

A SPHERICAL VISUALLY-GUIDED ROBOT

BIR BIKRAM DEY

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER
SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

JULY, 2020

© Bir Bikram Dey, 2020

Abstract

Spherical robots provide a number of advantages over their wheeled counterparts, but they also presents a number of challenges and complexities. Chief among these are issues related to locomotive strategies and sensor placement and processing given the rolling nature of the device. Here we describe Dragon Ball, a visually tele-operated spherical robot. The Dragon Ball utilizes a combination of a geared wheel to move the center of mass of the vehicle coupled with a torque wheel to change direction. Wide angled cameras mounted on the robot's horizontal axis provide a 360° view of the space around the robot and are used to simulate a traditional pan tilt zoom camera mounted on the vehicle for visual tele-operation. The resulting vehicle is well suited for deployment in contaminated environments for which vehicle remediation is a key operational requirement.

Acknowledgements

I would like to extend my deepest gratitude to Professor Michael Jenkin of the Department of Electrical Engineering and Computer Science at York University. He took the risk of buying expensive equipment for this project knowing all the risks. I would like to extend my sincere thanks to NSERC Canadian Field Robotics Network (NCFRN) for their support and help. I am also grateful to Robert Codd-Downey and Andrew Brown for all their help and support. I would like to acknowledge the support of my family.

Table of Contents

Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	x
Abbreviations	xvi
1 Introduction	1
1.1 Spherical robots: historical beginnings	4

1.2	Robot teleoperation	6
1.3	Visual teleoperation	8
1.4	Contributions	9
1.5	Structure of this thesis	9
2	Previous work	11
2.1	Historic spherical designs	12
2.2	Design principles	15
2.2.1	IDU modelling	17
2.2.2	Conservation of angular momentum	21
2.2.3	IDU design	22
2.2.4	House of Quality and IDU selection	30
2.3	Teleoperating spherical robots with ROS	31
2.4	Summary	31
3	Designing & building the Dragon Ball	33
3.1	Spherico	34
3.1.1	Spherico: shell design	37

3.1.2	Spherico: drive system	38
3.1.3	Spherico: dynamics	39
3.1.4	Driving Spherico	40
3.1.5	Lessons learned	41
3.1.6	Spherico: summary	42
3.2	DragonBall	43
3.2.1	Mechanical design	43
3.2.2	Outer shell manufacturing	44
3.2.3	Making the mold	53
3.2.4	Shell design	58
3.3	Drive system	62
3.3.1	Shock absorbers	65
3.4	Turning	65
3.5	Summary	67

4	Electronics system design of the DragonBall	69
4.1	Introduction	69
4.2	Electronics	70
4.2.1	System architecture	72
4.2.2	Power management	73
4.3	Sensors	76
4.3.1	Cameras	77
4.3.2	RTK GPS	78
4.3.3	Odometry	78
4.3.4	Torque sensor	79
4.3.5	IMU	79
4.3.6	Current and voltage	80
4.3.7	Temperature sensor	81
4.4	Motor selection	81
4.4.1	Drive motor	82
4.4.2	Turning motor	84

4.5	Battery safety	86
4.6	Communication	90
4.7	ROS	93
4.8	Summary	95
5	Control and visual teleoperation of the DragonBall	96
5.1	Introduction	96
5.2	Control	97
5.2.1	Drive system theory	98
5.2.2	Drive system implementation	99
5.2.3	Spot-Turning system	100
5.2.4	Spot-Turning implementation	102
5.3	Image processing	104
5.3.1	Image stabilization	105
5.3.2	Dewarping and stitching	107
5.4	Teleoperation	110
5.4.1	GUI	112

5.5	Example usage of the teleoperation system	113
5.5.1	Results and discussion	115
5.6	Summary	115
6	Summary and future work	117
6.1	Summary	117
6.2	Lesson learned	121
6.3	Future work	122
6.3.1	Shapes	122
6.3.2	Maneuvering	123
6.3.3	Image Stabilization	124
6.3.4	Autonomous Operation	125
6.4	Summary	125
	Bibliography	127

List of Figures

1.1	Large-scale disaster and a robot suitable for its investigation.	2
1.2	Spherical robot movement.	3
1.3	Amusement spherical toy design patent images.	4
1.4	Spherical Tank	6
1.5	Spherico and DragonBall	7
2.1	Toy design by J.L. Tate	12
2.2	Tumbleweed Robot	13
2.3	Spherical robots with different types of drive system.	14
2.4	Spherical robots with different types of drive system.	15

2.5	Spherical robot classification chart.	16
2.6	Free body diagram of forces acting on the robot	17
2.7	Free body diagram of forces acting on the robot	19
2.8	Comparison table of Spherical robots	22
2.9	Single and Hamster wheel robot.	23
2.10	Three robots with pendulum on rotating axis system drive	24
2.11	Gimbal mechanism and Single ball system robot	25
2.12	Three robots with omni wheel system drive.	26
2.13	Examples of a mass movement system robots	27
2.14	Example of a orthogonally mounted flywheels and a flywheel on pendulum system robots.	28
2.15	Example of a scissored-paired flywheel system robots from [1].	29
3.1	Spherico and DragonBall	34
3.2	Spherico: Three states of stability	35
3.3	Spherico: Drive system	36
3.4	Spherico: Detailed description	38

3.5	Spherico: C.G. movement	39
3.6	Spherico: Turning	40
3.7	Spherico: Oscillation after braking	41
3.8	Basic shell design	46
3.9	Mold design	47
3.10	Gore curve calculation	48
3.11	Steps in the fabrication of a gore curve.	49
3.12	Gap in Gore Curve cutouts	50
3.13	Graph of Equation 3.1 and 3.2	51
3.14	Carbon fiber laser cutout	52
3.15	Mold comparison.	53
3.16	Laser cut of the release ply.	54
3.17	Mold internals and ply placing.	56
3.18	Manufacturing process for the outer shell of DragonBall.	57
3.19	Overall assembly of DragonBall.	60
3.20	Jig design	61

3.21 Gasket placement.	62
3.22 Wheel holder design for Spherico	63
3.23 Force transfer beam design for DragonBall.	64
3.24 RC shock absorber.	65
3.25 Rollers in DragonBall for torque absorption.	66
3.26 Flywheel design.	67
4.1 Dragon Ball system architecture.	71
4.2 Sunburst graph of voltage, current and power.	75
4.3 Field of view for the Dragon Ball.	79
4.4 REACH EMLID GPS unit.	80
4.5 Optical sensors in the Dragon Ball	81
4.6 BNO055 IMU.	82
4.7 DragonBall voltage sensors.	84
4.8 DragonBall motors.	85
4.9 DragonBall batteries.	86
4.10 Torque motor current drain test.	87

4.11	LiPo voltage indicator and monitor.	88
4.12	Torque motor current drain test with filter.	89
4.13	User input comparison.	90
4.14	Communication architecture.	91
4.15	RC transmitter and receiver and the modified receiver.	92
4.16	ROS architecture of the Dragon Ball.	94
5.1	Effect of IDU angle to the drive direction.	97
5.2	Closed-loop control system for driving.	98
5.3	Cascaded PID control	99
5.4	Cascaded PID control vs Bang-Bang control.	100
5.5	Open-loop control system for turning.	101
5.6	Spot turning system.	102
5.7	Bar chart graph of Input target velocity to output orientation.	103
5.8	Camera positions in DragonBall.	104
5.9	Unstabilized vs Stabilized.	106
5.10	Camera views in DragonBall.	107

5.11	Example of blending dewarped left and right images.	108
5.12	Image processing pipeline for the DragonBall.	109
5.13	DragonBall possible turns.	110
5.14	DragonBall GUI	112
5.15	DragonBall GPS tab	113
5.16	Figure eight track	114
5.17	Teleoperating DragonBall	116
6.1	Super Ball Bot a Tensigirity robot.	122
6.2	Cubli Robot.	123
6.3	Sphericon	124

Abbreviations

6DOF Six Degrees of Freedom

DOF Degrees of Freedom

OTS Over the Shelf

FPS Frames per Second

GPS Global Positioning System

RTK GPS Real Time Kinematic Global Positioning System

IMU Inertial Measurement Unit

ROS Robot Operating System

SLAM Simultaneous Localization and Mapping

MDF Medium Density Fibreboard

CMG Control moment gyroscope

UAV Unmanned Aerial Vehicle

ESC Electronics Speed Controller

Chapter 1

Introduction

Unmanned vehicles are becoming increasingly popular for a range of tasks including mapping and exploration, especially in difficult to access or hazardous environments. That being said, deploying robots in hazardous environments can be challenging. For example, consider the task of deploying a robot such as the Clearpath Grizzly robot shown in Figure 1.1a in a contaminated environment such as the Fukushima power plant shown in Figure 1.1b. The vehicle must perform point-to-point navigation without the benefit of an effective pre-built map, it must interact with remote human operators, and critically, it must be possible to remediate the vehicle after operation. Given the complexity of the task domain there exist many interesting research problems associated with operating an autonomous vehicle in this type of environment. Consider the means of locomotion, for example. Many mobile robots are based on a locomotion system designed for humans such as the car (ground contact robot), helicopter (quad copter), aircraft (unmanned aerial vehicle) or boat (unmanned

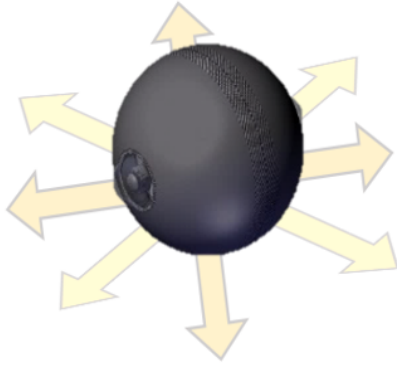


(a) Clearpath Grizzly Robotic Utility Vehicle. (b) Dai-ichi Nuclear Power Plant Accidents. Image reprinted from Fukushima Power Plant/Crisis/blogspot.

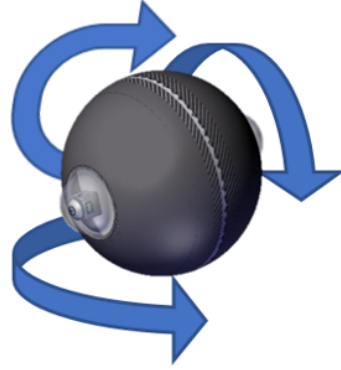
Figure 1.1: Large-scale disaster and a robot suitable for its investigation. (a) The Clearpath Grizzly Robot which is well suited for large-scale environment investigation. (b) The Dai-ichi Nuclear Power plant. An environment that has been investigated using autonomous systems and for which vehicle remediation post-deployment is key.

surface vehicle). Each locomotive strategy introduces its own constraints and capabilities in terms of the mobility of the device. Boats work well on smooth water surfaces, cars on paved flat surfaces, etc. A key operational issue for many of these devices when they are deployed for real world tasks is that they are not well sealed with respect to the environment. This introduces a number of complexities and complications when the device is deployed. Quadcopters, for example, are typically not easily deployed in the rain nor are they easily remediated after deployment. A key issue in terms of remediation is that penetration into the vehicle body itself is a potential vector for environmental contaminants to enter into the vehicle. Sealing shafts, sensors and the like from external environment contamination is difficult or impossible for many “off the shelf” autonomous robots.

