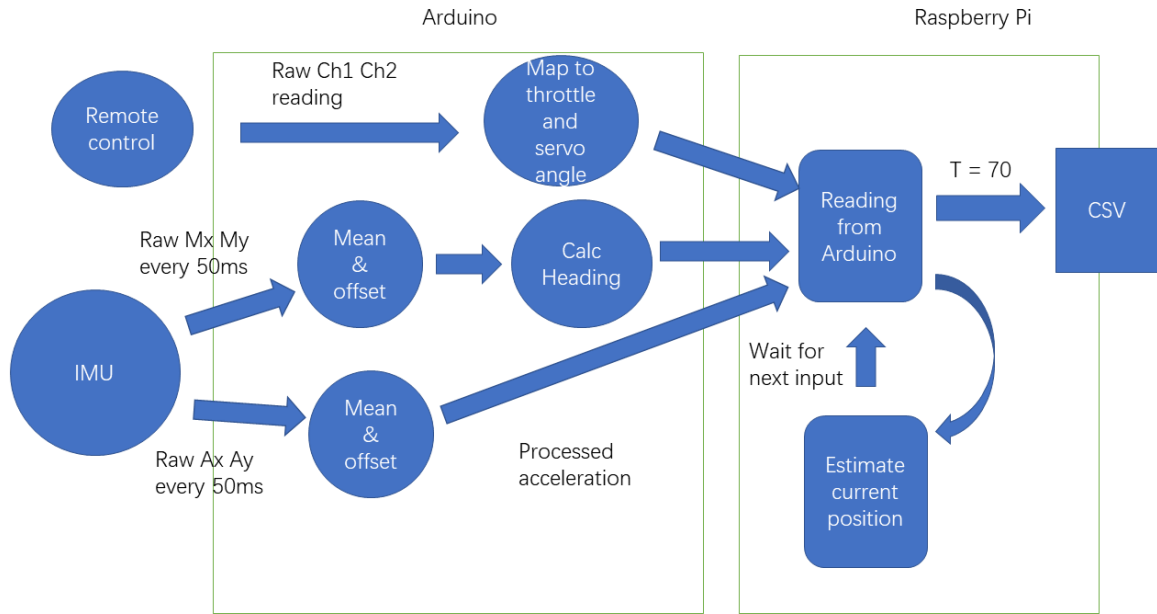


Figure 7: Program flow for uniform circular motion experiment.

In python program, all inputs from IMU are stored in different lists. New data entries like calculated acceleration, velocity and displacement in East and North direction are also stored in respective lists for further reference and analysis. There are seven lists that are outputted to local csv file whenever the trigger is pulled up-- `listA_x`, `listA_y`, `listAcc_E`, `listAcc_N`, `listV_E`, `listV_N`, `listD_E`, `listD_N`, `listH`. The output functionality is achieved with built-in csv module. Each list is written into separate lines into one csv file for experiment running.

#### 4. General theoretical support

As the goal of our algorithm is determining the current location of the car, multiple equations are needed. The data directly provided by the IMU are accelerations, angular accelerations and magnetic field strengths. To calculate the displacement of the car with respect to the reference coordinate, a double integration of accelerations needs to be performed. However, the IMU outputs are measured against the moving frame of IMU, so a rigid body transformation is required to transform the accelerations to the reference frame of interest.

##### d. Rigid body transformation

A 3x3 rigid body transformation matrix  $R$  translates and clockwise rotates a 2D vector from a coordinate to another coordinate for an angle  $\theta$ . It consists of a 2x2 rotation matrix on its upper left corner, and a translation column vector in column 3:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

For our purposes, we assume the car runs on a level surface. Thus, only the x and y accelerations are considered. As shown in Figure 8, the moving frame of the IMU is marked by axis Y and X reference frame, while the absolute reference frame of interest is marked by axis E and N.

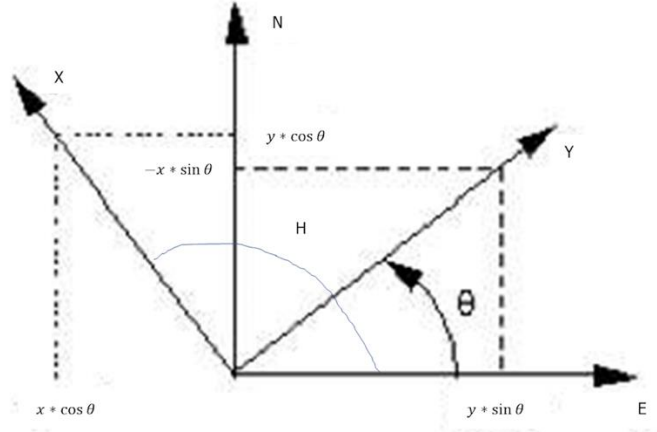


Figure 8. Rigid body transformation: rotation

Then the transformed accelerations are represented by:

$$\begin{aligned} a_E &= a_y \cos\theta - a_x \sin\theta \\ a_N &= a_y \sin\theta + a_x \cos\theta \end{aligned}$$

#### e. Heading calculation

To obtain the angle  $\theta$ , the orientation of the moving frame needs to be known.

While the conventional method of obtaining  $\theta$  is using data from the gyroscope, this method requires integration with respect to time and is thus prone to error. As an alternative, the calculation of headings based on magnetometer data is used in our research. This approach utilizes the fact that at a specific location, the earth magnetic field vector is constant, as shown in Figure 9.

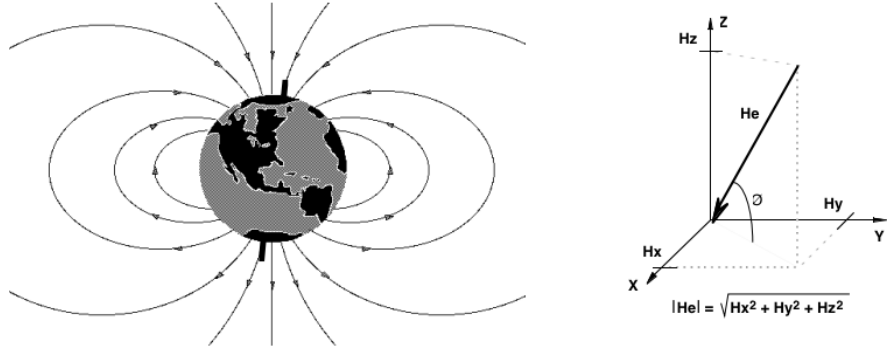


Figure 9. Magnetic field of the earth (left) and a magnetic strength vector (right)

Thus, as the car changes heading and the magnetometer rotates, the direction the car heading to can be calculated using the relation between magnetic field strength on the x and y direction as shown in Figure 10. Through these relations, we can obtain the heading of the car in the earth frame, which is an advantage over using gyroscope data.

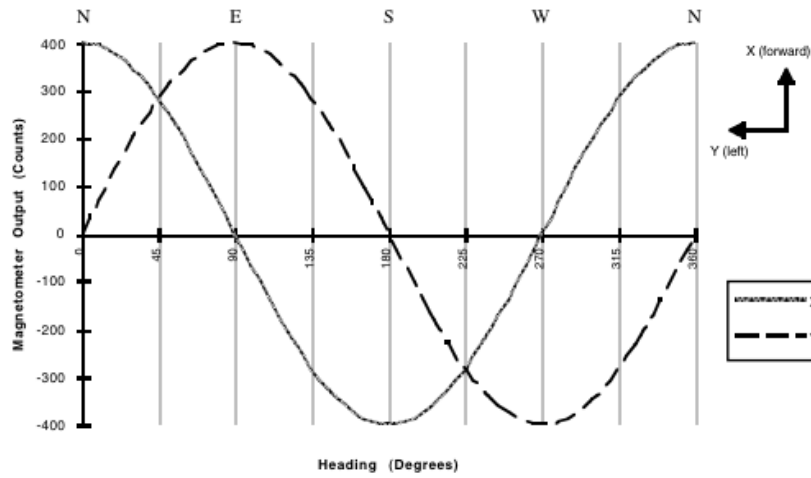


Figure 10. The relations between magnetometer outputs and heading

The calculations of heading angle in the earth frame with respect to the East are listed as the equations below:

$$\begin{aligned} \text{Direction (y>0)} &= 90 - [\text{arcTAN}(x/y)] * 180/\pi \\ \text{Direction (y<0)} &= 270 - [\text{arcTAN}(x/y)] * 180/\pi \\ \text{Direction (y=0, x<0)} &= 180.0 \\ \text{Direction (y=0, x>0)} &= 0.0 \end{aligned}$$

From the heading angle  $H$ , the angle  $\theta$  used in the rigid body transformation can be obtained using  $\theta = H - 90^\circ$ .

