Amphibious Spherical Explorer

Final Report

By

Kaiwen Chen

Zhong Tan

Junhao Su

ECE 445 Senior Design, Spring 2016

Project No. 30

TA: Luke Wendt

May 5, 2016

Abstract

Spherical robots are robots that have a spherical shell with all the mechanical components enclosed by the shell. Due to the nature of round objects, spherical robots are immensely difficult to control. There are many tradeoffs involved. Some designs which use a car like structure allow spherical robots to accelerate and turn quickly while having uncontrollable oscillation.

Depending on the application, being able to control fully control all the movement of the robot is important. Therefore, any oscillation must be controlled to be within a certain amount.

The method used to control the oscillation is to use a pendulum based implementation. The pendulum is allowed to rotate with two degrees of freedom. Each degree of freedom correspond to control over a particular axis which the robot has the potential to oscillate.

Using this mechanical configuration, a controller is designed to use feedback to control and eliminate all the oscillation present in other designs. Comparing the results with robots built by Academic and Guardbot, the implementation used in this project has greatly surpassed the existing spherical robots in terms of oscillation control.

Contents

1 **INTRODUCTION** 1

2 **DESIGN** 1

2.1 Design Overview and Procedures 1

2.2 Circuit Design 2

2.3 Mechanical Design 4

2.4 Motor Driver and DC Motor 6

2.5 Servo 6

2.6 Magnetic Encoder 7

2.7 Inertial Measurement Unit (IMU) 7

2.8 WiFi Module 10

2.9 Control Panel 11

2.10 Control System 12

3 **REQUIREMENTS AND VERIFICATION** 14

3.1 WiFi 14

3.2 Control Panel 14

3.3 Motor Driver and Motor 14

3.4 Servo 14

3.5 IMU 15

3.6 Magnetic Encoder 15

3.7 Power Circuit 15

3.8 Accleration 15

3.9 Wobbliness 16

3.10 Turning Radius 17

4 **COST** 18

4.1 Labor 18

4.2 Parts 18

4.3 Grand Total 18

5 **CONCLUSIONS** 19

6 **REFERENCES** 20

7 **APPENDIX** 21

1 INTRODUCTION

Inspired by BB-8 in latest version of *Star Wars*, we built the amphibious spherical explorer (ASE), which is a spherical robot that can perform data collection tasks on land or in water. Due to the unique shape of sperical robots, they can almost travel across surface of different textures, such as water, mud, sand or even water. Also, this kind of robot is durable, since the sperical shell protects all the mechanical parts inside, and any impact to the shell will be uniformly distributed. However, people have to put heavy mass like lead ingot in the bottom of the robot to keep it stable, which increase the dead weight. Also, the dynamical system of spherical robot is almost undamped on smooth surface, which makes the movement of the robot always accompanied by wobbliness.

Objectives In this project, we have two fundamental objectives:

- Arrange the mass distribution properly in order to avoid using very heavy pendulum.
- Control the wobbliness at rest as well as movement.

2 DESIGN

2.1 Design Overview and Procedures

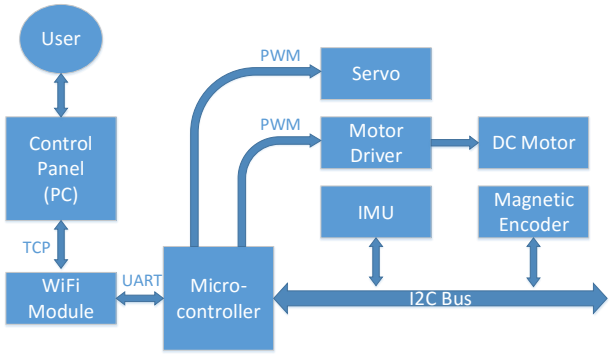

Figure 1. Block diagram of the ASE system.

The ASE robot works in a semi-autonomous mode, which means the user send commands to the robot via WiFi, based on these commands, the robot executes the user commands with the control strategies given by the internal control system. There are two sensors and two actuators in the robot: IMU to measure the attitude angles, magnetic encoder to measure the speed, DC motor to control the pitch angle and speed, servo to control the balance on roll angle. The microcontroller is the core of the system, all the control algorithm and software-hardware interfaces are programmed inside. The control panel is the human interface of the robot, it reads command line instructions or gamepad signal from the user, and format these into the commands readable for the robot, send them via WiFi. In the other direction, the control panel receive the data sent back from the robot, and visualize the feedback data for the user.

To design the whole system in the most efficient way, we divided the whole design in two phase. In phase one, mechnical design, circuit design and software design were carried out at the same time. In the second phase, the program structure, mechanical design and printings, PCB design and board were done, and we worked together to integrate the system and adjust the parameters in the control system.

2.2 Circuit Design

The Figure 2 is the schematic design our main PCB (Figure 3) implemented with eagle. The entire system is supplied with a 7.4V Li-PO battery, and in particular, the servo and motor driver will be supplied with 7.4V directly without any voltage regulation whereas the remaining circuitry will be powered by 3.3V which is output from the voltage regulators.

The main PCB is used to process and transmit data for the entire system. The three major components implanted on the main PCB are the microcontroller (Teensy 3.2), motor driver(DRV8835), and WIFI (ESP8266). The microcontroller communicates with WIFI through UART interface to send out commands and receive requested data. We use WIFI module to establish communication between user and the robot. The motor driver is controlled by the microcontroller with PWM signals. Power supply is located at the top left corner in the schematic, and the power supplied to servo and its control signal is located at the top right of the schematic.

Figure 4 shows the schematic of our sensor PCB (Figure 4) of the project, and the main purpose of this board is for positional data collection. The board is placed at the joint section of the shaft and motor as described in detail by mechanical part of our design. The board will also be powered by 3.3V which is supplied from the main board and meanwhile signal is sent through I2C protocol between microcontroller, IMU and magnetic encoder. The leftmost chip is the magnetic encoder (AS5048B), and the one right next to it is the IMU (MPU6050).