One vehicle design that avoids many of the issues associated with environmental contamination while providing good mobility in complex environments is the spherical robot. A sketch of this design is shown in Figure 1.2. There is no need for penetration of the



(a) Lateral movement of a spherical robot.

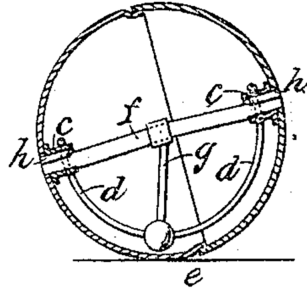


(b) Rolling movement of a spherical robot.

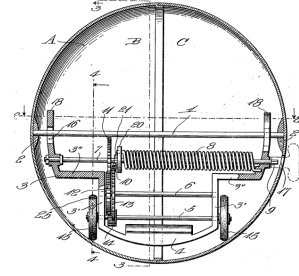
Figure 1.2: Spherical robot movement. Depending on its drive mechanism a Spherical robot (a) can move in any direction at any given moment. Similarly given an appropriate drive mechanism it can roll to change its orientation (b) at any given moment.

robot's shell to enable the robot to move. The entire robot is enclosed and can be sealed from the outside environment. In this design cleaning and decontamination after deployment is needed only on the outside surface. Also due to its shape, a spherical robot can be deployed in complex spaces and environments that are not possible for many wheeled or tracked vehicles. For example a spherical robot can be deployed in pipes and narrow corners easily, which can be a critical requirement for surveillance of hazardous environments. Such maneuverability enables a spherical robot to perform tasks that may not be suitable for its wheeled or tracked alternatives. Another key advantage of a spherical robot design is that a spherical robot “cannot be overturned” [2]. This can be a great advantage for autonomous robots operating in uncertain and complex environments. For many autonomous robots, if it were to end up upside-down, then it simply would not have the means to flip itself self back over, unless some other system was attached specifically to handle the situation.

Spherical robots present a number of interesting challenges in robotic control systems design [3], sensor placement and sensor data analysis. Furthermore, such devices have typically not been used as the underlying hardware for research into their usability for typical



(a) Spherical toy ball design by B.Shorthouse.



(b) Hamster Ball design by A.D.McFaul.

Figure 1.3: Amusement spherical toy design patent images. Images reprinted from [4] and [5].

autonomous tasks such as visual teleoperation. By design, a spherical robot can be extremely maneuverable compared to platform options such as a four-wheel drive vehicle or a tracked vehicle. The design does, however, introduce a range of complications in terms of sensing and control. Where should sensors be located? How best should teleoperational control be implemented? An operator is unlikely to be interested in seeing a rotating world, for example.

This thesis addresses two issues related to spherical robots:

- The design, development and demonstration of a visually-guided spherical robot for operation in hazardous environments.
- The development of appropriate visual teleoperational techniques for a spherical robot.

1.1 Spherical robots: historical beginnings

The first patented mechanical toy design of a spherical ball propelled by an internal spring appeared in 1893 by John L. Tate [6]. In his design a spherical hollow toy was equipped

with an axial rod from one end to other, and mass with spring attached would swing on that rod. When the toy was rolled across the floor, this would cause the internal spring to wind internally and make the mass swing in the opposite direction. The torque created by the swing of the internal mass would cause the ball to roll in order to conserve angular momentum. Following this design other designs for toys were developed, including one by B. Shorthouse. He improved the design such that it could follow a predefined path. This design is shown in Figure 1.3a [4]. The next notable design is attributed to A. D. McFaul. He patented his “hamster-ball” design in 1918 [5] where the mass would move up using friction between the inside surface of the ball and wheels attached to the mass, as shown in Figure 1.3b. One advantage of this design was that it requires less torque to move. This design was adopted by many researchers in the design of spherical robots. Spherical designs are not only for toys and robots. For example the idea of a spherical mobile platform went as far as the design of a mobile tank that appeared on the cover of the Popular Science in the July 1936 issue (Figure 1.4). As is discussed in the next chapter there are many different spherical robot design strategies.

In my thesis I constructed two different spherical robots. Figure 1.5 shows these two robot designs. The first design (Figure 1.5a) was a prototype, and there was no sealed external shell on this robot. For the second design, a carbon fiber shell was constructed to seal and enclose robot and provide the necessary external rigidity for the robot. Full details of these robots are given in chapters 3 and 4.

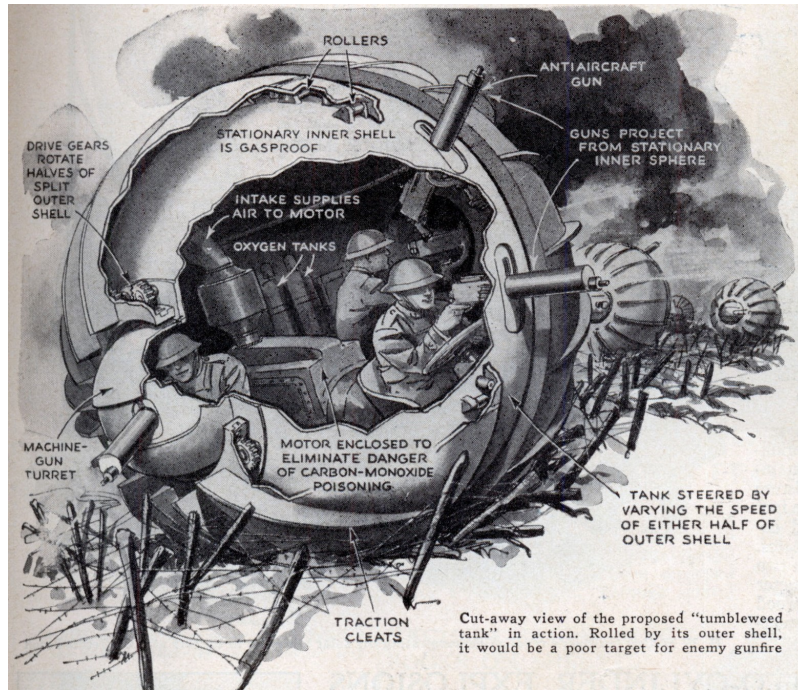
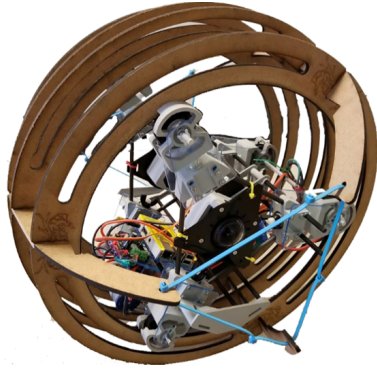


Figure 1.4: Spherical Tank Image reprinted from Popular Science, July, 1936.

1.2 Robot teleoperation

The ability to control a device from a remote location opens up a large range of applications. The process of providing remote control or teleoperation has application from the remote control of a television to controlling drones by a pilot in an air force. Radio controlled devices have been around for a long time, and the integration of teleoperational control in robotics application have opened up applications of robotic devices in various fields. The list of application of teleoperation of robots is large, but perhaps the most significant of these is the remote control of the Mars rover from the earth [7].

Teleoperation is commonly considered to be direct device or actuator control, but often this is not the case. Teleoperation can be as low level as controlling the power of a motor, or it can be as high level as sending commands for the robot to change some high level device



(a) Spherico.



(b) DragonBall.

Figure 1.5: The prototype (Spherico) and the final robot (DragonBall). The prototype was built to answer a range of fundamental and practical questions related to the construction of a spherical robot.

state. It can even be a combination of commands at different control levels. Controls such as data logging may be done via direct control from the user, whereas commands such as the direct control of individual motors are commonly not performed via direct control. With better telemetry and system design a remote operator may have a good estimate of the state of his robot, yet it may still not be ideal to have the operator take direct control due to lag in information transfer, human response time, etc. It may also be difficult for a user to incorporate separate system properties and capabilities when using direct teleoperation. For example, if the battery power is low and yet the user would like to drive the motors at high speed, the robot internally might adjust the user's request given current state of the robot.

Teleoperation of a device can be divided into three basic approaches [8]; *closed loop control*, *coordinated teleoperation* and *supervisory control*. Under closed loop control the user controls the motors directly, such as in a remote control aircraft. Under coordinated teleoperation there is an internal control algorithm to perform tasks based on the user's inputs. Under supervisory control, the user only sends high level commands that the robot performs

autonomously. For a spherical robot coordinated controls is perhaps most appropriate. Direct teleoperation of a spherical robot can be hard [9]. Understanding sensor data directly from a spherical robot may be difficult as well. Unlike a traditional wheeled robot a spherical robot will continuously change the orientation of on board sensors with motion of the robot. This certainly complicates the teleoperation of such a robot. This thesis demonstrates a gravity-frame stable frame for visual teleoperation that simplifies visual teleoperation of this type of vehicle.

1.3 Visual teleoperation

Controlling a teleoperated robot requires a good human-robot interface design. Proper perception of the robot's environment is critical for the user to control the robot properly. A good user interface will help to ensure that the intended control commands are executed properly. Typical challenges for teleoperation include communication delay, sensor integration issues, and limited data bandwidth. Many of these challenges are a function of sensor choice, placement and integration, type of robot platform used and intended tasks. Since there is a constraint on the number of sensors that can be installed and the available computational budget, pragmatic yet effective choices must be made. The choice of robot platform plays an important part in sensor integration. An aerial vehicle requires a different set of sensors than an underwater vehicle. The intended applications of the robot plays a role. An unmanned ground vehicle (UGV) equipped with GPS and a camera can be operated fairly easily on land, whereas if the intended application is on another planet then GPS may not be available and the problem is more difficult. Another factor is data bandwidth. While a Wi-Fi connection is very easily integrable in a UGV it is not suitable for an underwater robot as wireless communication doesn't work well underwater. Available alternates such

as acoustic modems have significantly lower data bandwidth and higher latency. In such situations transmitted data has to be carefully selected even if that means substantial data processing the data before transmission. Given its high bandwidth, video transmission is typically an issue. Processing the video to reduce its data footprint, e.g. by only transmitting a region of interest (ROI) and reducing image resolution are options here. Latency in command data transmission is another challenge. Inputs that provide control commands in real time may not be interpreted properly due to delays in data transmission. Effective teleoperation requires that proper vehicle operation continues in the presence of these delays.