#### f. Approximation of integration

With the accelerations in the earth frame, velocities in the East and North can be calculated. Since the accelerations are non-predictable when the car runs, direct integration is not possible. A midpoint Riemann sum is applied to estimate the area under the acceleration curve, as shown in Figure 11. The actual measurements  $a_i$  are taken every  $\Delta t$  at  $t_i$  time, and the height of the rectangles are approximated using:

$$a_{i+1}^* = \frac{a_i + a_{i+1}}{2}$$

Thus, the estimated velocity at  $t_{i+1}$  is given by:

$$v_{i+1}^* = v_i^* + a_{i+1}^* \Delta t.$$

Similarly, the displacement at  $t_{i+1}$  can be obtained by:

$$x_{i+1}^* = x_i^* + v_{i+1}^* \Delta t.$$

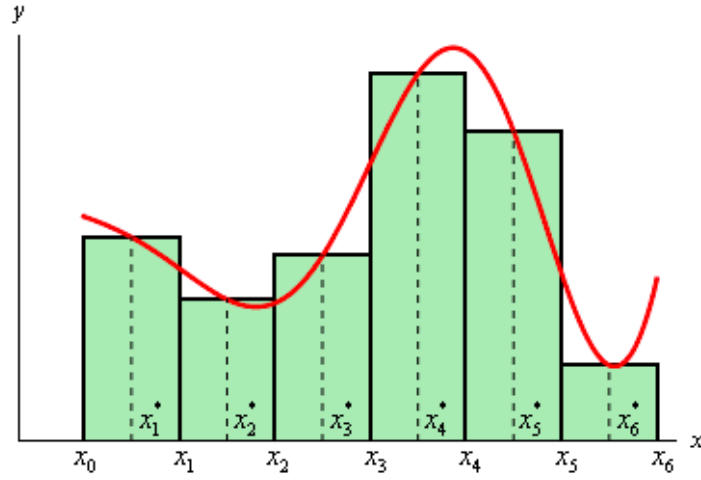


Figure 11. Midpoint Riemann sum.

Thus, the displacement in East and North direction  $x_E$  and  $x_N$  can be obtained as an estimate of the current location of the car with respect to its starting point.

## 5. Experimental case analysis

### a. Theoretical analysis of uniform circular motion

High school physics tells the nature of uniform circular motion. There is no acceleration component along the tangential direction of trajectory. However, there is a acceleration vector with constant magnitude that is always normal to the direction of moving. Thus, the acceleration and velocity along East or North direction should act with sine wave behavior. The theoretical result of acceleration and velocity along East and North direction are simulated under the assumption that radius is 0.5m, centripetal acceleration is  $0.4655 \text{ m/s}^2$ , the object starts a uniform circular motion counterclockwise at (0.5,0). Under such setting, the tangential speed is calculated to be 1.4175 m/s. The acceleration and velocity should have behavior predicted by the following equations.

$$Accel_N = -0.4655 * \sin(\theta)$$

$$Accel_E = -0.4655 * \cos(\theta)$$

$$Velocity_N = 1.4175 - 0.4655 * \cos(\theta)$$

$$Velocity_E = -0.4655 * \sin(\theta)$$

Where  $\theta$  is the angle between the position vector and positive E-axis.

The results are also simulated in Excel—shown in diagrams below. In the first two plots, the horizontal axis represent how many digital steps are from the initial start of 0. Each step has length of  $2\pi/1000$ .