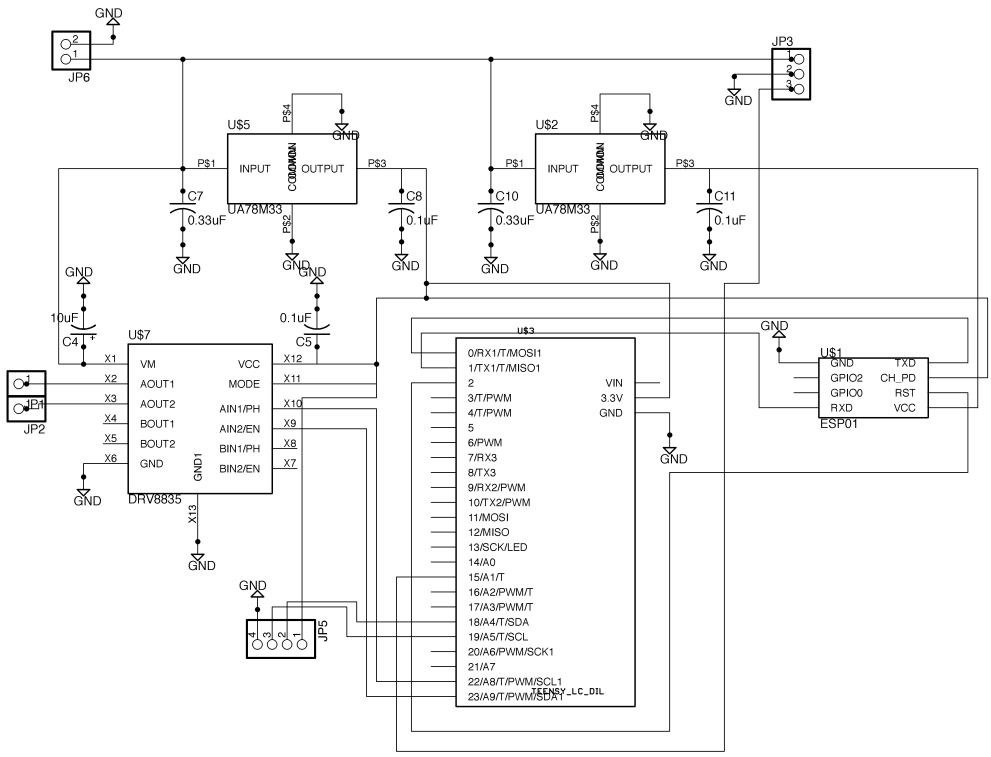

Figure 2. Schematic of the main board.

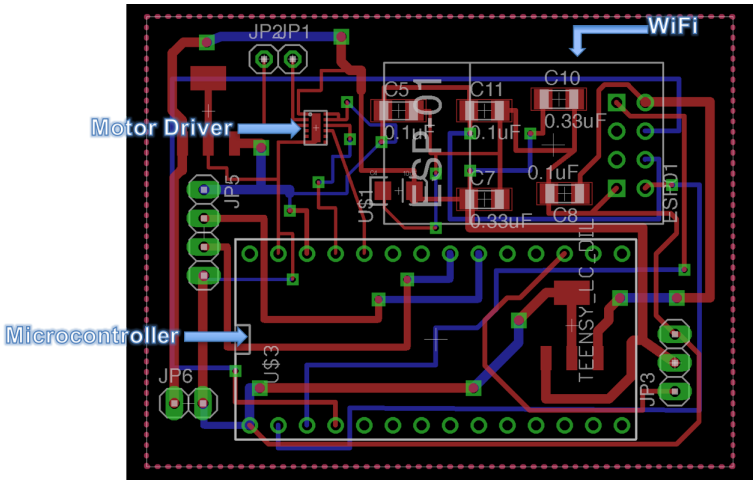

Figure 3. PCB of the main board.

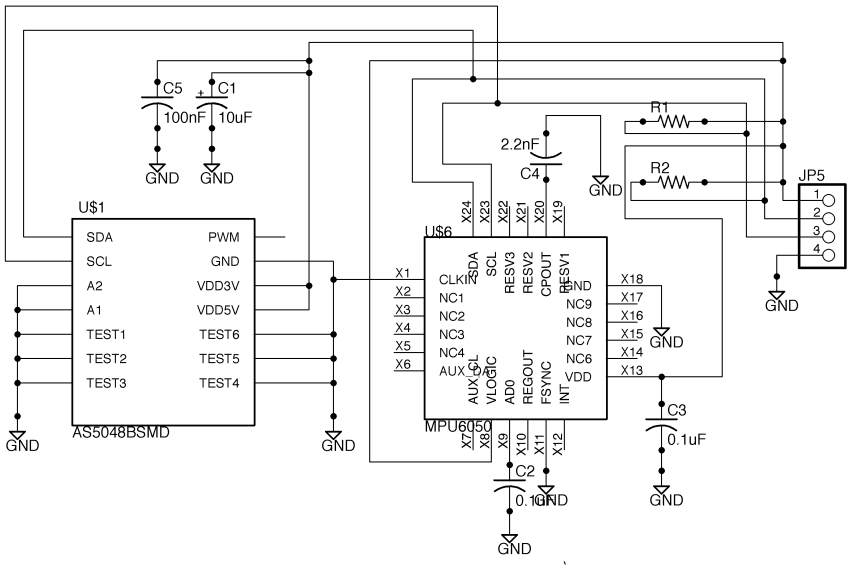

Figure 4. Schematic of the sensor board.

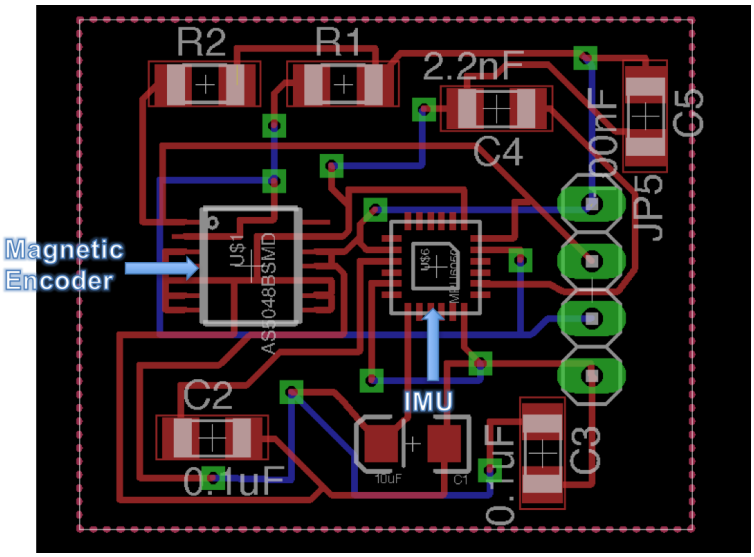

Figure 5. PCB of the sensor board.