1.4 Contributions

The key goal of my thesis is to develop a spherical robot that can be visually-guided for operation in hazardous environments. Spherical robots have the ability to maneuver holonomically, which makes this robot perform better in many situations. But it also makes it difficult to implement a conventional teleoperation system. Unlike conventional wheeled vehicles the internal structure of the robot rotates completely, which makes it hard to assign a stable reference frame for robots operations. Specific contributions of this work include:

- Design and implementation of a sealed, visually teleoperated spherical robot.
- Development of an effective visual teleoperation system for the robot.

1.5 Structure of this thesis

There are six chapter in this paper. A summary of each chapter is given below.

1. Chapter 1: Introduction. Introduction to this thesis, explains the overall idea behind this research.
2. Chapter 2: Previous work. This chapter reviews the works done previously on spherical robots. This chapter also explains how to categorize previous works.
3. Chapter 3: Building the DragonBall. This chapter describes the physical structure of the spherical robot developed here. Thus includes the mechanical, electronics, and sensor subsystems of the robot.
4. Chapter 4: Low-level and teleoperation control of the DragonBall. This chapter describes the visual teleoperation with the vehicle, includes the process of providing a visual state display for the operator.
5. Chapter 5: Visual teleoperation of DragonBall. This chapter describes how visual teleoperation was designed for this vehicle, including a method of optimizing visual data transfer over WiFi network.
6. Chapter 6: Summary and future work.

Chapter 2

Previous work

The invention of the wheel was a gigantic leap for mankind. Wheels enable a wide range of vehicle design possibilities in terms of mobility. They have been used in devices from Egyptian chariots to Mars rovers. The wheel is an ideal design for working on a two dimensional plane. But wheels are not perfect in a 3D world. While a wheel is an ideal device for motion along a line, it requires other mechanisms to enable the mechanical structure to maneuver on a two dimensional plane. The sphere, on the other hand is designed to move on a surface. Given an appropriate control mechanism it can roll in any direction. Just as the circle is the ideal design for the 2D plane, the sphere is the ideal mechanical device for motion on a plane. Given this, it is thus perhaps not surprising, that there are number of benefits to building spherical robots.

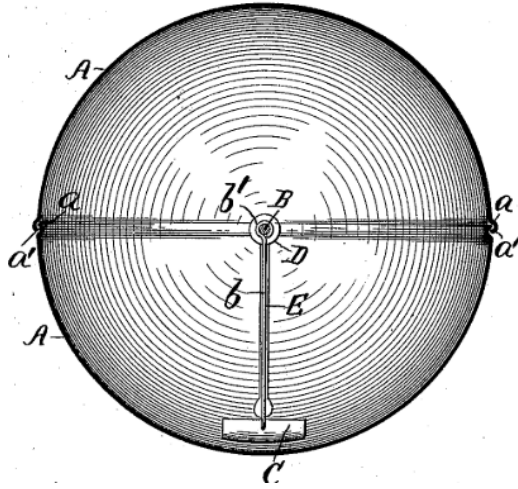


Figure 2.1: Toy design by J.L. Tate in 1893. Image reprinted from [6].

2.1 Historic spherical designs

Spherical vehicle designs can be traced back to 1893. Early toy devices include the design shown earlier in Figures 1.3 and in Figure 2.1 in this Chapter. Before exploring the design choices for a spherical robot formally, it is first worth noting that the design space for spherical robot is very large. In order to highlight this a few examples are provided in this chapter to illustrate this first before providing a systematic review of the design space.

Perhaps the most unusual spherical robot design would be the Tumbleweed robot [10], which was evaluated for planetary exploration (Figure 2.2). The basic idea here is to use a large inflatable ball that would be windblown by the planet's atmosphere. The necessary payload would be protected inside the inflatable robot. It is a concept design for planetary exploration, which if deployed, would be very energy efficient. The initial idea came after noticing that the "Mars Pathfinder Lander travelled farther bouncing on its airbag than the Sojourner rover travelled" [11]. One of the main contributions of this design is its capability of traversing long distances without the need for power. This makes the Tumbleweed design

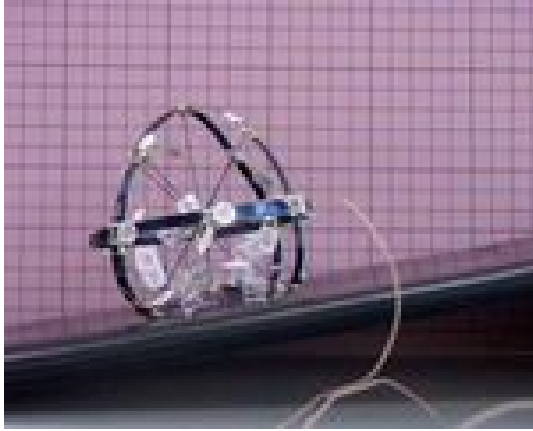


Figure 2.2: Tumbleweed Robot. Image reprinted from NASA JPL/Tumbleweed.

an appealing design for planetary exploration. This concept design has been improved by several organizations including the Swiss Federal Institute of Technology and Texas Technical University. Other spherical robots such as the Thistle robot [12] also harness the power of wind to propel themselves.

Another interesting spherical robot design is the robot Koharo from Ritsumeikan University in Japan, shown in Figure 2.3a. Koharo uses a deformable shell made of shape memory alloy (SMA) actuators [13]. The SMA actuators are used to deform the spherical shaped shell of the robot. With proper control of the external shape of the robot it can be deformed such that it can crawl and jump. The system described in [13] was a tethered prototype to evaluate feasibility of such design. While this prototype was not application specific, the simulation data obtained from the research shows fascinating capabilities and the range of design possibilities for spherical robots.

The Rollo is a spherical robot with a transparent acrylic cover [12], developed at the Automation Technology Laboratory of Helsinki University of Technology (Figure 2.3b). There have been three design iterations of this robot. The robot has two degrees of freedom, and has all of its sensors and controllers located inside the shell. The acrylic shell is connected to



(a) Deformable spherical robot. Image reprinted from Ritsumeikan University/Koharo.



(b) Rollo spherical robot. Image reprinted from Ritsumeikan University/Rollo.

Figure 2.3: Spherical robots with different types of drive system.

a rim and the inner masses pivot from this rim. The rim can rotate with the shell to make the robot move forward, and to move side to side the inner mass rotates with respect to the rim. In order for the inner mass to rotate the inner mass must be in a horizontal orientation. This makes the vehicle quite difficult to control. This robot design was proposed for various applications including as an interactive telepresence robot and for entertainment [14].

Spherobot, developed around 1999 at the Michigan State University has a propulsion system different from other spherical robots, as shown in Figure 2.4a. Spherobot has an external shell with spokes fixed radially inside, on which weights can move [15]. The distribution of the weights on these spokes determines the center of gravity of the robot. Changing the distribution enables the robot to “accelerate, move with constant velocity or servo at a point” [15]. Other than the propulsion system of the robot, it has other unique features, such as a retractable camera and telescopic limbs. These features are only for use when the robot is not moving. Later versions of this device were made with fewer internal actuated weights [16].

Another novel way of controlling and propelling spherically robots is by exploiting con-



(a) Spherobot. Image reprinted from Michigan State University / Spherobot.



(b) V. Joshi's spherical robot with two motors. Image reprinted from [17].

Figure 2.4: Spherical robots with different types of drive system.

servation of angular momentum. V. Joshi [18] developed a spherical robot with two electric motors mounted inside the shell. The motors are paired diametrically opposite to each other, making two pairs on two axes. Modelling is complicated compared to other designs, yet in [18] a simple path planning algorithm was presented to control the robot.

2.2 Design principles

The previous section illustrates the range of possibilities in spherical robot design. Although there is a wide number of different designs a number of “standard” splendid spherical robot design have emerged. It is typical to characterize spherical robot designs by their driving mechanism. The nomenclature described in [16] extends this idea by further dividing robot designs by other attributes. Here the categorization is extended further to capture more of the design space, as shown in Figure 2.5. Spherical robots can be divided into two main design classes, active and passive locomotive designs, as done in [16]. Examples of passive spherical robots include those which are powered by an external source such as wind,

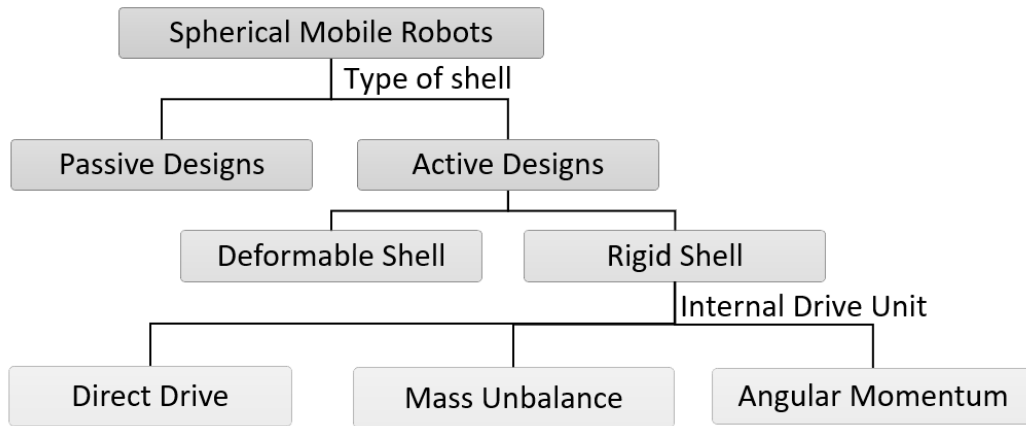


Figure 2.5: Spherical robot classification chart.

including the Thistle [12] and Mars Ball [19] platforms. Active designs can be further divided into deformable and rigid shell designs. Deformable spherical robots are designs which can morph into different shapes to move forward such as the robot described in [13]. Rigid shell design are perhaps, the most common. Rigid shell spherical robots are those whose shell provides an external spherical shape to support internal parts and remains in that shape. Examples of such robots includes the Rollo [12] and the spherical robot described in [18].

Although the spherical robot shape has various advantages as a robotic platform it also has disadvantages. Disadvantages include problems involved in rolling up or down an inclined plane, as is discussed later in this chapter. Spherical robots that can deform clearly have an advantage from others due to properties of their deformation process and the advantages associated with their deformed shape. But there are also drawbacks associated with a deforming shell. For example, unlike a rigid shell a deformable shell is not easily sealed and thus has pockets within which debris can hide.