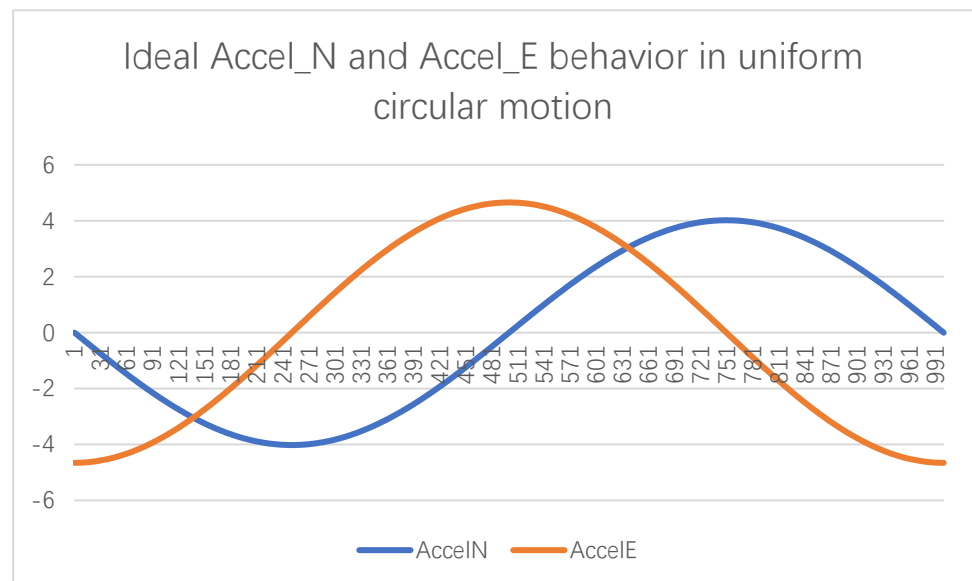


Figure 12: Theoretical plot of one period of  $Accel_N$  and  $Accel_E$



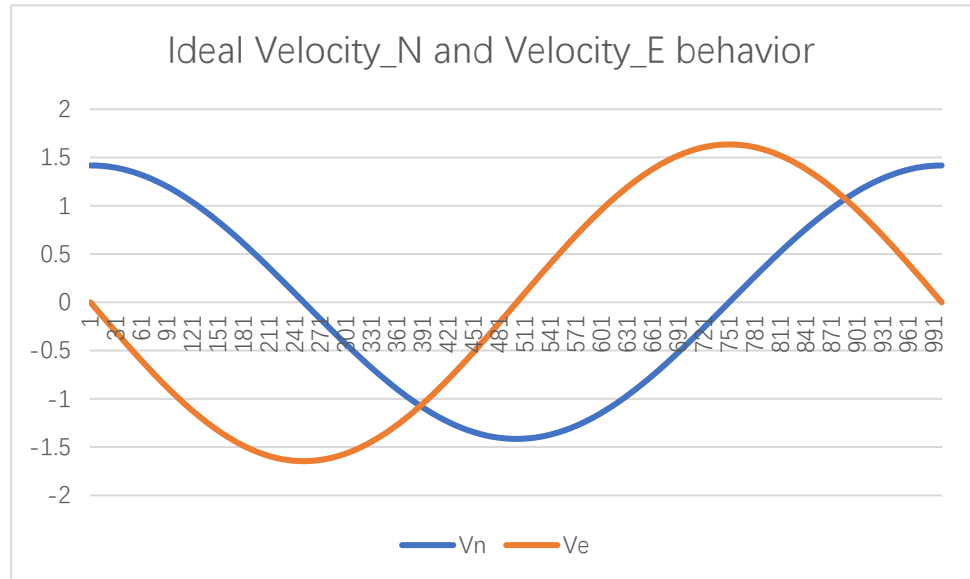


Figure 12: Theoretical plot of one period of  $Velocity_N$  and  $Velocity_E$

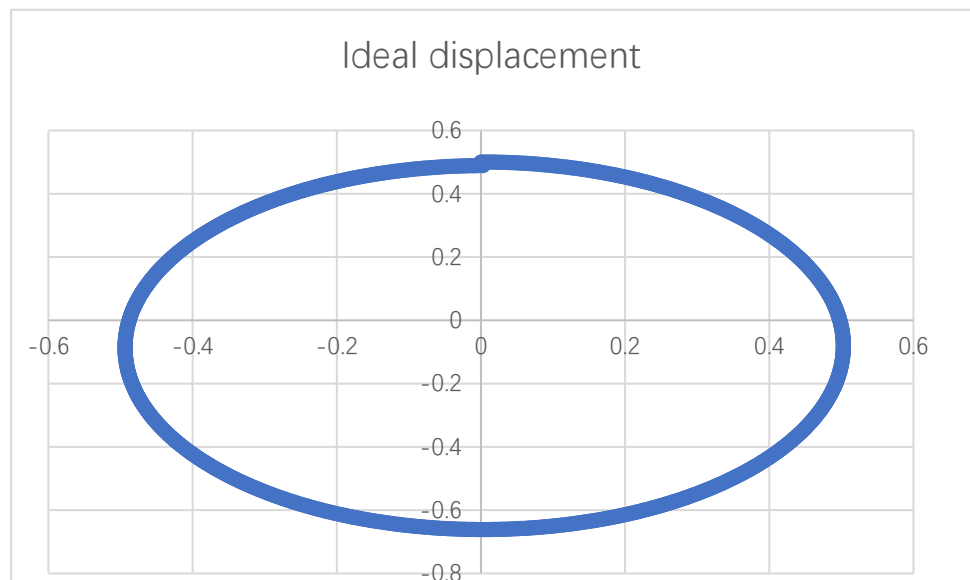


Figure 13: Theoretical plot of uniform circular motion trajectory of one period

## b. Methodology for data processing

### i. Method for magnetic sensor reading offset

As mentioned above, there is a significant difference of expected behavior of  $M_x$  and  $M_y$  and their realistic behavior. Assumed that the maximum and minimum value of  $M_x$  or  $M_y$  are found with the `imu calibration.ino` program. The correct offset value should be given by the equation below:

$$offset = \frac{max - min}{2} - max = -\frac{max + min}{2}$$

Where *max* and *min* are measured maximum and minimum of *Mx* and *My*.

ii. Method for acceleration reading offset

To correctly offset the acceleration vector, the theoretical value of *Ax* and *Ay*. Because *Ax* is pointing toward the direction of car' head and *Ay* is pointing to the center of the circular motion. With period of car running measured and recorded as *T* (s) and radius of circular motion measured as *r*(m). The theoretical tangential speed can be calculated by:

$$v = \frac{2\pi r}{T}$$

Where the dominator stands for the circumference of the circle and the numerator is the time consumed for the car to finish the loop.

With average speed calculated, the theoretical magnitude of centripetal acceleration can be calculated with the equation below:

$$a_c = \frac{v^2}{r}$$

Where *v* is the theoretical speed and *r* is the radius measured in meter.

The average of value of *Ax* and *Ay* are used to generate a valid offset for the value. With the average of *Ay* in periodic loop denoted as *Ay<sub>ave</sub>* and average of *Ax* in periodic loop denoted as *Ax<sub>ave</sub>*, the offset for *Ax* and *y* is given the equations below:

$$Ay_{off} = a_c - Ay_{ave}$$

$$Ax_{off} = - Ax_{ave}$$

(keep in mind that the theoretical value of *Ax* is 0, it is on the tangential direction of centripetal acceleration)

Furthermore, the initial value of *VN*, velocity on positive North direction, needs an offset. The PiCar can accelerate into its ideal speed in so short time that the integration method of velocity approximation may not be able to catch the peak of acceleration due to low sampling

rate. Thus, a manual offset with the magnitude of calculated theoretical tangential speed is necessary.

## 6. Results & experiment

### a. Heading experiment

#### i. Findings about magnetic sensor inside lab

The offset experiment is first carried out inside Green lab. The PiCar is moved with hand but without any change of its heading. The car is facing south at the very first; it is horizontally moved along positive west direction, then the north direction, then the east direction and finally the south direction. For each direction, the car is moved for one meter. The mx and my reading plot is recorded as below.

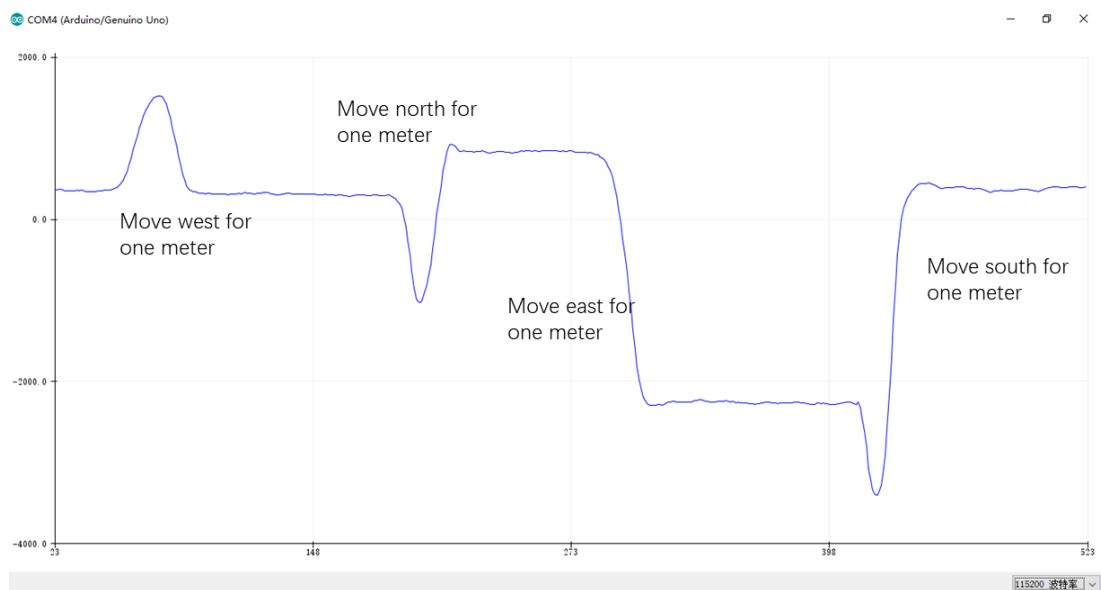


Figure 14: Plot of Mx behavior against time when moving the car without change heading inside lab