2.3 Mechanical Design

The robot is designed with the consideration of having the best control in order to eliminate oscillations and wobbliness. Therefore, the idea behind the mechanical design was to use a pendulum that was allowed to swing in two different axes independently. The follow diagram is a simple illustration of the overall design.

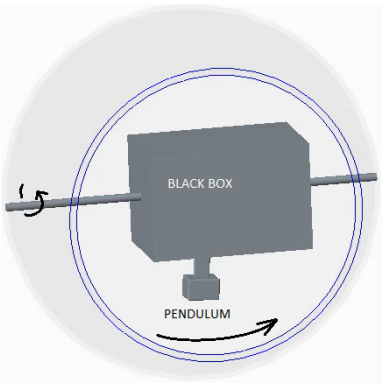

Figure 6. Concept model of ASE.

The pendulum is allowed two swing in both axes labeled by the black arrows. Once the pendulum is driven upward or side to side gradually, the ball rolls forward in response to the torque generated by gravity. The axle is fixed to the ball while the black box is allowed to rotate around it.

The mechanical design of this project evolved through four main stages. The first two stages were not realized due to the inefficiency of the design. The third design was realized. However, it had many physical constraints due to the asymmetry caused by the imperfections of handcrafting. Finally, the fourth design succeeded for being the most mechanically efficient. The following diagram illustrates the final design of the mechanical system.

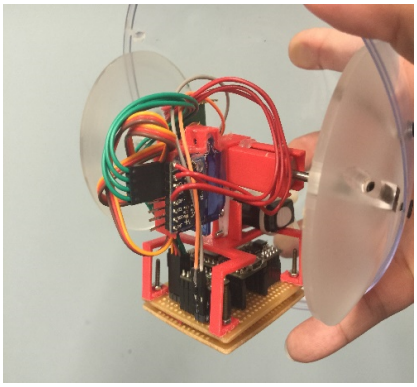

Figure 7. Internal sturcture of ASE.

The axle is broken into two disjoint parts and the motor itself is integrated with the axle. The direct output of the motor can be used to drive the ball forward. The electrical components are used as the weight of the pendulum which is attached to the servo. The axle is fixed to the inner surface the inner surface using the acrylic sphere segments on both sides of the shell. Credit goes to the ECE Machine Shop for designing the mount for the axle to the shell.

One significant benefit of this configuration is the placement of the magnetic encoder. Because the axle is broken into two parts, the magnetic encoder is placed between the gap of the two pieces. This allows for the direct angular velocity of the shell to be read, since the axle is fixed to the shell. Refer to the figure below for an illustration of the placement of the encoder.

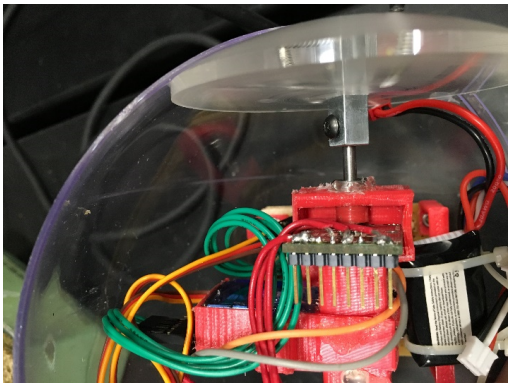

Figure 8. Disjoint axle for encoder placement.

In order to satisfy the speed requirement of 1m/s, the parameters of the motor must be chosen appropriately. The following worst case calculations are performed to determine the minimum torque and RPM required.

$$v_{max} \geq 1 \frac{meter}{sec} = 60 \frac{meters}{minute} \longrightarrow \text{137RPM required}$$

$$\tau_{max} = g * M_{pendulum} * R_{shell} \longrightarrow \text{0.34Nm required}$$

* R_{shell}= 0.7 $meters$
* C_{shell}= 0.44 $meters$
* $M_{pendulum}$ ~ 0.5kg

2.4 Motor Driver and DC Motor

The DC motor is controlled by the Pulse Width Modulation (PWM) signal output from the motor driver. The motor driver works in "Phase-Enable" mode [1], which means the "Phase" signal control the polarity of the output PWM signal, and drive the motor rotating in different direction, while "Enable" signal determines the output voltage is high or low, and control the motor to run or stop correspondingly. The duty cycle of the "Enable" PWM signal determine the effective voltage applied to the motor and therefore determined the speed of the motor.

2.5 Servo

The servo is controlled by an internal PID controller that is not accessible to users. However, the reference position is given from the input pulse witdh. A pulse with a width of 1.5 ms will let the servo point at the center. The 0.7 ms and 2.3 ms will bring the servo to the left maximum and right maximum angle, which is about $-90°$ and $90°$, respectively. The operating frequency (frequency at which the pulse is sent) is recommended to be 50 Hz [2]. In the experiment, it showed that the frequency can be raised up to 100 Hz, but the servo will become oscilating at certain angle in that case. When the operating frequency drops below 20 Hz, the servo will be no longer strong enough to hold the pendulum.

2.6 Magnetic Encoder

To apply speed control, we use a rotary magnetic encoder to measure the angle of motor rotation, and take the differentiation of the angle signal to get the angular speed and linear speed. However, we need to do two modifications to guarantee the measurement accuracy and precision.

First, we need to compensate for the "roll-over" effect. The output of magnetic encoder will experience a sudden change (360 to 0 or 0 to 360) at the boundary of each revolution, as shown in Figure 9. Thus, we need to shift the signal when a sudden drop or sudden rise is detected to make the angle signal continuous. Otherwise, there will be a huge pulse in the differentiation signal, which will make the speed controller collapse.

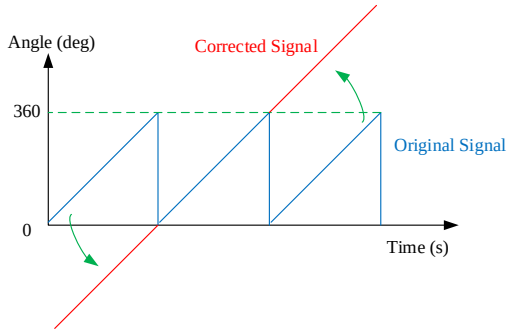

Figure 9. Roll-over effect and Roll-over compensation.