For a typical rigid shell spherical robot, the outer shell rotates using the torque generate by the internal motion system of the robot. Various mechanisms of internal driving have been developed over the years. The challenge is to design a drive system that can generate output

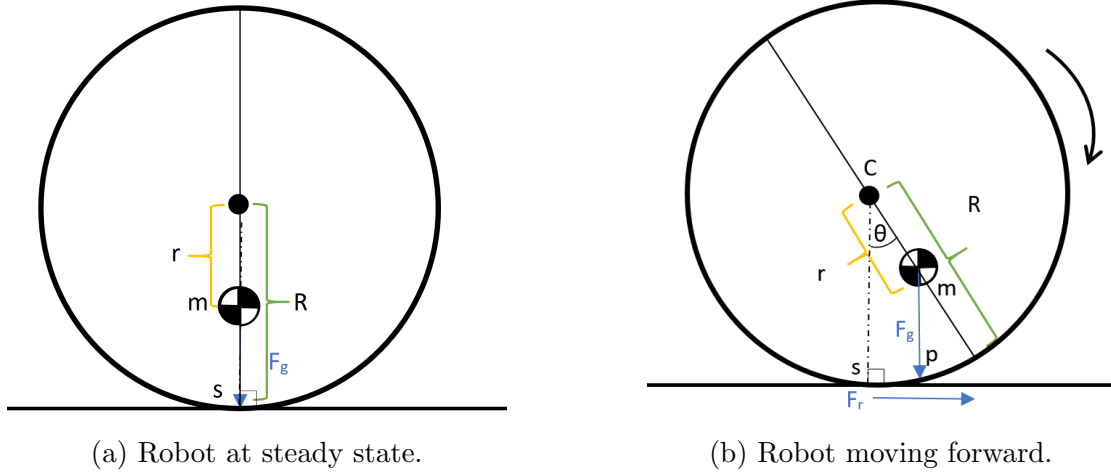


Figure 2.6: Free body diagram of forces acting on the robot

torque to drive the outer shell. Most rigid shell spherical robots rely on either conservation of angular momentum, direct drive, or mass unbalance to drive the robot. The drive system, or “Inside Drive Unit” [2] or IDU, of a spherical robot can be divided into three different classes, as shown in the bottom layer of the classification chart given in Figure 2.5.

2.2.1 IDU modelling

Before exploring further how the IDU's of spherical robots are classified, it is worthwhile to first look into a simplified dynamic model of a rigid shell spherical robot. The IDU is different in different robots which makes it hard to generalize the dynamics of the system. However, we can start by analyzing a simple 2D wheel-based system. From Figure 2.6b the following equations can be derived;

$$\omega = \frac{d(\theta)}{dt}. \quad (2.1)$$

$$v = R\omega. \quad (2.2)$$

Where θ is the angular velocity of the robot, t is the time and v is the forward velocity of the robot.

At steady state, as shown in Figure 2.6a, the center of the sphere and the center of gravity (C.G.) fall on a vertical line. The force of gravity, F_g , passes through the point of contact at point s . F_g is balanced by the normal force from the ground. Since all the forces are at balance the robot stays stable. In order to move forward the motor inside the robot creates a torque at the center to move the C.G. up in the counter clockwise direction. This exerts a clockwise counter torque in the body of the robot, which makes the robot move forward (to the right). The total τ_{total} torque can be calculated as;

$$\tau_{total} = I_{shell}\alpha - mgr\sin\theta. \quad (2.3)$$

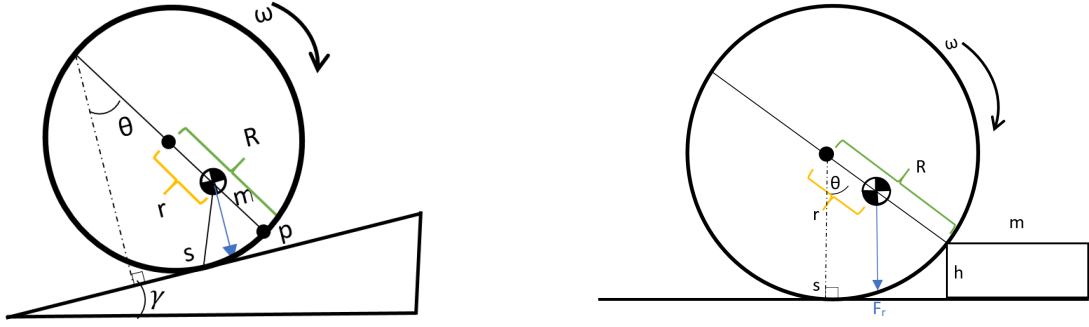
Here I_{shell} is the moment of inertia of the robot and α is the angular acceleration of the shell. At constant acceleration:

$$\tau_{total} = 0. \quad (2.4)$$

Therefore, at constant acceleration:

$$I\alpha = mgr\sin\theta. \quad (2.5)$$

If the angular moment of inertia I is considered to be $\frac{1}{2}MR^2$ (the mass distribution is uniform over a disk of radius R), then at constant acceleration:



(a) Free-body diagram of robot going uphill. (b) Free-body diagram of robot going up an obstacle.

Figure 2.7: Free body diagram of forces acting on the robot

$$\frac{1}{2}MR^2\alpha = mgr\sin\theta. \quad (2.6)$$

Therefore:

$$\alpha = \frac{2mgr\sin\theta}{MR^2}. \quad (2.7)$$

The key points to note here are:

1. The larger the value of r , the higher the α value will be. Note $r \leq R$.
2. The maximum α occurs when $|\theta| = 90^\circ$.
3. The mass of the robot should be as small as possible, so that the angular acceleration of shell is high.

It is important to understand the theoretical limitations of the robot on different terrains. Uphill motion and going over an obstacle are two common analyses that are performed on spherical robot designs. The limitation in uphill motion can be easily modeled as shown in Figure 2.7 and as described below.

$$\tau = l \times m = m(r \sin \theta - R \sin \gamma). \quad (2.8)$$

Since $l \times m \geq 0$

Then $(r \sin \theta - R \sin \gamma) \geq 0$, Therefore:

$$\gamma_{MAX} = \sin^{-1}\left(\frac{r}{R}\right). \quad (2.9)$$

The critical observation here is that the mass of the robot, or the size of the robot is not critical. Rather the ratio between the radius of the robot and the internal pendulum length is critical. The closer the two values are to each other the higher robot will be able to climb. The other analysis commonly done is climbing up a steep obstacle. Only the result for the maximum step (h_{max}) is given below. The derivation can be found in [20] and [21].

$$h_{max} = R - \sqrt{R^2 - \left(\frac{rm}{M+m}\right)^2}. \quad (2.10)$$

An obvious observation here is that the larger the value of R the larger the value of h_{max} . Also it should be noted that this if the robot has to run over the obstacle while running forward with kinetic energy, it should be able to climb higher obstacles [20].

All spherical robots moving using torque experienced at the center of the shell must deal with these realities. This includes; spherical robots with a flywheel IDU, such as the one developed by V. Joshi [18], which move as a consequence of added angular momentum. Whereas for spherical robots with a mass movement based IDU inside the robot such as the Spherobot by Mukherjee et al. [15], movement is achieved by the movement of the C.G. which creates a torque at the center of the shells. While for direct drive IDU where the

motor torque is directly transmitted to the outer shell as the driving force for the robot [22]. Many spherical robots use mixtures of these three systems as shown in the classification chart by C. Chiu [22] in Figure 2.8.

2.2.2 Conservation of angular momentum

From Equations 2.9 and 2.10 it is evident that movement using shifting mass has its limitations. Given this reality some spherical robot designs have adopted other means of propulsion. For example spherical robots have been built using systems which utilize conservation of angular momentum. This includes the Reaction Wheel, Momentum Wheel and Control Moment Gyroscope [1] designs. The idea behind the Reaction Wheel design is to use the reaction torque from a flywheel when it is accelerated. The reaction wheel is normally not set to rotate. Only when it is needed to turn the vehicle does it start to rotate. This puts a constraint on the system as a motor with a high torque is required to start a flywheel with a high moment of inertia. A second approach is the Momentum Wheel. Here the motor keeps on rotating the flywheel at a high speed. When needed a braking system is used to stop the flywheel quickly to exchange momentum [1] and the flywheel is accelerated to turn in the other direction. While an additional braking system is required in this system, the motor does not need to have as high a torque. Hence a smaller motor can be used. A third approach is the Control Moment Gyroscope. The Control Moment Gyroscope (CMG) approach is widely used in satellite and spacecraft. In a CMG system the flywheel's speed remains constant and instead precession torque generated by the flywheel is harnessed.

										
Robot Configuration	Single wheel	Hamster car	Pendulum on rotating axis	Gimbal mechanism	Single ball	Omni wheels	Mass movements	Orthogonally mounted flywheels	Flywheel on pendulum	Scissored-paired flywheels
Driving mechanism	D	D	D	D	D	D	G	A	D/A	A/G

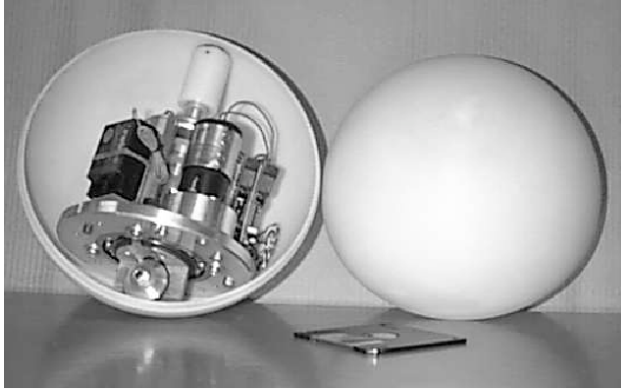
Figure 2.8: Comparison table of Spherical robots with different internal drive unit, updated from [22]. *D* stands for direct drive, *G* stands for drive using gravitational force and *A* stands for angular momentum drive.

2.2.3 IDU design

Many different types of internal drive systems have been used for spherical robots, a number of which are illustrated in Figure 2.8. Each design shown in Figure 2.8 has its own advantages and disadvantages. It is essential to go through each approach to identify their weakness and strengths.

Single wheel This system incorporates a method to reduce slippage between the shell and IDU, as shown in Figure 2.9 and described in [23]. While increasing the tension between the shell and the IDU reduces slippage it also helps the IDU to not lose contact if the vehicle is airborne. Due to the simple design construction and design is much easier compared to other designs shown in Figure 2.8. Due to the simplistic nature of the design, modeling is also much easier and hence control of the system is easily implemented. All these advantages make this a great platform for research. Unfortunately adjusting the spring to reduce slippage increases friction, which is not ideal for a mobile robot that is powered by a battery. Another design constraint is that “the IDU system must also be extremely well balanced. An off-axis center of mass may cause the robot to travel in an unwanted pattern” [24].