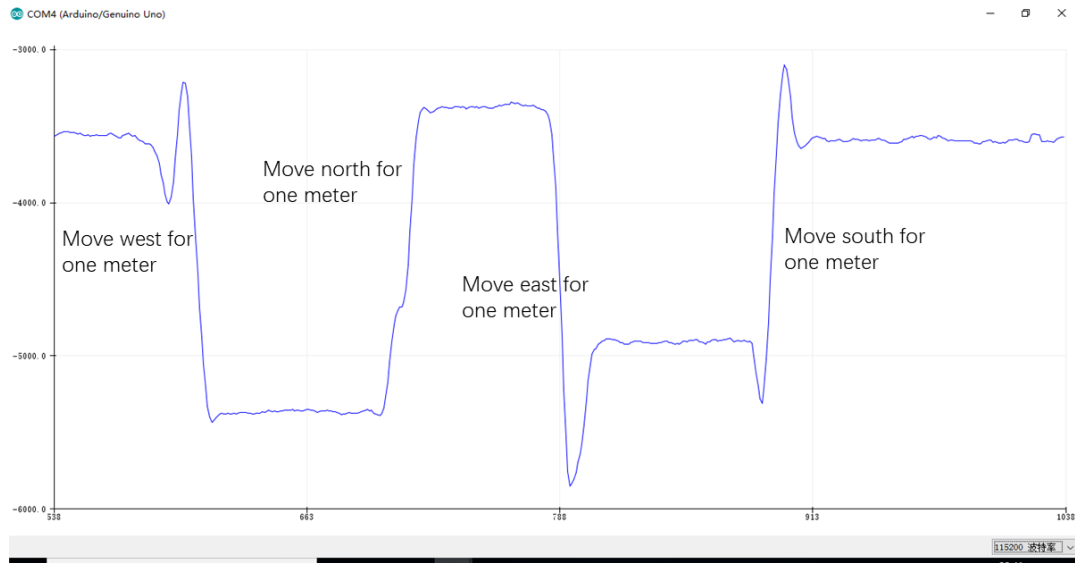


Figure 15: Plot of My behavior against time when moving the car without change heading inside lab

ii. Result of magnetic sensor offset outside lab

The magnetic sensor is also studied outside the Green building (on open ground less magnetic interference). The maximum, minimum, amplitude and calculated offset value are recorded in the Table below:

Table 3: data measured outside Green lab

	Mx	My
Maximum	2400	-3470
Minimum	10	-5800
Amplitude	1195	1165
Offset	-1205	+4635

With the offset value calculated, the Mx and My have the behavior shown in the figure below:

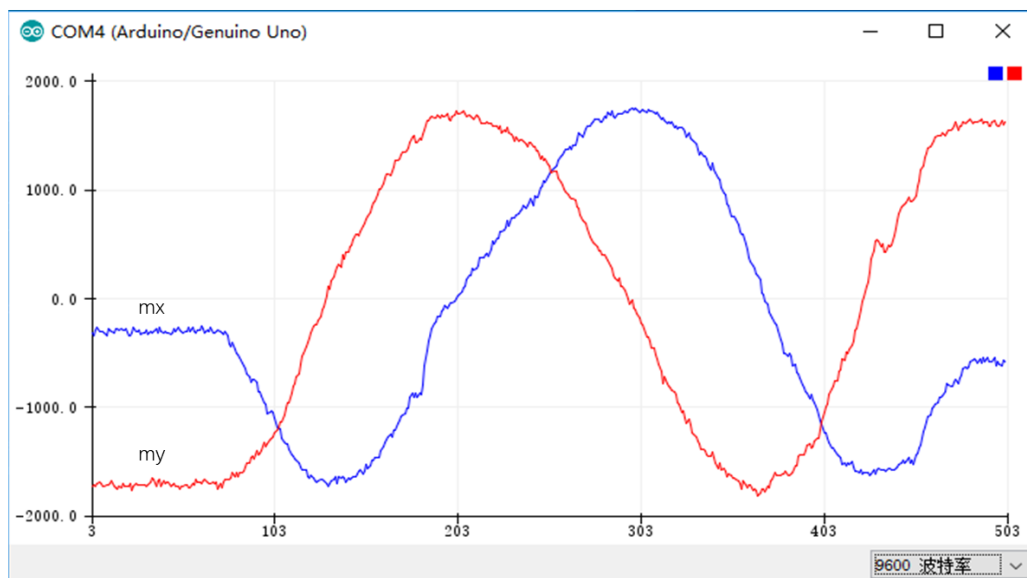


Figure 16: Behavior of Mx and My with offset outside Green

Turning the picar 360 degrees can generate heading values ranging from 360 to 0. It represents the angle from positive East direction and the x-axis of picar. The plot is shown in the diagram below.

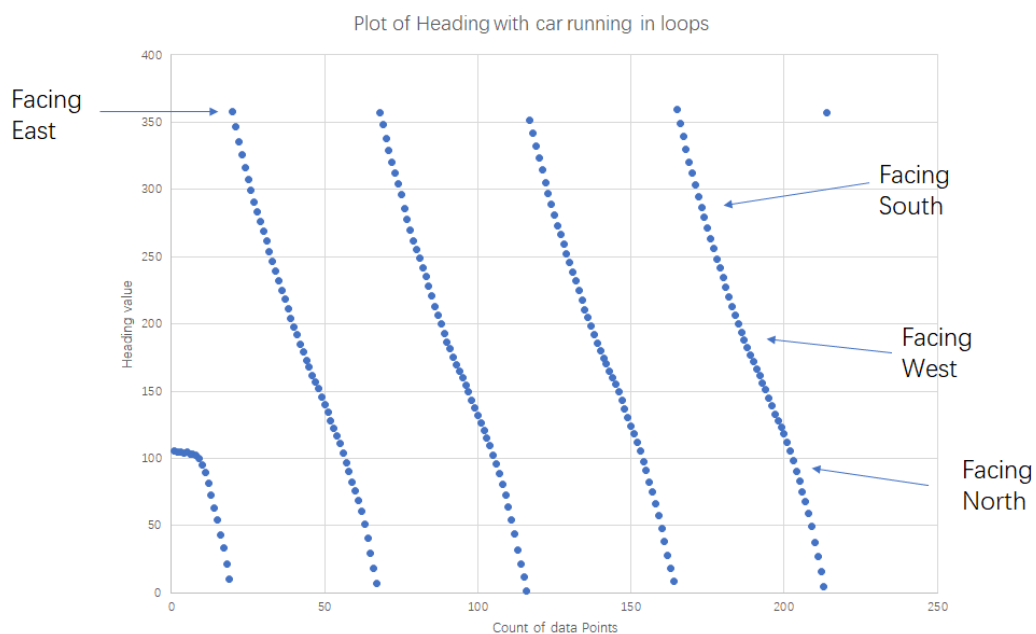


Figure 17: Plot of heading with car turning 360 degrees clockwise

iii. Result of magnetic reading with solder mesh covered inside loop

To investigate how to avoid the problem of magnetic interference, a mesh of solder is created to encompass the IMU. A figure of the encompassed IMU is shown below.

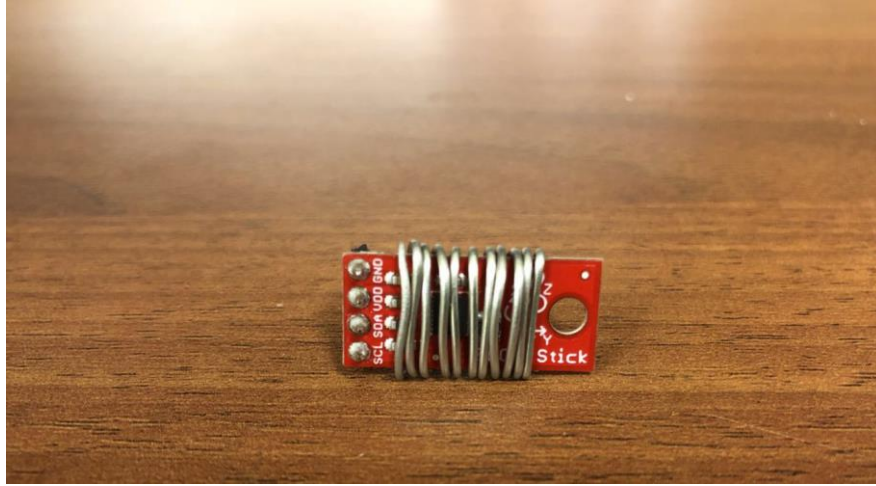


Figure 18: IMU physically encompassed by solder mesh

Similarly, as the IMU is physically encompassed by solder mesh, the picar is moved along the same path detailed in section i and the result of the plot is shown below:

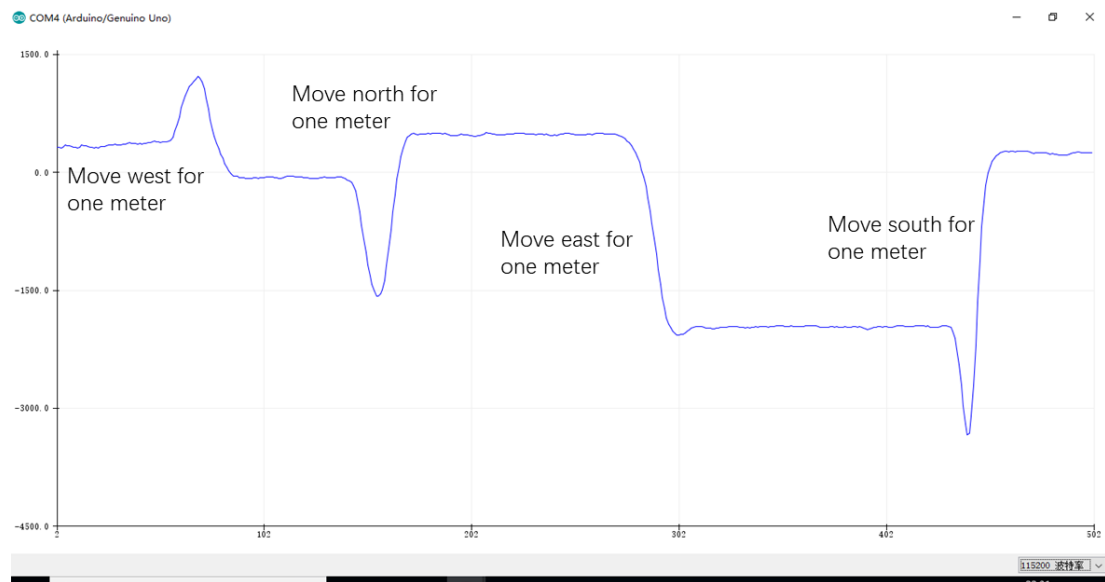


Figure 19: Plot of Mx behavior against time when moving the car without change heading inside lab

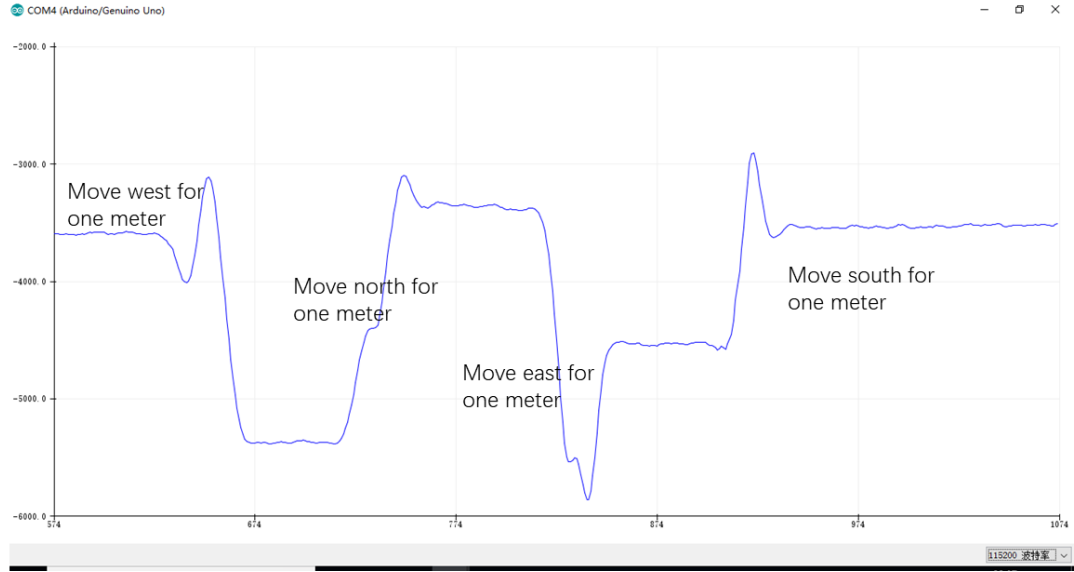


Figure 20: Plot of  $M_x$  behavior against time when moving the car without change heading inside lab

#### b. Uniform circular motion experiment

For the uniform circular motion experiment, we analyzed 4 sets of data for each trial:

- 1) accelerations in IMU frame  $a_y$  and  $a_x$
- 2) heading angle of car in Earth frame  $H$  (in radians)
- 3) accelerations in Earth frame  $a_E$  and  $a_N$
- 4) velocities in Earth frame  $v_E$  and  $v_N$
- 5) displacements in Earth frame  $x_E$  and  $x_N$

The experimental data from one of the trials was extensively plotted for analysis. With offset values of  $+0.3972$   $a_y$  and  $-0.11985$  for  $a_x$  applied before collecting the data, plotted accelerations  $a_y$  and  $a_x$  are shown in the figure below. The mean of  $a_y$  is calculated to be  $0.65763 \text{ m/s}^2$ , and the mean of  $a_x$  is calculated to be  $-0.00077 \text{ m/s}^2$ . The horizontal axis is marked by number of intervals of  $\Delta t$ .

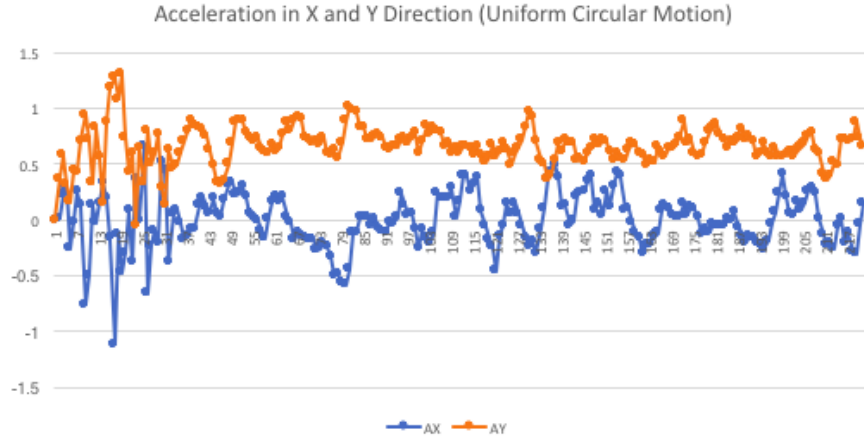


Figure 21. Acceleration in IMU frame  $a_y$  and  $a_x$  VS time  
The calculated heading angles in radians are plotted below.

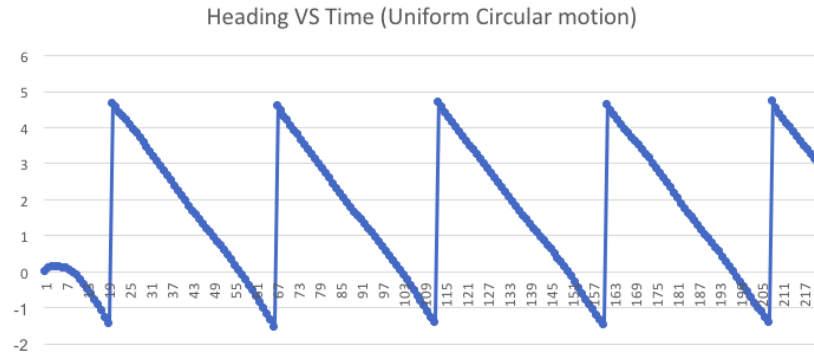


Figure 22. Heading angle H in Earth frame VS time  
The accelerations in Earth frame  $a_E$  and  $a_N$  VS time are plotted below.