Second, the differentiation signal is noisy even after applying roll-over compensation. We still need an low-pass filter to eliminate the noise. Here, we design a first-order Butterworth filter and implement it in all-integrator circuit. [3] The continuous-time integrator is subsituted with a discrete-time accumulator. It calculates integral numerically using Euler method. The advantage of this design over traditional digital filter is we don't need to re-design the filter when the sampling frequency changes. We just need to adjust the time step of accumulator accordingly.

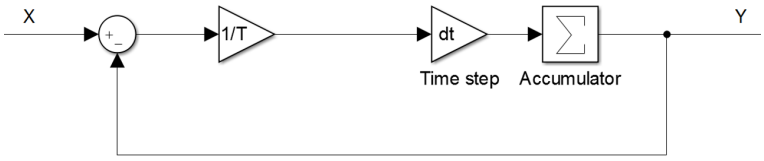

Figure 10. Implementation of first-order Butterworth filter.

2.7 Inertial Measurement Unit (IMU)

MPU-6050 is an raw IMU processor, which means if the user does not pay the manufacturer for an extra patented algorithm, the chip can only give raw data from the accelerometer and gyroscope without data fusion instead of the attitude angles. In this case, we designed an alternative algorithm called complementary filter for attitude measurement. [4]

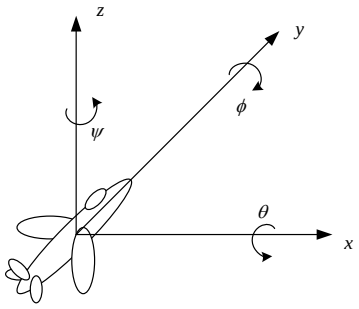

Figure 11. Definition of attitude angles.

First, define the orientations shown in Figure 11 (here we use a plane to represent ASE robot instead of a ball to show the orientation clearly): y → front, x → right, z → up. Then, the angle of rotation along each axis (related by Right-Hand-Rule, RHR) can be defined as: z → yaw angle ψ, x → pitch angle θ, y → roll angle ϕ, which is the yaw-pitch-roll convention of Euler angles.

If we want to transform an vector from the ground frame, we can use the following transformation matrices Eq.(1). [5]

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix} \quad R_y(\phi) = \begin{bmatrix} cos\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{bmatrix} \quad R_z(\psi) = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Left multiply them in the sequence of "yaw-pitch-roll", we can transform the gravity acceleration vector in ground coordinate to the body coordinate:

$$\begin{aligned} g_{body} &= R_y(\phi)R_x(\theta)R_z(\psi)g_{ground} \\ &= \begin{bmatrix} cos\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \\ &= g \begin{bmatrix} cos\theta sin\phi \\ -cos\theta \\ -cos\theta cos\phi \end{bmatrix} \end{aligned} \tag{2}$$

$$\begin{aligned} \Rightarrow \theta &= -asin\frac{g_{body,x}}{g} \\ \phi &= asin\frac{g_{body,x}}{gcos\theta} \end{aligned} \tag{3}$$

Eq.(3) gives the complete expression of attitude angle expressed by accelerometer data. However, since the controller does not require a absolutely accurate angle measurement for full range of attitude angles, we can

linearize Eq.(3) to reduce computational load, which gives Eq.(5)

$$g_{body} = g \begin{bmatrix} \hat{\phi} \\ \hat{\theta} \\ -1 \end{bmatrix} \tag{4}$$

$$\Rightarrow \begin{bmatrix} \hat{\theta} \\ \hat{\phi} \end{bmatrix} = \frac{1}{g} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} g_{body,x} \\ g_{body,y} \\ g_{body,z} \end{bmatrix} \tag{5}$$

Note that in practice, $\hat{\theta}$ and $\hat{\phi}$ are estimated by secant approximations of asin() function between $-30°$ and $30°$ (instead of tangent approximations at $0°$. This modification can keep the error of approximation consistently small in a larger range, though it will less accurate near origin compared to tangent approximation.

Accelerometer signal is accurate enough for static attitude measurement. However, when the robot is accelerating and decelerating, the inertial forces can change the gravity field "felt" by IMU, and makes the measurement inaccurate. The other senor in the IMU, gyroscope, also suffers from drift error when it using integration of angular velocity to estimate the angle. If we interpret this problem in frequency domain, the inertial force noise is high frequency disturbance and the drift error is considered as low frequency disturbance. Thus, the accelerometer is good at low frequency signal while the gyroscope is good at high frequency signal. If we design a complementary filter for each of them to let them switch their roles and make both of them work in the region they are good at, then the problem will be solved.

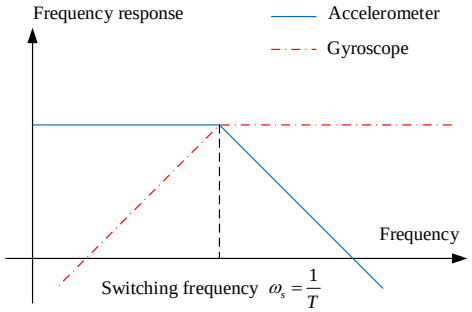

Figure 12. Frequency response of complementary filter.

Figure 12 shows the desired frequency response of the complementary filter in linearized Bode diagram. Let the switching frequency be f_s, the transfer function of the low-pass filter is $G(s) = \frac{1}{Ts+1}$, where $T = \frac{1}{w_s} = \frac{1}{2\pi f_s}$ is the switching time constant of the system. Since it is a "complementary" filter and we want an unitary gain, the high-pass filter should be $[1 - G(s)]$. Thus, we can derive the complete transfer function of

the complementary filter as Eq.(6).

$$\begin{aligned}\Theta &= [1-G(s)]\Theta_g + G(s)\Theta_a \\ &= (1-\frac{1}{Ts+1})\Theta_g + \frac{1}{Ts+1}\Theta_a \\ &= \frac{Ts}{Ts+1}\Theta_g + \frac{1}{Ts+1}\Theta_a \\ &= \frac{T}{Ts+1}(s\Theta_g) + \frac{1}{Ts+1}\Theta_a \end{aligned} \tag{6}$$

Implement Eq.(6) in all-integrator circuit, we can obtain the equivalent block diagram Figure 13.