Hamster wheel The Hamster wheel is perhaps the simplest IDU design amongst all the designs, illustrated in Figure 2.9b. The name of this IDU describes the idea behind this



(a) Prototype of a single wheel spherical robot.
Image reprinted from [23].



(b) Prototype of a hamster wheel.
Image reprinted from [25].

Figure 2.9: Single and Hamster wheel robot.

system. Essentially an RC car is placed inside a plastic shell, as done in [25] just like a hamster inside a wheel. Having most of the mass near the surface of the shell makes the system efficient in transferring torque to the shell, as mentioned in the static modeling section. Ease of manufacturing and modeling makes this an attractive designs. A simple control system can be implemented. One disadvantage of this approach is that the RC car can loose contact with the shell. Also, odometry data can be poor due to slippage between the car chassis and the robot's shell. This introduces more noise in the control system of this vehicle.

Pendulum on Rotating Axis Just as the name suggests in this design a mass hanged from the center of a shaft which is connected to each half of the sphere. The mass swings forward, backward and left and right, thus generating torque to move the robot forward and to tilt it from side to side. Moving the mass to one side and then rolling forward makes the vehicle turn. Good examples of a vehicle like this include the Rotundus [26], M. Nagai's robot [21], and the Roball [27]. Instead of carrying a mass for the pendulum, these robots uses their internal parts, such as motors, batteries, etc., as their bob of the pendulum. There is no slippage or efficiency loss due to friction. This is a simple design and relatively easy to

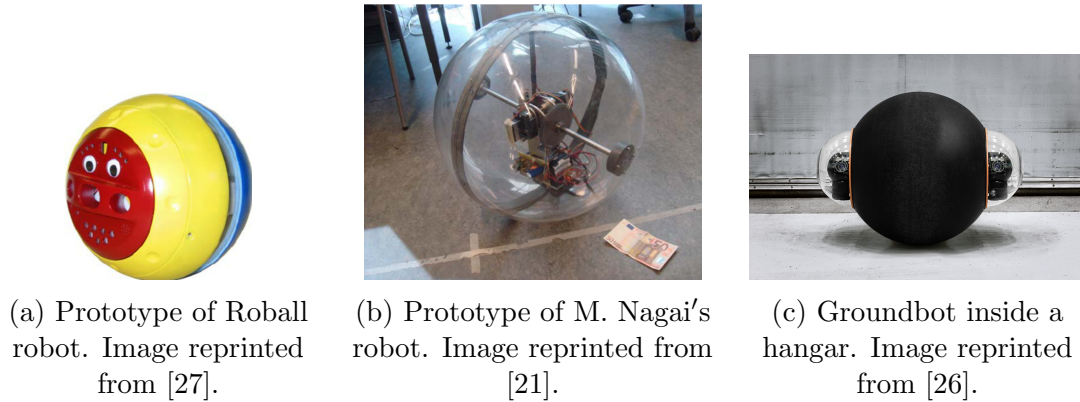
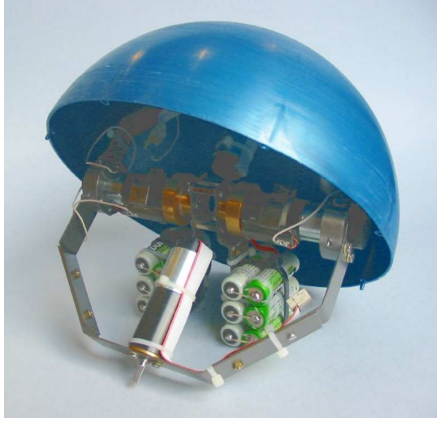


Figure 2.10: Three robots with pendulum on rotating axis system drive.

manufacture. One disadvantage of this approach is that it requires a larger turning radius than the previous systems, since the mass cannot all be in the bob, and neither can the mass “be shifted outside of the boundaries of the shell” to tilt the vehicle to turn [24]. The torque requirement is much higher for the bob to move forward as compared to the system described earlier. Hence a motor with high torque will be required. The shaft on which the entire system hangs from needs to be strong to withstand impacts and other loading.

Gimbal Mechanism The construction of Roball inspired the development of the “RoBall” with an internal gimbal mechanism [28] as shown in Figure 2.11a. Later other robots adopted a similar design. The idea behind this design is to improve the construction method of the previous system. Doing this improves the robustness of the previous system while keeping all the advantages of it. It is easier to mount equipment without being concerned about loading issues on a shaft. Just as with the previous system it has better tracking, hence improved odometry data. Modelling is fairly easy as well. The design and manufacture are not as obvious as for the other designs. Complicated mechanism increases the potential for failure in the design and hence this design may not be as robust as other approaches.



(a) The RoBall overview. Image reprinted from [28].



(b) Spherical robot OmniQiu. Image reprinted from [29].

Figure 2.11: Example of a gimbal mechanism and a single ball system robot.

Single Ball In the single wheel design the wheel can only move forward in one direction. In order to change the direction of the wheel another motor is used [23]. The single ball system is essentially an improvement on the single wheel system. Whereas in the single wheel system the wheel was transferring torque to the shell, here both of the motors transfers the torque to a single ball which is transferring torque to the shell. A good example of implementation of such system is the OmniQiu robot [29] shown in Figure 2.11b. Just as with the single wheel design this approach has a simple control system that is also easy to manufacture. It also inherits some of the issues of the single wheel design such as slippage between the ball and the shell. Using stiffer springs may reduce this effect that but this will then increase friction and therefore results in loss of power. While in the single wheel system there was only one point of slippage that is between the shell and wheel, here there are multiple points between both the rollers and the ball and between the single ball and the shell. This can result in poor odometry data [22].

Omni Wheels This approach can be considered as an enhancement to the hamster wheel design. Essentially a vehicle is placed inside a spherical shell. Unlike the hamster wheel

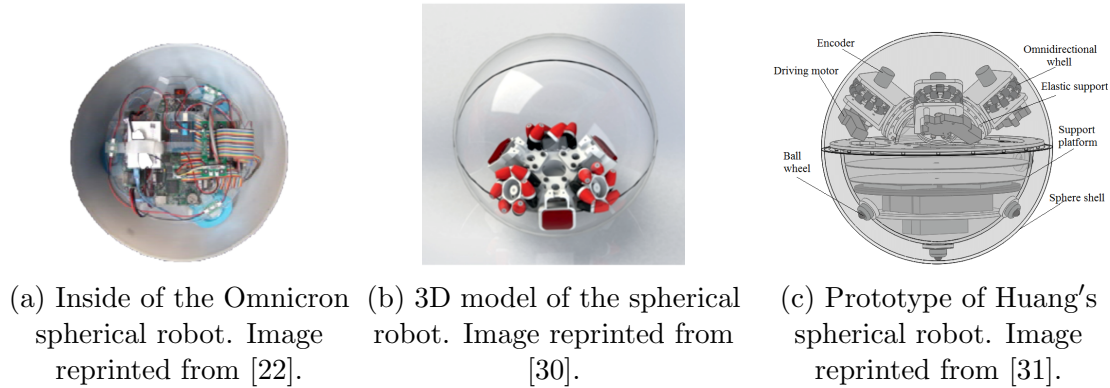
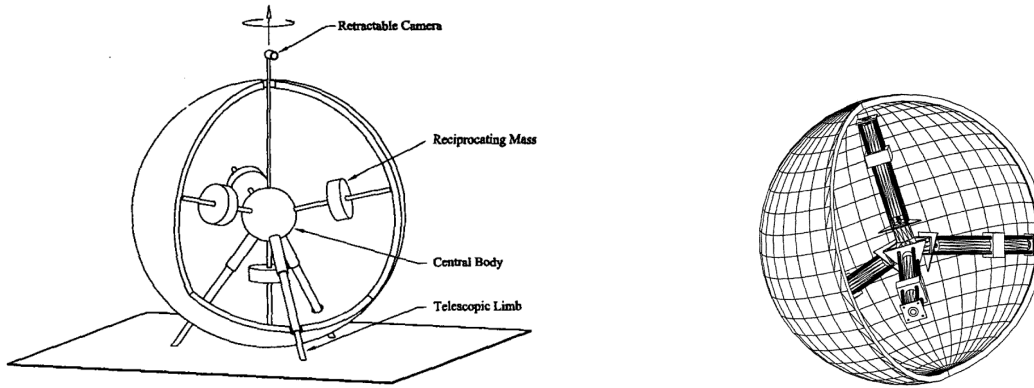


Figure 2.12: Three robots with omni wheel system drive.

where the vehicle inside is a four wheel drive vehicle, here an omnidirectional wheel system with three wheels is placed inside to achieve “3-DOF planar locomotion by simple 3 to 3 mapping” [22]. Examples of such vehicles include [30], [31] and [22] (see Figure 2.12). The system is comparatively easy to manufacture. Also developing a mathematical model of the vehicle is easier and hence implementing a controller is straightforward as well. Issues with the original hamster wheel design are still prevalent in this design however. For example, if airborne contact may be lost between the wheels and the shell. Odometry data will still have noise just as the hamster wheel.

Mass Movement Systems These robots use a design based on moving multiple masses in different directions to move the center of mass, which in turn generates torque to move the shell. Inside the shell, multiple radials spoke like structures guide the movement of the masses within the robot. This allows the net center of mass to move. This also results in a true holonomic design, as the robot can move in any direction from any orientation. Coordination of all the mass movements can become complicated and require real-time computation. The design itself is fairly complicated compared to the other designs. Also, even simple motion, such as moving forward fast, requires the motors to move the internal masses quite quickly on their spokes. This requires high power motors and the system may not be energy efficient.



(a) The Spherobot. Image reprinted from [15]. (b) Inner propulsion system of August. Image reprinted from [32].

Figure 2.13: Examples of a mass movement system robots.

These issues introduce design and construction challenges. Modeling is not straightforward either as it requires more input than all other systems, which results in complicated control system. Implementations of this approach can be found in [32] and [15].

Orthogonally Mounted Flywheels These robots use another novel idea, based on the conservation of angular momentum. First introduced by V. A. Joshi in [17], two pairs of diametrically opposed CMGs are installed inside the shell of the robot. True holonomic motion can be achieved using this approach. Modeling of this system is much harder than most of the previously mentioned systems, hence control becomes more difficult. A simple task such as obtaining constant velocity could require real-time state space calculation. While the design of the system can be much simpler than the previous system, implementation of the control system is much harder.