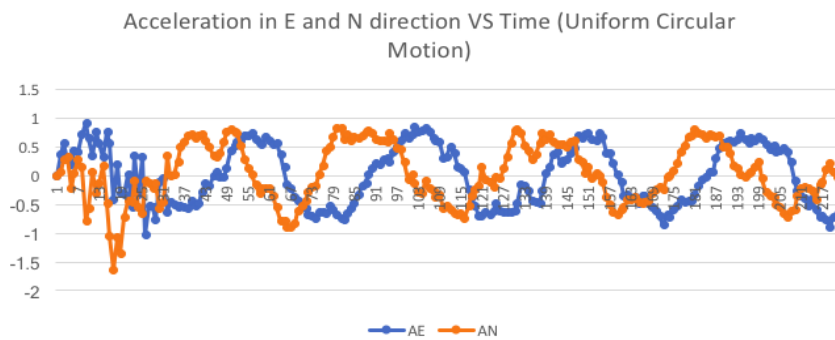


Figure 23. Accelerations in Earth frame  $a_E$  and  $a_N$  VS time  
The calculated velocities in Earth frame  $v_E$  and  $v_N$  VS time are plotted below. With an offset of 2.23 m/s applied for  $v_N$  to compensate for the missed instant acceleration, the mean of  $v_E$  is 0.6549 m/s and the mean of  $v_N$  is 3.3524 m/s.



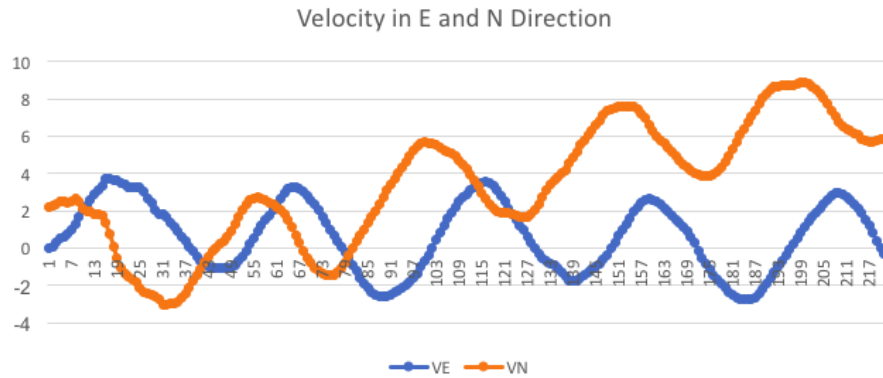


Figure 24. Velocities in Earth frame  $v_E$  and  $v_N$  VS time

The Displacements in Earth frame  $x_N$  VS  $x_E$  are plotted below. This graph traces the spatial movement of the car throughout the uniform motion experiment.

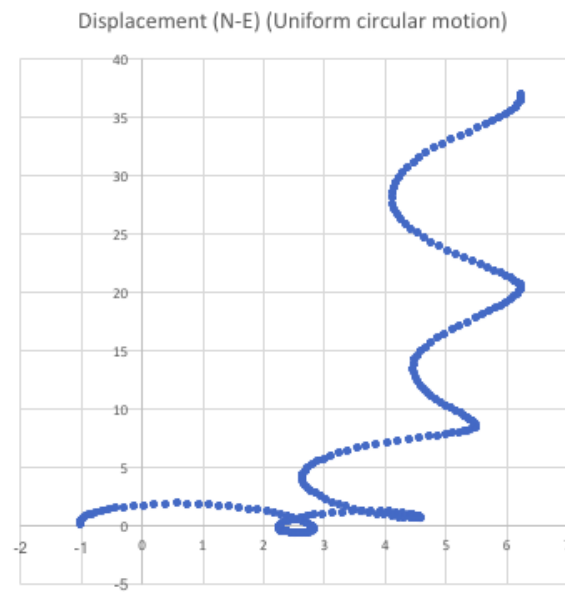


Figure 25. Displacements in Earth frame  $x_N$  VS  $x_E$

## 7. Discussion

### a. Heading experiment

Theoretically speaking, the magnetic vector over a small space (in lab for example) is considered to be constant; namely, moving the car without changing directions should not change the  $M_x$  and  $M_y$  values significantly. However, the result shown above in Figure 22 indicates that there's a great

jump of  $M_x$  and  $M_y$  readings when the car moving without changing directions. This forms a contrast with the magnetic readings and heading calculations outside the lab. It reveals that there is a significant yet unbalanced distributed magnetic field interference in the Green lab. It caused huge irregular jump in  $M_x$  and  $M_y$  reading.

To investigate how to reduce the strong interference, the solder mesh is encompassed around the IMU in an attempt to block the extra magnetic interference. Yet, comparing Figure 19 to Figure 20, it's noticeable that there existed a reduction of jump about 10% to 20%, but the jump of  $M_x$  and  $M_y$  readings are still conspicuous. Thus, the solder mesh might not be an effective to filter out extra magnetic interference.

The experiment outside the lab gives data pattern as expected in theoretical analysis section. The  $M_x$  and  $M_y$  has the behavior of sine wave but with a phase difference about 90 degrees. And the result of heading also followed the pattern as expected. It drops from 360 (facing east) to 0 (facing east); it is rotated along clockwise direction. In the uniform circular motion experiment, the heading in one period has already forms a straight line. The result supports the validity of magnetic sensor reading and the algorithm to calculate the heading as the angle between x-axis of IMU and positive East axis under world coordinate. Yet, the interference in lab leaves as more questions about how to reduce the possible magnetic interference.

#### **b. Uniform circular motion experiment**

Although still noisy, the oscillation patterns of  $a_E$  and  $a_N$  corresponds to the heading change, and conform with the ideal  $a_E$  and  $a_N$  of uniform circular motion. This proves that our calculations for heading angle and the rigid body transformation calculations are correct. The velocity oscillation patterns semi-circular patterns of the displacement also indicate that our integration method is correct.

From the pre-experiment, the ideal centripetal acceleration of this uniform circular motion was calculated to be  $0.6340\text{m/s}^2$ . Compared to this ideal value, the mean of  $a_y$  we obtained is fairly close yet still has a difference around  $0.02\text{m/s}^2$ . This difference indicates that these errors propagated throughout time, and resulted in the accumulating error of  $v_N$ . As can be seen from the Figure 24,  $v_E$  has been oscillating around the horizontal axis as expected. This is due to the fact that  $a_x$  value is much smaller in magnitude and resulted in a much smaller error in value. After integration, the error in  $v_N$  was magnified in displacement  $x_N$ . In the most ideal situation, the traces of the car should be circular and return back to the origin in the end of the trial. It is observed that the trace of the first half of the loop the car ran in the experiment still faithfully reflects the shape of its path. The car only drifted afterwards as time elapsed. At the end of the trail, the  $x_E$  value drifted 5.3m to the east of where it is expected to be, while the  $x_N$  drifted 37.5m from the origin when the trial ended. These errors are significant. Clearly, the errors propagate throughout both time and number of times of integration.

The propagation of error in accelerations is the major contributor of the non-ideal results in displacements. However, this challenge is difficult to resolve because the unpredictability of the accurate offset value. We could only obtain offset value through previous trials, but there are always subtle changes in parameters from trials to trials that could not be eliminated. To prevent errors from propagating, an algorithm that dynamically offsets the value has to be applied.