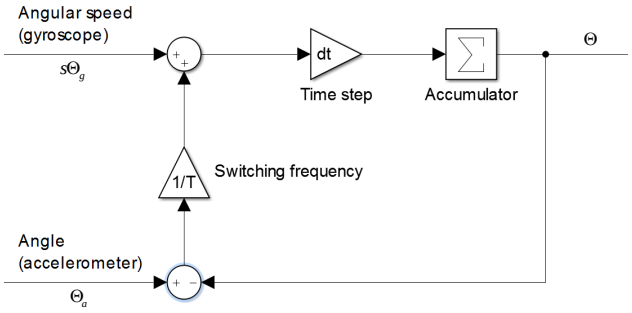

Figure 13. Implementation of complementary filter.

2.8 WiFi Module

ASE robot uses Espressif ESP8266 WiFi module to communicate with the control panel in the computer. it has a 32-bit SoC together with a WiFi chip to built an interface between the microcontroller and the WiFi chip. Thanks to the interface given by the manufacturer, the microcontroller of the robot can talk to WiFi module in so-called "AT command set" [6]. Every command in "AT" language goes as"AT+<command>=<parameter>", sent as ASCII strings via UART. After the WiFI SoC receives the string command, it decodes the string and translate the command into detailed instructions for WiFi chip.

To WiFi module generate an WiFi network and provides an access point for other clients to join. After the control panel connects the WiFi access point, the commands sent to the WiFi will be automatically writen to the microcontroller.

To send data back to control panel, we follow the procedures shown in Figure 14. Although TCP/IP protocol is robust and can guarantee the correctness of message sent, it requires a lot of acknowledgment procedures, which makes the constant cost for each transmission. Thus, we

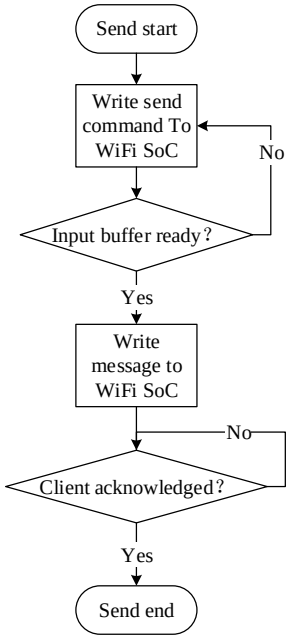

Figure 14. Flowchart of sending process.

should buffer the data first and send the sequence in a burst, instead of sending them packet after packet. This approach can significantly reduce the total time cost on acknowledgment.

2.9 Control Panel

The control panel is a MATLAB Graphic User Interface (GUI). There are two methods to write the command: use command line instructions or use gamepad interface.

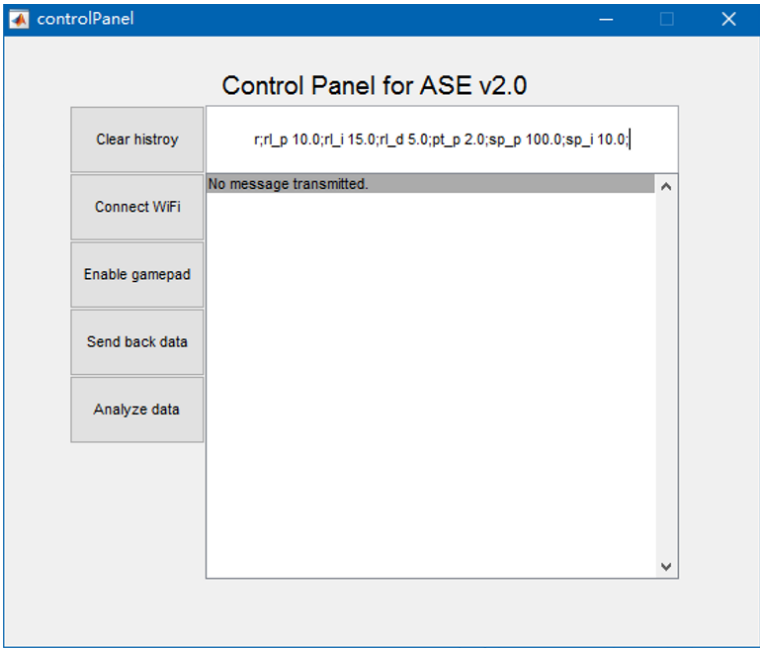

Figure 15. Control Panel for ASE system.

The command line instructions are very similar to C language. Each command end with ";" and multiply commands can be concatenated together. Two types of commands are used: the imperative command goes like "<command>;", without parameters, which will trigger a certain action directly; the assignment command goes like "<command> <value>;", which is used for variable assignment and will affect the actions that related to that variable indirectly.

The use can also enable the gamepad interface to translate the input from gamepad analog joystick to the command line instructions. Then, the command line instructions will be sent to the robot in the same way as the first method.

The control panel can also receive the feedback data from the robot. After receiving data, the control panel will read data from the buffer and re-format the data in a vector. This vector will be stored as an ".mat" file and the figure will be ploted to visualize the result.

Table 1. Command line instructions of ASE

Type	Command	Function
Imperative	w;	Increase motor power.
Imperative	s;	Decrease motor power.
Imperative	a;	Turn left with the preset servo actuation angle (negative value).
Imperative	d;	Turn right with the preset servo actuation angle (positive value).
Imperative	x;	Brake the motor.
Imperative	c;	Center the servo.
Imperative	r;	Re-calibrate IMU.
Imperative	f;	Send back data stored in the buffer.
Imperative	o;	Open-loop mode.
Imperative	l;	Closed-loop mode.
Imperative	$;	Restart everything.
Assignment	mt <parameter>;	Set motor power (always positive).
Assignment	sv <parameter>;	Set servo angle (always positive).
Assignment	sp <parameter>;	Set the speed reference, positive value for moving forward, negative value for backward.
Assignment	rl <parameter>;	Set the roll angle reference, positive value for turning right, negative value for left.
Assignment	sp_p(sp_i, sp_d) <parameter>;	Set the P(I, D) gain of the PID controller for the speed control.
Assignment	pt_p(pt_i, pt_d) <parameter>;	Set the P(I, D) gain of the PID controller for the pitch balance control.
Assignment	rl_p(rl_i, rl_d) <parameter>;	Set the P(I, D) gain of the PID controller for the roll balance control.
Assignment	rc <parameter>;	Record selected data in the buffer. (1 for speed, 2 for pitch angle, 3 for roll angle)

2.10 Control System

The control system consists of three controllers: roll controller, pitch controller and speed controller, which give output to two actuator: motor and servo. The controllers are mainly based on different variation of PID controller.