Flywheel on pendulum This approach is an interesting modification of the pendulum on a rotating axis system. The bob of the pendulum is replaced with a CMG system [33]. True holonomic motion cannot be achieved using this type of as it requires a composition of two separate systems. Turning motion is achieved using a flywheel and direct drive from the

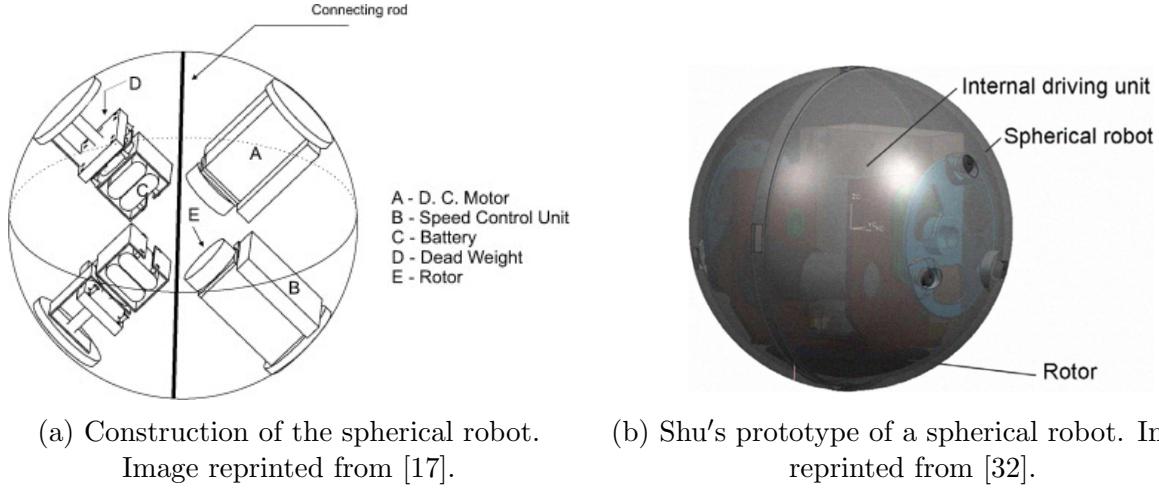


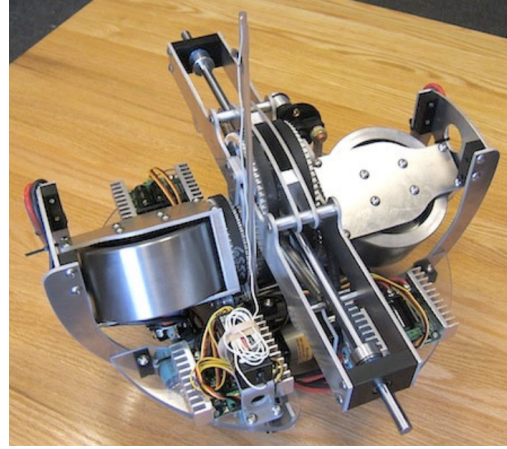
Figure 2.14: Example of a orthogonally mounted flywheels and a flywheel on pendulum system robots.

torque of the pendulum system. The design of this system is complicated and requires the implementation of a real-time control system to properly control the vehicle. Similar to the pendulum system, the torque requirement for the motor to move the bob will much higher, which introduces a design constraint. Just as with the pendulum system the shaft has to be able to withstand heavy loads, impact, plus any dynamic loading from the CMG system. Also, it is notable that flywheel of the CMG is a dead weight which the robot has to carry for the purpose of turning and turning only.

Scissored-Paired Flywheels The Scissored-Paired Flywheel approach is perhaps the most interesting and most novel idea out of all the systems mentioned previously. Using a CMG scissor pair to add a third force to a torque generated by a pendulum system makes the system more efficient than all the previously mentioned system [1]. The maximum torque generated by this system is $\tau_{max} = mgr \times \sin(\theta) + h_{net}$, which is more than the typical maximum torque of $\tau_{max} = mgr \times \sin(\theta)$ generated by other systems.



(a) Fully assembled prototype.
Image reprinted from [1].



(b) Internal pendulum assembly.
Image reprinted from [1].

Figure 2.15: Example of a scissored-paired flywheel system robots from [1].

The two CMGs inside the robot are rotated in the same magnitude but in opposite directions, to cancel the torque reaction due to rotation. However any action on the pair causes a reaction torque h_{net} [24]. This extra torque works within a set range of 180° , after that it flips direction. This is an added complication for the control system design. Also it is a complicated system to model and control. It should be also noticed that two flywheels need to be carried as well, which requires more energy. The design of the system is very complicated and thus manufacturing would be difficult.

All of the above described systems have their own unique strength and weakness. Each design utilizes either an off center mass or conservation of angular momentum or direct drive method. Each system uses either combination of these three principles or just one. All of the systems can be tabulated as shown in Figure 2.8. Implementation of these system also brings unique compromises and advantages. It is vital to come up with design choice that is optimized to the need of this project. A systematic approach to such selection method was used to make this decision.

Value	Symbol	Description
0		None
9	□	Strong
3	●	Medium
1	○	Weak

Requirements	Importance	Percentage	Single wheel	Hamster car	Pendulum on rotating axis	Gimbal mechanism	Single ball	Omni wheels	Mass movements	Orthogonally mounted flywheels	Flywheel on pendulum	Scissored-Paired Flywheels
Precise motion	5	19%	●	○	□	□	□	●	○	□	□	●
Sharp turn	4	15%	○	●	○	●	○	□	□	●	□	□
High speed	3	11%	□	●	□	□	□	□	○	●	□	□
Energy efficient	3	11%	○	○	●	□	○	○	○	●	□	□
Simple control system	3	11%	□	□	□	□	□	●			○	
Robust	4	15%	●	□	□	□	●	○	●	□	□	□
Manufacturing Ease	3	11%	□	□	○	○	●	●	○	□	●	○
No Dead weight	2	7%	□	□	●	□	●	●	○	○	●	
Total	27	100%										
	Importance Weight		4.93	5.07	5.81	7.22	4.93	4.04	2.37	5.19	7.00	5.33
	Importance Percent		9%	10%	11%	14%	9%	8%	5%	10%	13%	10%

Table 2.1: House of Quality matrix for IDU designs with different internal drive units.

2.2.4 House of Quality and IDU selection

The choice of IDU is very import for the performance and construction of the robot. A systematic approach is needed to make a proper choice of the IDU design. In engineering typically there is no perfect solution and every solution comes with a compromise. Coming up with a proper solution is often involves finding the best compromise of all solutions to the requirements. The House of Quality Control is a matrix from Quality function deployment where a common point of reference can be established for design and successive analysis [34]. This allows one to systematically rationalize design decisions from requirements and also characterize the compromises.

A matrix of requirements and described systems is shown in Table 2.1. Importance weights are assigned to each requirement. It should be noted the values assigned each of the requirements is subjective. From Table 2.1 it can be seen that the *importance percentage* of

the flywheel on pendulum two and gimbal system are higher than the others. This suggest that the solution should be based around these approaches. It should be noted that a more precise evaluation of each system could have been done, but due to lack of information of other robots it was not possible.

2.3 Teleoperating spherical robots with ROS

The robot operating system (ROS) [35] is a standard software middle-ware for research robots. It provides a message-passing framework that can be easily modified and updated by the user. It enables users to “create functionalities that can be shared and used in other robots without much effort” [35]. Given its wide adoption within the research community, ROS is used in the spherical robot developed here. Unfortunately there does not, at present, exist a standard software stack for ROS-based spherical robots. Thus a challenge for this work is the development of an appropriate ROS stack for spherical robots generally.

2.4 Summary

Spherical shaped robots present an interesting challenge in robotic control systems design. Unfortunately their relative mechanical complexity has limited their development and as a consequence common autonomous vehicle tasks such as visual teleoperation are relatively poorly understood for such vehicles. By design a spherical robot is extremely maneuverable compared to many other options, such as a four-wheel drive vehicle or a tracked vehicle. Having a external shell and internal drive unit protects the vehicle from external debris and

particle. All these features make it a great platform for field robotics. The design does, however, introduce a range of complications in terms of sensing and control.

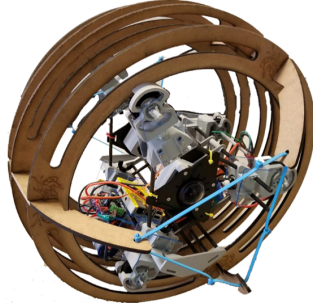
A large literature on spherical robot design exists. Spherical robots can be divided by their shell and IDU designs. This offers an effective approach to categorize spherical robot designs. Most spherical robots built to date are mostly used for experiments in the lab. There is a need to develop spherical robots for more robust and challenging outdoor operation.

Teleoperation is a key challenge for all mobile robots. But this is especially true for spherical robots for which the problem is complicated by the fact that the sensor platform rotates with the motion of the robot. Classical approaches to teleoperation includes state feedback control design which can not be implemented easily for spherical robots [9]. Recent developments in microcontrollers and single board computers has made implementation of spherical robot control system more practical. Can these components be easily adopted to enable the construction of a spherical robot that can be visually teleoperated in a complex environment? This question is considered in the following chapters.

Chapter 3

Designing & building the Dragon Ball

Given that there exists a large design space for spherical robots, it can be difficult to make informed design choices without prototyping a full system and using this prototype to identify the strengths and weaknesses of different approaches. Recognizing this, an iterative design methodology was followed and a prototype robot was built first to address design decisions related to an appropriate IDU and to review other design aspects of a spherical robot. Given the decision to proceed with a prototype ahead of the final design, the question becomes one of the nature of the design: should it be pen and paper or a physical construct? A physical prototype provides a better insight into designing and manufacturing a spherical robot than a pen and paper simulation. Such physical design work also helps in understanding manufacturing capability given the resources available. Another key issue in actually building a spherical robot is the rapid prototyping facilities available. The increased popularity of 3D printers and laser cutters has made rapid prototyping an integral part of engineering prac-



(a) Spherico.



(b) Dragonball.

Figure 3.1: The prototype (Spherico) and the final robot (Dragonball). The prototype was built to answer fundamental and practical questions related to the construction of a spherical robot.

tice. Leveraging these advances and recognizing the additional prohibitive value a physical prototype has over a simulated one, a physical prototype, called “Spherico”¹, an inexpensive spherical robot prototype was built first. The design and construction of Spherico relies on both laser cutter and 3D printing technology as well as off-the-shelve (OTS) components. Driven by a Raspberry Pi, the ROS (Robotic Operating System) was used as the underlying software architecture for controlling Spherico. Spherico takes less than 72Hrs to build. The resulting robot, shown in Figure 3.1a, is an inexpensive and robust platform for research into spherical robot control. Details on the design and construction of Spherico appear in [36]. We begin by reviewing the design of Spherico and the lesson learned with this prototype.