## 8. Obstacles & Troubleshooting

### a) RaspberryPi Powering

One of the major obstacles we ran into is powering of the RaspberryPi. In the very beginning, powering RaspberryPi using Arduino 5V output was first

attempted. However, RaspberryPi kept rebooting, indicating that the voltage was not stable. This is probably due to that the Arduino was unable to provide a steady current to support RaspberryPi. Later, we switched to powering the RaspberryPi out of the 5V output of ESC. This method worked for a while when a 11.1V 3-cell LiPo battery was used to power the ESC. However, after that battery died accidentally, we had to switch to a 2-cell battery with lower voltage and this method stopped working. Even if we purchased a new 3-cell battery later to power the motor and ESC, the RaspberryPi rebooting problem was solved, but the Pi displayed a yellow lightning symbol indicating low voltage. Since operating in low voltage may result in data loss of the RaspberryPi, we decided that adopting other methods would be a more discrete choice. Before we resolved to power distribution board developed by Meizhi which worked in the end, we also attempted a naïve method: building a voltage divider using resistor to give 5V voltage supply to the RaspberryPi. It turned out to be a failure, as the current going through RaspberryPi is not constant.

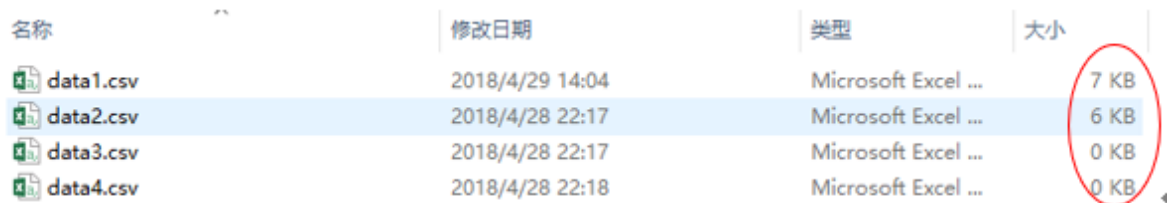
#### **b) Remote control**

As mentioned above, the value Arduino read from remote control is noisy. The primary reason is that it is the spring inside remote control that influences the strength of its output signal. The unstable spring will unavoidably generate an oscillating signal. Thus, direct mapping from the raw remote control to selected range of motor or servo signals causes the servo to shake and the motor to twitch even if nobody pulls the remote control.

One of an easy way that is used in our experiment is to measure the oscillating range of remote control signal when nobody pulls the trigger or turns the knob. Then, all values that fall in the range can be regarded as signal that indicate trigger is not pulled or knob is not turned. Of course, there are many other way to process the reading from remote control. Yet, concerning the fact remote control is not the primary focus of the experiment, we didn't bother applying other methods

### c) Empty output file

In the first a few running of the uniform circular motion experiment, it also exists the problem of empty output file. Sometimes, when the trigger is pulled up to tell Raspberry Pi to output data as csv file and Arduino to stop running, there is only one empty csv file outputted. This appeared to be quite frequent problem which significantly disturb our process of experiment. For example, in our first running outside, there are two out of four files that are empty (shown below in Figure 26).



名称	修改日期	类型	大小
data1.csv	2018/4/29 14:04	Microsoft Excel ...	7 KB
data2.csv	2018/4/28 22:17	Microsoft Excel ...	6 KB
data3.csv	2018/4/28 22:17	Microsoft Excel ...	0 KB
data4.csv	2018/4/28 22:18	Microsoft Excel ...	0 KB

Figure 26: Frequent empty file output

In fact, empty file output appears when one running of experiment is finished but the power supply of the Raspberry is manually cut off so quickly that the RAM of Raspberry Pi just lost itself. It is tested that if we let python program to shutdown itself instead of manually unplug the power supply the file can be saved. Thus, we add a snippet of code that lights up a LED to indicate the success of outputting csv file and then shutdown the Raspberry Pi. This mechanism effectively protects the security and validity of our data, paving our way for future experiment.

### d) Compute average time with $O(1)$ runtime

Because the IMU reading of acceleration data and magnetic reading are noisy itself, we take the average of the most recent five readings as the value used in further calculation and communication. To minimize the time of the calculation, we devise a more elegant way to compute average rather than summing the most recent five and doing the division.

We can view the average calculation as moving buffer with length of five. The value that is going to leave the buffer can be called `oldValue`; the new reading from IMU that will enter the buffer is `newValue`. If the previous mean is

known and length of buffer, the new mean calculated with O(1) time by the equation below:

$$newMean = oldMean + \frac{newValue - oldValue}{bufferLength}$$

This will effectively cut the time of average computation which is frequently called by both Heading calculation function and acceleration calculation function.

**e) Enable different modes of driving**

In most of our experimental setup, the picar is running under driver control mode to make an easier way to get the data. Yet, in realistic scenario, we have to implement the functionality to allow the car drive based on Raspberry Pi feedback. We deal with problem by identifying a special initial signal sent from python. In our design, if somebody wants the picar to run in driver control mode, the python program would send ‘!’ to Arduino. If the self-navigation mode is the ideal testing mode, python sends ‘?’ to the stream. Identifying different signals allow Arduino knows which mode to enter. In both modes, the kill switch is always added for safety reason.

For now, the self-navigation mode can only give constant steering angle feedback because most parts of our position are not valid and ideal. However, the idea and structure of the program has already been established and tested for further experiment.

**f) SPI communication → serial (slow calculation) serial communication is enough to transfer**

When we first get touch with the Picar system, we expect to use SPI communication instead of serial communication for data transfer between Arduino and Raspberry Pi. SPI has a primary advantage in its communication speed. Practically speed, the maximum communication speed of SPI can reach 10 Mbps (10<sup>7</sup> bits/second) while the serial communication can only has baud rate of 115200 bits/second—100 times less than SPI!

However, testing the running time of Arduino and python program, we find out that it takes 38 ms for Arduino to finish operation of reading from IMU,

reading two values from remote control, sending it to Raspberryp Pi and writing to servo-motor combo. It takes python program 10ms at most to process each data entry sent from Arduino. Thus, at most ideal situation, the period to update IMU data and send to Raspberry Pi is about 50 ms. During the period, we only need to send 24 bytes of data—including acceleration in two dimensions, calculated heading data and current throttle value. The serial bus at baud rate of 115200 bits/second can send all 24 bytes in 1.67ms, which only takes less 5% of our entire operation time. Thus, considering minor effect of the slow communication speed of serial communication, we embrace it as an easier to facilitate our experiment.

In fact, we find that it is the process of reading remote control value but not sending Serial data that takes the time of computation. Reading values from two channels of the receiver takes and executing all other operations take 38ms. However, reading values form one channel of the receiver takes and executing all other operations only take 22ms. Thus, it is evident that communication between Raspberry Pi and Arduino is not the primary impedance of increasing sampling rate. Also, as long as we need to read from Remote control as a kill switch function or changing mode signal for any safety or experimental reason, the sampling rate is really hard to increase. It does take a short period of time to finish all operations and send to Raspberry Pi.

#### **g) Raspberry Pi wireless interface not found**

When the first time the Raspberry Pi booted, it reported error “wireless interface not found” when we tried to connect to the Internet. This issue occurred because the wireless interface configuration was changed for some reason, so setting it back to the default configuration would fix the problem. Thanks to a post in raspberrypi.org, typing the code in the Figure 27 below to `/etc/network/interfaces.d` can reset the configuration to default. This solution was proved to work after our attempt.