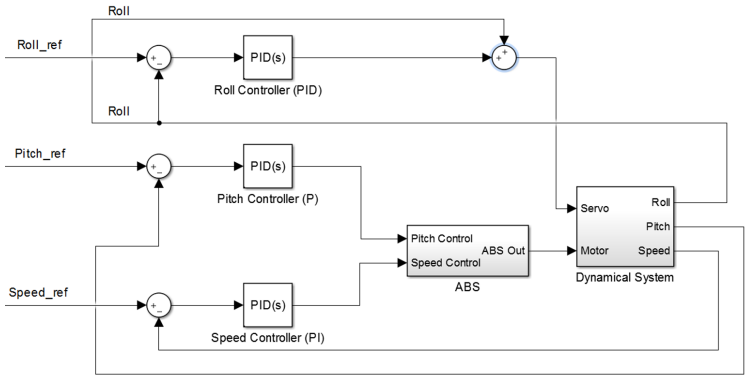

Figure 16. Control system of ASE.

Roll controller Roll controller uses PID control. The differenctial control (D) can significantly increases the damping ratio of the system and prevent oscillation. The integral control (I) is used to eliminate the error caused by mechanical asymmetry and enhance control effort when the ball is rolling and suffering from gyrosopic effect of the shell.

Pitch controller Pitch controller uses P control (proportional control) since pitch control is mainly used for keeping the pendulum from rotating over, and there is no other requirement on this channel. P control gives the simplest but effective solution.

Speed controller Speed controller uses PI control. The integral control (I) can eliminate the speed error from the speed reference as well as make acceleration faster.

Antilock Brake System (ABS) When the robot decelerate from a very high speed. The speed controller will be overloaded and make the pendulum rotate to the back for more than 90 degrees. In this case, the gravity torque given by the pendulum is actually decreasing as the pendulum keep rotating. To get the largest gravity torque to brake, we will cut-off the speed control to let the pendulum restore to horizontal position and then continue speed control again. This idea is very similar to the ABS system in modern automobiles.

Actuation angle compensation From Figure 17 we can see that due to the roll angle (ϕ) of the ball, the actual actuation actuation angle of the servo (α) is not the angle that gives gravity torque ($\alpha - \phi$, effective actuation angle). For a linear controller, the output of the roll controller means to control the effective actuation angle, instead of the actual actuation angle. To compensate this, we need to positively compensate the roll angle ϕ to the output of roll controller. After this compensation, the whole control system becomes Figure 16.

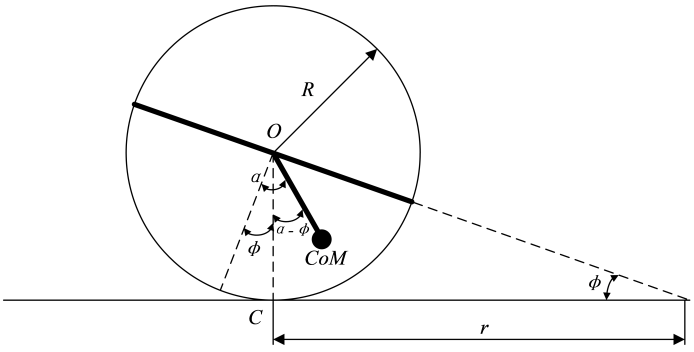

Figure 17. Effective actuation angle and actual actuation angle compensation.

3 REQUIREMENTS AND VERIFICATION

For more detailed requirements and verifications procedures, refer to APPENDIX part.

3.1 WiFi

The WiFi and the control panel (computer) is separated by about 20m. The WiFi receives 100 of the 100 packets sent by the control panel. It is ensured that no information is lost while the Spherical Explorer is operating within the specified range.

3.2 Control Panel

The time responsiveness of the control panel. A delay of no more than 2s is required for a robust control. Test commands are sent repeatedly and the robots responses to the signals are observed with respect to time. The robot responds instantaneously (much less than 2s) and this satisfies the timed response requirement of the control panel.

3.3 Motor Driver and Motor

The motor must be able to change speed and direction. Using varying positive PWM input, the motor responds by moving counter clockwise with speed proportional to the PWM duty cycle. By reversing the voltage of the PWM signal, the motor is driven in reverse at the speed corresponding to the duty cycle.

3.4 Servo

The appropriate servo must be chosen for the robot such that it rotate within the range of +45 degree to -45 degree without intensive oscillation. The servo was able to rotate over the full range required and there is negligible oscillation at steady state.

Table 2. Test result of servo.

Pulse length (μs)	700	1500	2300
Actuation angle (degree)	-92	0	97

3.5 IMU

Using protractor to draw lines on a paper from the same origin, from $-45°$ to $45°$ with $15°$ step size. Tape this paper on the wall, and let the edge of the MPU board coincide one line each time. Read the data measured and analyze the error. The result shows that the IMU test also satisfy the design requirement.

Table 3. Test result of IMU.

Actual Angle	$-45°$	$-30°$	$-15°$	0	$15°$	$30°$	$45°$
Measured Angle	-42	-30	-16	0	15	30	41
Relative Error	6.7 %	0 %	6.7 %	0 %	0 %	0 %	8.9 %

3.6 Magnetic Encoder

The magnetic encoder must be able to measure the speed of the robot in order to make proper control decisions. The actual constant speed across 20m is about 11 seconds. This corresponds to an actual steady state speed of 1.82m/s. Comparing this steady state speed with the logged steady speed shown below, the recorded data reached a steady state speed of 1.99m/s. The measured speed satisfies a conservative error of 50%. At the same time, this satisfies the maximum speed requirement of 1m/s.

3.7 Power Circuit

Components of the Spherical Explorer require a specific voltage. The output of the voltage regulator is measured using the oscilloscope. The oscilloscope outputs a voltage of 3.364V and this is within the range for proper functionality.

3.8 Accleration

The acceleration must not exceed a certain amount. Otherwise, the shell may not react fast enough to the torque and the pendulum may be driven over the axle. Using the speed, the acceleration overshoot is determined. Referring to the figure below, the overshoot of both positive and negative acceleration (brake) is within the specified value of 50%.