3.1 Spherico

Given the various constraints of IDU design noticed in precious section it was noted that *Flywheel on Pendulum* and the *Gimbal Mechanism* were closer to the requirements of this

¹Full details of Spherico appear in [36].

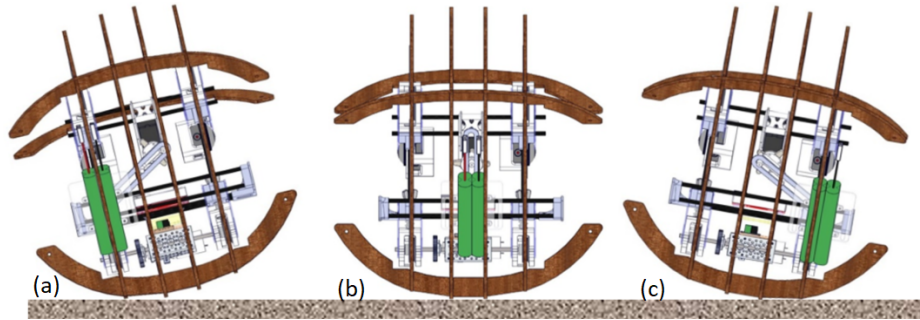


Figure 3.2: Three states of stability for Spherico. When the center of gravity is at the middle the robot stays on its two central support and moves forward (b). When the center of gravity is on the left side the robot will tilt to the left and roll in the left direction (a). Similarly the robot will move to right, when the center of gravity is tilted to the right side (c).

project. A hybrid of the gimbaled and flywheel on pendulum design is used in Spherico. This approach allows rolling movement about one axis using a timing belt. Using a timing belt helps to prevent slippage between the device motor and the frame. Motion out of this plane is accomplished through shifting the C.G. of the robot out of this plane. The C.G. of the robot is moved from side to side using a sliding mechanism, as standard rack and pinion systems are relatively expensive. Manufacturing is straightforward for this system and controlling it is less complicated as well.

The external spherical body of the robot should be light and strong. A common approach in many spherical robot designs is the use of an acrylic dome as the outer shell. The acrylic dome provides the robot with two things; a structural shape and a transparent exterior. Unfortunately, a custom sized dome can be difficult to source and can be expensive. The size of the exterior dome is a direct function of the electronic and other equipment that will be carried within the body of the robot. While acrylic offers a clear view of the outside, it can easily lose its transparency by becoming scratched or marked from rough surfaces. This means the exterior will not be completely transparent for any internal cameras. Transparent acrylic is also relatively weak and can easily crack from impact, such as those encountered

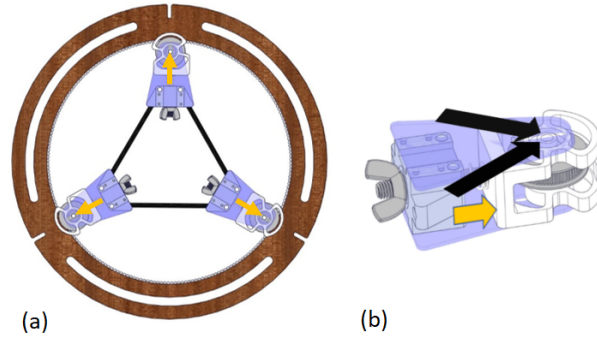


Figure 3.3: Pulley holders exert force on the belt (a) to have them engaged all the time. The arrows in (a) indicates the force from the carbon fiber rods to shaft and the yellow arrow in (b) shows the force from the spring to the pulley.

when falling from even short heights. All these considerations must be taken into account when fabricating the external shell, however for a prototype the ease of use of a stock acrylic dome may be a consideration as well. Another alternative is to just forgo the actual external shell itself, as will be demonstrated below.

While commodity 3D printers can print load bearing parts with complex 3D shapes, they cannot print structures with high stiffness. Laser cutters can cut composite material with high strength and stiffness compared to PLA. This restriction of material strength and shape requires special attention when designing load bearing parts. Cost effective 3D printers typically manufacture parts out of thermoplastic materials such as polylactic acid (PLA) and Acrylonitrile butadiene styrene (ABS). Although high end 3D printers can print in a range of different materials including metal, low end 3D printers typically can only print thermoplastic materials. Spherico was built using PLA extrusion from a Makerbot Replicator+ 3D printer. The PLA extruded design offers the design flexibility required for making complex parts. Unfortunately, it cannot offer the stiffness required in a robot's drive system. The printed parts are also limited in size due to the bed size of the 3D printer and the cost associated with the printing material. Spherico is designed in a hybrid manner,

where complicated parts are designed using 3D printed PLA extrusion and the structures that require stiffness are made from laser cut stock.

3.1.1 Spherico: shell design

Spherico's design was inspired by the spar and rib design of aircraft. It is hard to make a perfect spherical shape from flat parts and fabricating a sphere using a 3D printer would require a substantial amount of PLA, so the spherical shape was reduced in certain orientations. Moving forward requires the robot to be able to roll in it's pitch axis, where a perfect circular shape is required. When turning the robot, rotation in the roll axis is needed. The Spherico sphere design uses three spar and four ribs to make the frame. The ribs are of two sizes, the central ones are of same diameter and the sides ones are of a smaller diameter. The central ribs are separated by 1.8 inches for stability. It should be noted that without a spherical shell the robot has three stable rotational angles. The first state occurs when the robot rolls back and front, i.e. when the robot is stable on the middle two ribs. The second stable state occurs when the robot turns to the right when moving in the forwards or backwards direction. This state occurs when the robot is rolling on the right two ribs. Similarly, if the robot is tilted to the left side, it turns to the left when rolling in the forward or backward direction, as shown in Figure 3.2. With an external spherical outer shell, the robot would have a range of different turning radii. The Spherico prototype lacks this external shell and thus there only exists these three stable orientations.

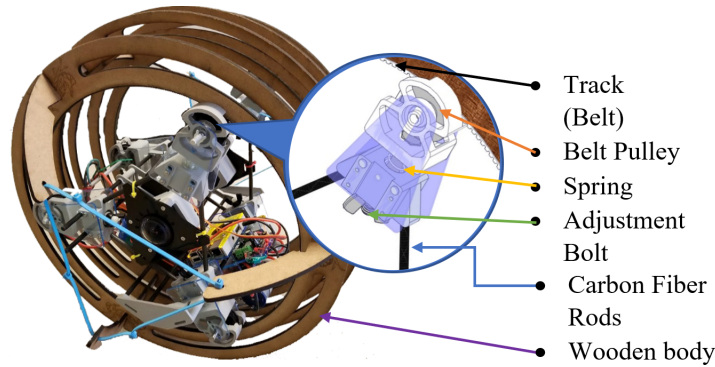


Figure 3.4: In Spherico the drive track is constructed using carbon fiber rods, wooden body, 3D printed holders and Acrylic sheets. The holder holds the belt pulley, spring and adjustment bolt.

3.1.2 Spherico: drive system

The robot uses separate motors for forward drive and maneuvering. A DC motor connected to the drive system of the robot is used to move Spheirco forward and a servo motor is used to make it turn. The single wheel design is modified with three wheels on a track as shown in Figures 3.3 and 3.4. This allows the robot's locomotive design to have a more stable structure. The resulting design is a triangular shape with spring and tensioning pulley at each corner as shown in Figure 3.3a. The drive system is designed using a positive drive system and an adjustable belt tensioning spring, as shown in Figure 3.3b. The drive system is designed to minimize slippage. The force exerted from the springs associated with the other wheels helps to compensate for any slippage, for which proper compliance is required. The springs also act as shock absorbers for the robot.

In order to maneuver from side to side, the required movement of C.G. is performed by moving the batteries between the left and the right sides of the robot. This movement is controlled by a servo motor, as servos have position control precise control of sliding is possible. Figures 3.2 and 3.5 illustrate how rotating the servo clockwise or counter clockwise

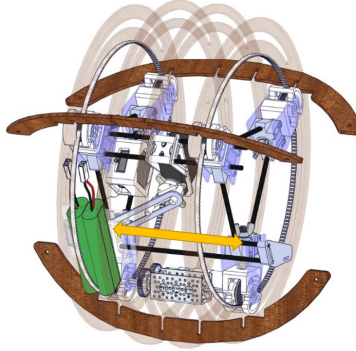


Figure 3.5: The C.G. of the robot moves from the left to the right due to the sliding movement of the batteries. The batteries are pushed by a link system connected to the servo.

causes the batteries, which are a convenient mass within the robot, to move from the left to the right.

3.1.3 Spherico: dynamics

The prototype robot has three specific states of stability, center, right and left as shown in Figure 3.2. A shift in the center of gravity towards the left leans the robot to the left. If the robot is tilted left, then it will roll to the left and vice versa. When moving forward the center of gravity is kept in the middle of the robot. The 1.8 inches wide base makes the robot very stable from external disturbances, such as an uneven ground plane, small obstacles on the floor or even small impacts from the side. This width ensures stable operation when the robot is in its central stability state. While the 1.8 inches wide base of the prototype provided stability, it introduced an extra distance that the sliding system needed to cover for the system to change rolling direction. The displacement of the C.G. needed to be large so that it can tilt the robot on the side. As a consequence of the positioning of the outer spars the turning radius of Spherico is approximately 54 inches.

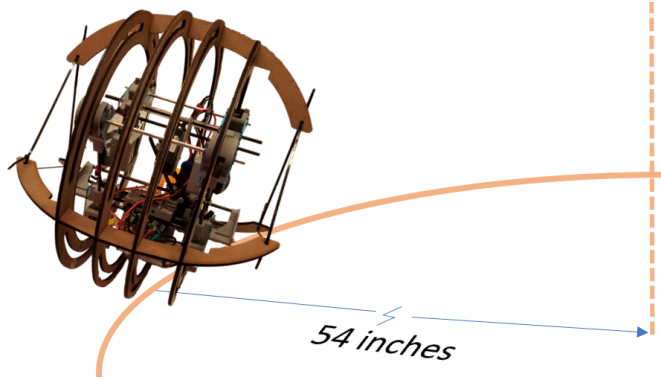


Figure 3.6: To turn, Spherico sits on its outer most two ribs. Given the difference in the rib diameters, the vehicle makes a turn of a 54 inches radius.