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
```

Figure 27. Instructions to fix wireless interface

#### **h) Keyboard not matching**

Another issue we ran into about RaspberryPi configuration is that the keyboard used did not match the configuration, so that the symbol “#” which is extensively used to comment in python code could not be typed. This issue was resolved after we changed the keyboard mapping configuration from UK to US.

#### **i) ESC calibration & initial value**

In the beginning, we could not get the assembled car to run by sending `esc.write()` signals from the Arduino. After looking up the ESC blinking patterns, we found the cause of the problem: the ESC needs to be calibrated. According to a tutorial on [droneandrovs.wordpress.com](http://droneandrovs.wordpress.com), the instructions for calibrating the IMU are as follows:

- i. Power up the ESC while the having maximum forward throttle applied.
- ii. You’ll hear a tone and some beeps and after a while (usually 2 seconds) you’ll hear a confirmation tone and the led will blink a few times with a different color: this indicates that the ESC has measured the wavelength of max throttle.



- iii. At this point apply zero throttle (in a fwd/reverse ESC this means full throttle reverse), wait again few seconds for the tones and led to blink: full reverse measured.
- iv. Then move to central (only for forward/reverse ESCs) and wait again for the tone and blinks.
- v. Have fun with the motors.

We calibrated the IMU following the instructions, and this solved the issue.

The other thing that hindered the functioning of the ESC is that an initial signal of value 90 has to be present before sending other signals for ESC to function properly. When writing the Arduino code, we found that the motor would not turn if only values other than 90 were written to the ESC.

## **9. Future Work**

### **a. Magnetic interference**

It is found that there is a strong magnetic interference in Green which disturbs the reading of our magnetic sensor. Shielding are two of the most ways to deal with magnetic interference. Shielding is the way to block out the magnetic field by placing a ferromagnetic case around the IMU. For those kinds of material, the electron distribution inside will be influenced by the magnetic field strength that encompass it. It will effectively reduce the strength of the magnetic interference. The figure below illustrates the effect of ferromagnetic material on magnetic field strength.

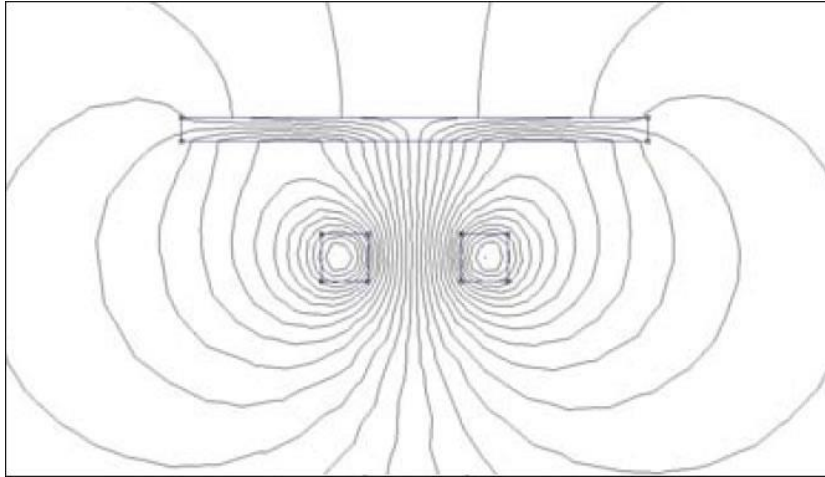


Figure 28: Reduction of magnetic field strength with a ferromagnetic material

Yet, there are two primary problem about this solution. First of all, the casing of the IMU needs to be flat and perfectly enclosed—only leaving a gap for wires to enter. If the gap is comparable to the wavelength of the magnetic interference, the interference will still influence the reading of IMU. Another problem is that even if the IMU is perfectly enclosed by an iron case, it will block all magnetic field around which includes the geomagnetic field we need. It's hard to distinguish and filter out other magnetic interference if little knowledge about the interference source is known.

In fact, a substitute to the magnetic sensor is to use gyroscope as a temporary solution. For example, if a strong magnetic field other than geomagnetic field is detected, then we initialize the gyroscope with its last heading value and stick with the gyroscope estimation until no more magnetic interference is found.

#### **b. Improving the mechanical structure**

We would like to improve the mechanical structure of the car using more accurately 3D printed parts. One of the issue with the mechanical structure of the car is that its front wheels are not perfectly aligned. There is a slight angle between the two front wheels while they are supposed to be in parallel with each other. Thus, when the car was remotely controlled to run in a straight line, it was running in an arc that directs to the left. This issue posted obstacle in driving the car in a straight line, and in controlling the radius of the circle

during uniform circular motion. More consistent parts in the front assembly of the car and tighter connection between the Ackerman servo horn to the servo would improve the performance. Refining the 3D printing model of the parts or using a better 3D printer can achieve this goal.

**c. Dynamic offset**

As it is not practical to set an extremely precise offset value prior to each trial, a dynamic offset algorithm needs to be developed in order to reduce the propagation of errors. There are multiple references that can be used for updating the offset value as time lapses. In the case of the uniform circular motion experiment, the heading value can be used to determine if the car finished a loop. We can set a checkpoint, for example, at every time the car finished one loop of path. The average error in x and y accelerations resulted from running each loop could be experimentally determined though an experiment that contains a large number of loops. Then we can write the code so that this offset is applied to the x and y accelerations every time the car goes pass the checkpoint heading value. As the errors in accelerations can be limited using this scheme, the error in velocity and in turn displacement will significantly decrease assuming the offset value is appropriate.

In automobiles, location prediction of car solely based on IMU data is hardly used because of the error propagation. Usually automobiles rely on the GPS signal to determine its location. When GPS signals are not available, automobile would switch to using IMU until the signal is available again. Thus, having access to external information about acceleration, velocity or location can largely help improve the reliability of the algorithms.

**d. Sensor fusion**

Another approach to increase the accuracy of location determination of the car is using sensor fusion. As the precision of the IMU is limited, other sensors could be applied to provide additional information for acceleration, velocity or displacement. For example, an encoder can be added to the PiCar motor to measure the rotary position of the shaft. By converting rotary position into

analog and then digital signals, the encoder would enable us to calculate the angular displacement and in turn how many revolutions the motor shaft has been through. With this information and the radius of the wheels known, we would be able to calculate the displacement of the car. Angular velocity can also be determined using the encoder outputs if needed.

With additional information from encoder as a control, filtering algorithms can be applied to give an estimate of the current location based on the fusion of the two types of sensor data. For example, the Bayes filter provides an optimal estimation of probability of a state utilizing three pieces of information: previous estimate based on the calculated position upon last update, the measurement update that comes from IMU in our case, and the control update which comes from the data of the other sensor added. The equation of Bayes filter is shown below. When the variables are linear and normally distributed, the Bayes filter reduces to the Kalman filter. If another sensor is added to the system, applying filtering algorithms could optimize the estimation of current position of the car.

$$\underbrace{p(x_t | z_{1:t}, u_{1:t})}_{\text{New Estimate}} = \underbrace{\eta}_{\text{Measurement Update}} \underbrace{p(x_t | x_{t-1}, u_t)}_{\text{Control Update}} \underbrace{p(x_{t-1} | z_{1:t-1}, u_{t-1})}_{\text{Previous Estimate}} dx_{t-1}$$

Figure 29. Equation of Bayes filter.

## 10. Reference & useful links

- a) ESC calibration video  
<https://dronesandrovs.wordpress.com/2012/11/24/how-to-control-a-brushless-motor-esc-with-arduino/>
- b) Brushless motor  
<https://forum.arduino.cc/index.php?topic=270309.0>
- c) Free IMU-Update library  
<https://github.com/mjs513/FreeIMU-Updates/wiki/04.-FreeIMU-Calibration>
- d) SPI communication between Arduino & RaspberryPi  
<http://robotics.hobbizine.com/raspiduino.html>
- e) PySerial communication  
<https://pythonhosted.org/pyserial/pyserial.html#overview>
- f) Run a program on Raspberry Pi upon reboot  
<http://www.instructables.com/id/Raspberry-Pi-Launch-Python-script-on-startup/>
- g) Wireless interface default setting  
<https://www.raspberrypi.org/forums/viewtopic.php?t=134165>
- h) Change keyboard mapping on Raspberry Pi  
<https://raspberrypi.stackexchange.com/questions/40074/symbol-not-on-my-keyboard>
- i) Bayes filter  
<https://hackernoon.com/ghost-iv-sensor-fusion-encoders-imu-c099dd40a7b>