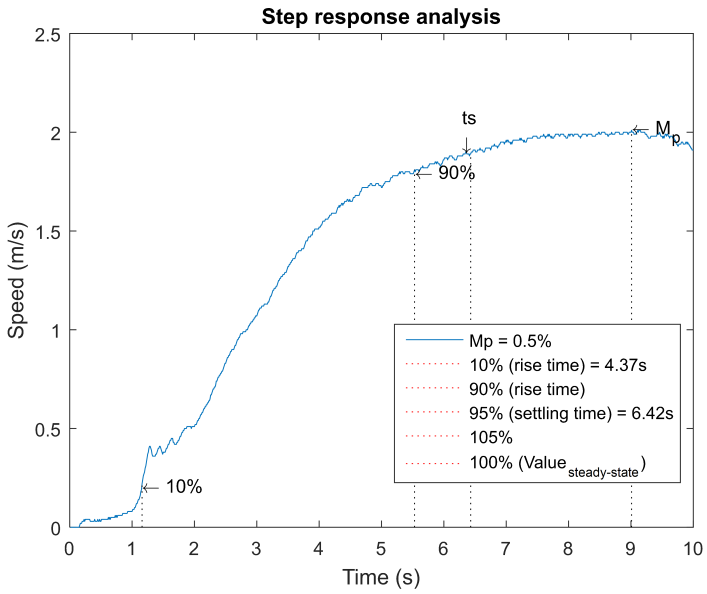

Figure 18. Acceleration Test.

3.9 Wobbliness

One of the main challenges addressed in this project is the oscillation control. To test this, the robot is given an offset of 30 degree and it is required to settle within 20 oscillations. A quick step response analysis of the control system shows that the robot settles to equilibrium within 2 oscillations. This satisfies the requirement by a great amount.

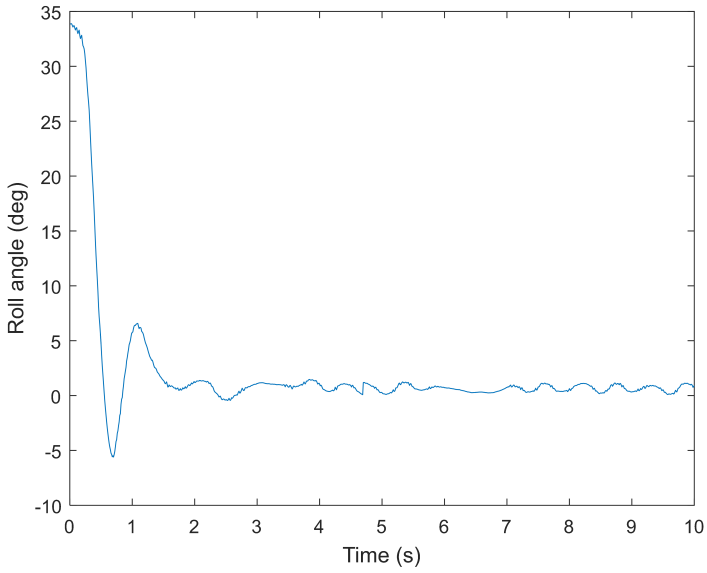

Figure 19. Restoration from wobbliness.

3.10 Turning Radius

Turning radius of our spherical explorer is considered to be a key factor of showing the versatility of it when controlled by the user. We set up the camera and recorded the robot circling motion from a high vantage point. Comparing it with the known preset radius of 2 meters, we found out the turning radius of the robot is much smaller than that. Therefore, the result proves the turning radius is within the requirement of 2 meters. The turning radius is dependent on the traveling speed of the robot, and therefore it can be better when turning at slower speed.

4 COST

4.1 Labor

$$\begin{aligned} \text{Labor Cost} &= \text{Hourly Rate} \times \text{Total Hours} \times \text{Number of People} \times 2.5 \\ &= \$25.00 \times 400 \times 3 \times 2.5 \\ &= \$75000 \end{aligned}$$

4.2 Parts

Table 4. Parts Costs

Part	Quantity	Cost($)
WiFi Module (ESP8266)	1 pc	6.65
Servo (Power HD 3688HB)	2 pcs	39.90
Motor Driver (DRV8835)	2 pcs	8.98
DC motor (Maxon A-max 22 diameter)	1 pc	20.00
IMU (MPU-6050)	1 pcs	6.84
Magnetic Encoder (AS5048B)	2 pcs	69.90
Microcontroller (Freescale MK20DX256 MCU)	1 pc	25.93
Camera (OV7670)	1 pcs	13.59
Plastic Ball	4 pcs	9.96
3D Print Filament	1 roll	18.00
Gears	2 bags	3.92
Li-Po Battery (Venom Fly 30C 2S 210mAh)	2 pcs	15.63
Total		239.30

4.3 Grand Total

$$\text{Grand Total} = \text{Labor Cost} + \text{Part Cost} = \$\ 75239.30$$

5 CONCLUSIONS

By the completion of the project, we were able to fulfill the goals we set to begin with. The first goal is to improve the control on wobbliness, and in order to do that we had successfully implemented a system of feedback controller to enhance the stability of the entire robot when in motion and is static. The second goal is to remove redundant masses and dead weight by increasing the mechanical efficiency.

For future works, we are considering adding real time camera for home security purpose. In addition, the controller can be programmed to learn the environment, for instance, traveling on swamp is different than traveling on a normal land, and to achieve consistency, the parameters of the controller will be tuned based on the data.

Possible ways to commercialize the product would be a smart fishing buoy or an area mapper. For example, the project can be modified to be water resistant and attached to a fishing line to travel to farther locations in the sea and relocated as desired. It can also be used to map an unknown place such as inside the pyramid or a cave using a camera or a location tracking device.

6 REFERENCES

[1] "DRV8835 Dual Low-Voltage H-Bridge IC," Datasheet, Texas Instruments Inc., 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/drv8835.pdf

[2] "SG90 9 g Micro Servo," Datasheet. [Online]. Available: http://www.micropik.com/PDF/SG90Servo.pdf

[3] R. Paz, "Analog Computing Technique," Lab notes for ECE 486, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, accessed December 2015.

[4] "The Balance Filter - A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform," Web page.

[5] M. V. Cook, *Flight Dynamics Principles*, 2nd ed., Burlington, MA: Elsevier, 2007.

[6] "ESP8266 AT Command Set," Datasheet. [Online]. Available: http://www.pridopia.co.uk/pi-doc/ESP8266ATCommandsSet.pdf

7 APPENDIX

Table 5. Table of Requirements and Verification

Module or overall performance	Requirements and Verification	Points
Control panel on PC	• R: Use command line to send commands to the robot and show all the commands are executed correspondingly.(5 pts). Use game controller to send commands and show all the commands are executed correspondingly (5 pts). • V: First send four imperative commands (roll left, roll right, accelerate, and decelerate), and two assignment commands (change motor PWM, and change servo position). Then, enable gamepad control in the control panel, check whether the robot can move accordingly.	10
WiFi	• R: Packet loss rate $< 10\%$ • V: To verify this, separate the robot and the computer by 20 ± 2 m. Then, ping the WiFi module in terminal with 20 packets. Check at the terminal for the number of packets sent and received. The definition of packet loss is $$\text{Packet loss rate} = 1 - \frac{\#\text{ received}}{\#\text{ transmitted}}$$	5
Microcontroller	• R: Interrupt cycles should be executed at frequency > 20 Hz, that is, each task schedule cycle (a list of tasks that run in peroidic cycles) has to cost < 0.05 s. • V: Modify the test program, let the microcontroller pull-up an idle GPIO at the beginning of each task schedule cycle, and pull-down the same GPIO at the end of that task schedule cycle. Using oscilloscope to measure the pulse width generated on that GPIO port, so that we can measure the time cost for each task schedule cycle.	5

Continued on next page

Table 5 – continued from previous page

Module or overall performance	Requirements and Verification	Points
Servo	• R: Give the servo proper PWM signal so that it can rotate within the range about $-45°$ to $+45°$, with tolerance $\pm 5°$ (use correct pulse length, 2 pts). The servo should also converge to the desired position smoothly without random intensive oscillation (use proper pulse frequency, 3 pts). • V: Change the program to open-loop mode. The microcontroller will output pulse with width of 1000 μs to 2000 μs, according to the input on the control panel. Use an oscilloscope to view the PWM and record pulse length as well as frequency. change the pulse width using command line, check if the servo can rotate over the full range required, and observe if there is intensive oscillation at steady state.	5
Motor driver and motor	• R: The motor should be able to change speed (5 pts) and direction(5 pts). • V: Use test program, which maps an analog input from 0 to 3.3 V, to a PWM output with duty cycle from 0/255 to 255/255, to generate PWM wave given to the motor driver. Check if the speed increases monotonically as the input voltage increases, and whether the direction of rotation reversed when the voltage level on the phase pin of motor driver is flipped.	10

Continued on next page

Table 5 – continued from previous page

Module or overall performance	Requirements and Verification	Points
Magnetic encoder	• R: Communicate with the microcontroller (4 pts). The angle for speed calculation should be compensated for the modulus effect, which means $360° + \alpha$ should be recorded when the angle goes to α in the next revolution, to avoid unexpected sign change in speed measurement (4 pts). The error of the speed measurement should be within 50% (2 pts). • V: To verify speed measurement, we first run the robot into steady speed. Then we measure time it takes to run 10 meters and at the same time, buffer the data inside the microcontroller and send the buffered data to computer via WiFi or serial port after the test. Take the averaged buffered data and compare with the averaged speed measured using stopwatch. If the result measured by magnetic encoder and result measured using stopwatch closely matches and the error is within 50 % , then the requirement is verified.	10
IMU	• R: Communicate with the microcontroller (5 pts). The angle measured by IMU should be within $1 \pm 20\%$ of the actual angle, within $-45°$ to $45°$. • V: Draw a collection of lines from $-45°$ to $45°$ with increment of $15°$. Tape this paper on the wall, and let the edge of the MPU board coincide one line each time. Read the data measured and analyze the error.	10

Continued on next page

Table 5 – continued from previous page

Module or overall performance	Requirements and Verification	Points
Power circuit	• R: Circuit take input voltage from 7.4V battery, output 3.3V ±10% (3 pts). Power circuit should be decoupled for power devices (motor and servo) (2 pts). • V: Use oscilloscope to measure if the average output voltage is correct. Run the motor and add different drag force to it, and check if there is visible pulses of amplitude more than 50% of the output voltage.	5
Moving Speed	• R: The average constant maximum speed when moving forward straightly should be $\geq$ 1 m/s. • V: First run the robot with full PWM duty cycle into steady speed. Then we measure time it takes to run 10 meters using stopwatch. Take the averaged speed across the 10 meters and check if the speed above 1 m/s.	10
Acceleration/brake	• R: The rising time of the speed step response (1 m/s) of the robot should be $\leq$ 15 s (4 pts). The time taken to stop the robot from the maximum speed should also be $\leq$ 15 s (4 pts). The overshoot of both process should be $\leq$ 50% (2 pts). • V: Send command to let the robot run to the maximum speed and buffer the speed data meanwhile. Log the speed data to control panel via WiFI or serial port after the test and measure the rising time and overshoot of the both step response using MATLAB. If there is too much noise in the speed measurement, apply piecewise cubic fit to the data and then measure the response specification.	10

Continued on next page

Table 5 – continued from previous page

Module or overall performance	Requirements and Verification	Points
Wobbliness	• R: Apply a initial angle of 30° to roll angle, the robot should be able to re-balance itself to ±15° from equilibrium roll angle in less than 20 oscillations. • V: Use the protractor to place the robot with a roll angle of 30°. Then, release the robot and let IMU measure roll angle. The data will be buffered during the test and sent back computer after the test via WiFi or serial port. Log the data to MATLAB and plot the data versus time to count the total oscillations before it goes into steady state (region within ±10°).	10
Turning radius	• R: The minimum turning radius should be $\leq$ 2 m. • V: Set the robot to rest and let it start to turn in one direction. After the turning motion reaches its steady state, put a reference object with known dimension near the turning circle and use a camera to take three pictures from the same position right above the estimated center of the circle (from a height directly above such as the ECEB second floor looking down to the lobby). Merge the three pictures into one and use the three positions of the robot to reconstruct the turning circle. Use the relative size of the reconstructed circle and the length of the ruler in the picture, we can calculate the real size of the turning circle.	10