3.1.4 Driving Spherico

Initially Spherico was controlled by controlling the PWM of the motor, thus controlling the power of the drive motor. It was quickly observed that this makes it difficult to control the Spherico as the internal IDU keeps on climbing up the track and the center of gravity passes through the gravity line. Once the center of gravity passes through the gravity line of the robot, the torque generated by the offset mass goes to zero and eventually becomes negative. Implementing a speed limit made it easier to control Spherico. The speed limit also reduced the forward speed of the robot. A better control system was required to achieve higher speeds.

The offset angle between the gravity line and the moment arm is proportional to the torque generated by the system, as discussed in Chapter 2. So controlling the offset angle is critical. A PD controller was implemented in Spherico to control the vehicle. The joystick is mapped to a maximum angle and minimum offset angle. The maximum angle is set such that the offset mass is at 90 degrees from the gravity line, thus generating the maximum torque. And the minimum is set to generate maximum torque in the opposite direction. The error input is obtained from the pitch of the IMU.

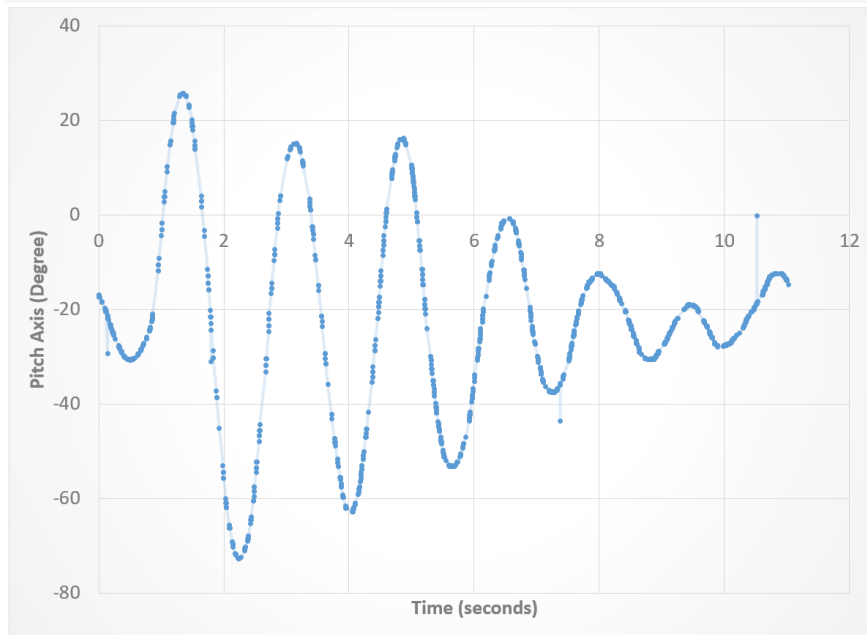


Figure 3.7: IMU data after applying break in the Spherico. The oscillation was measured to have a frequency of 0.6 Hz.

One other issue that was noticed while driving Spherico is that the robot would swing back and forth when the vehicle was stopped after moving forwards. This oscillation happens due to the fact that the robot is stopped while there is an offset between the gravity line and the moment arm, thus it swings like a pendulum. Pitch data was recorded when the robot started to swing after breaking. A typical data set is shown in Figure 3.7. The frequency of this swing was calculated to be 0.6 Hz.

3.1.5 Lessons learned

The rocking motion described previously is commonly known as “*Sloshing*”. Commonly the term sloshing is used to describe the motion of liquid inside a container due to some sort of external disturbance such as sudden acceleration or deceleration. A means control of sloshing in fuel tankers, rockets, submarines are frequently required [37]. Although the

Spherico is not carrying any liquid, the motion of the offset mass is similar to when a liquid mass undergoes oscillatory motion due to sudden braking or acceleration and then decays to equilibrium. Control of liquid sloshing has been studied in depth for application in satellite, tankers rockets etc. Various method of sloshing control have been developed, an appropriate method needs to be adopted for the DragonBall in future.

The on board computer used for Spherico was the Raspberry Pi. While it is a very popular single board computer with huge resources, the computational capabilities of it are limited. In order to process 4K video a more powerful computer was required for the DragonBall.

3.1.6 Spherico: summary

The crucial contribution of the prototype was to understand the drive system design of spherical robots. A few different variants of the drive system were built before settling for the one shown in Figure 3.4. Spherico's drive system is a positive drive system with belt tensioning spring. The compliance of the springs is critical to ensure that the drive does not slip. If the robot is sitting on the bottom spring, the pull of gravity ensures that the belt drive is engaged. When the robot starts to move forward, the center of gravity (C.G.) starts moving up and the springs need to maintain proper engagement of the gear to the belt. Proper compliance is required to ensure that the gears stay engaged to the belt. Another important property that was learned from the prototype robot was the nature of the transition phases between one dynamic state to another. If the robot stops suddenly, the entire body starts to swing back and forth around the center of gravity. An appropriate method to stop this movement is required.

3.2 DragonBall

The DragonBall is designed to be a fully enclosed spherical robot with enclosed actuation and sensor systems. The basic requirement behind designing the DragonBall was to design and build a robot that can carry sensors within a sealed spherical shell. As the DragonBall is to be teleoperated visually it is critical that the operator has a stable view of the environment from the rotating platform. In order to accomplish this given an opaque shell, camera ports are required. In order to minimize the number of such ports and at the same time provide a wide field visual field DragonBall relies on two wide field of view cameras mounted in ports along the vehicle's axis. This leads to a shell design similar to that of the Rotundus (Groundbot) vehicle [26]. Such ports also provide access for wireless communication and a port for charging and wired access to internal electronics.

3.2.1 Mechanical design

Other than the internal mechanical design aspects of the robot design, it is also important to consider usability issues of the robot. This robot is intended for field deployment. There are several key issues to consider while traveling with robots. One issue is weight and the other is size. While determining the size of the robot is dependent on sensors, onboard computer, and design of internal, it also plays a critical role in transportation. Traveling is easier with a robot that can fit inside a case that can be transported as baggage by commercial airlines. The case protects the robot while travelling. While keeping all this in mind, several different manufacturing factors had to be considered too, such as the laser cutter size, standard material prices etc. Considering all these factors, it was determined that a 16-inch sphere would

be the smallest possible spherical robot that can be built with all the sensors inside. In comparison the weight of the robot was easily determined to be as low as possible.

3.2.2 Outer shell manufacturing

Research robots are often built to test the viability of such platforms to perform certain tasks. Quite often such research platforms are designed only to work within the lab or in an indoor environment. DragonBall however, was built to validate the design of a spherical robot platform for both indoor and outdoor experiments. Unlike a typical indoor robot it is important for an outdoor robot to be robust to a wide range of conditions. This introduces further design challenges. Mobile robots operating in an outdoor experiment can be easily damaged due to mishandling, bad weather, rough terrain, etc. As discussed previously, the shortcomings of acrylic dome design made it necessary to search for materials with light weight and good stiffness.

Composite building materials have become very popular due to their availability for commercial applications. Fiberglass and carbon fiber are perhaps the most popular composite materials. For this project carbon fiber was chosen over fiberglass as it provides more stiffness than fiberglass. This choice offers increased strength and better scratch resistance while maintaining lower weight over fiberglass. Although the DragonBall robot is light in weight, having a strong body ensures that if operations such as falling from a height are required, the robot will be able to do so. The operation of this robot will be not only limited to land, it should be able to navigate the surface of bodies of water as well. If the design is watertight, travelling over water should even be possible! Modifying this robot for operation in contaminated environments will also be straightforward as the robot has no penetration of the shell.

Composite materials are anisotropic, i.e. their mechanical properties along the thickness direction differ from their mechanical properties along other directions. Designing carbon fiber parts required a proper understanding of the load paths. Aligning load paths with the carbon fiber orientation will result in an optimal design. This observation has implications for the vehicle shell. A similarly shaped part from some industrial application would be a dome shaped carbon fiber pressure vessel [38]. Sourcing such a part proved to be difficult and extremely costly. In industrial applications such parts are commonly manufactured using the *filament winding method* [39]. A simpler way to make such a part is the *polar stacking method*, where multiple layers of carbon fiber sheets are stacked one top of each other. This method for manufacturing a dome has been tested previously [38]. Determining the optimal fiber orientation to align to load path or performing a FEA analysis is beyond the scope this work. Pressure vessels or submarine hulls made with carbon fiber have very high pressure ratings. The shell of DragonBall will not be operated anywhere near such pressures. The typical application of the shell would be to work under one atmospheric pressure with added force distribution from the mass of the robot and the effect of ground objects. While these numbers are very low, one instance when they will be high is when the robot falls from a height and then the shell has to withstand the impact. Composite material is not ideal for impact loading, unlike metal [38], yet it is a much better choice than acrylic.

Amongst the various methods of manufacturing carbon fiber parts there are three different methods which are commonly used [40]. The first one is called *hand layup* or *wet layup*. This is the most simple and widely used method. Layers of carbon fibers are laid out manually on the mold, then resin is applied on the fabric. Removal of air bubbles and uniform distribution of resin is achieved by using hand rollers over the fabric. With proper pressure and many repetition all the bubbles are removed and the proper thickness is obtained. Then the structure is cured. The second method is known as *resin infusion layup*, where

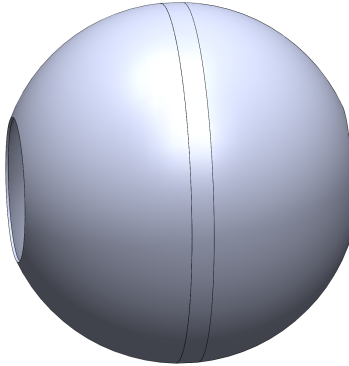


Figure 3.8: Basic shell design: The shell needs to have a flat region at the center and holes on both sides such that the camera can be placed there.

a vacuum is used to press the carbon fiber sheet to the mold while drying. This way the carbon fiber dries out to be thinner than the previous method. The third technique is known as *oven cured prepeg layup*. This is one of the most convenient methods to work with carbon fiber. Instead of buying carbon fiber cloth, sheets of carbon fiber cloth with pre-impregnated are used. This glue only begins to cure under high temperature. This means that the sheets can be taken and positioned again and again until the proper alignment is obtained. This method is widely used in Aerospace industries [40]. One of the downsides of this approach is this that it requires an oven with a vacuum pump. Access to such vacuum is not easily available. Since resources for manufacturing the robot shell are limited, some choices had to be eliminated. The use of prepeg carbon fiber is not possible as a vacuum system or curing oven is not available. Vacuum forming with a hand layout was not practical either. A quotation was obtained from a carbon fiber manufacturer, but the price was beyond the budget of this project.

Before going further into the details of manufacturing the carbon fiber shell, it is important to understand the design requirements of the outer shell. The robot is 16 inches in diameter, and has camera ports extruding from its sides. The sides have opening for the camera dome, as described below. It is desirable to have a flat region at the center of the

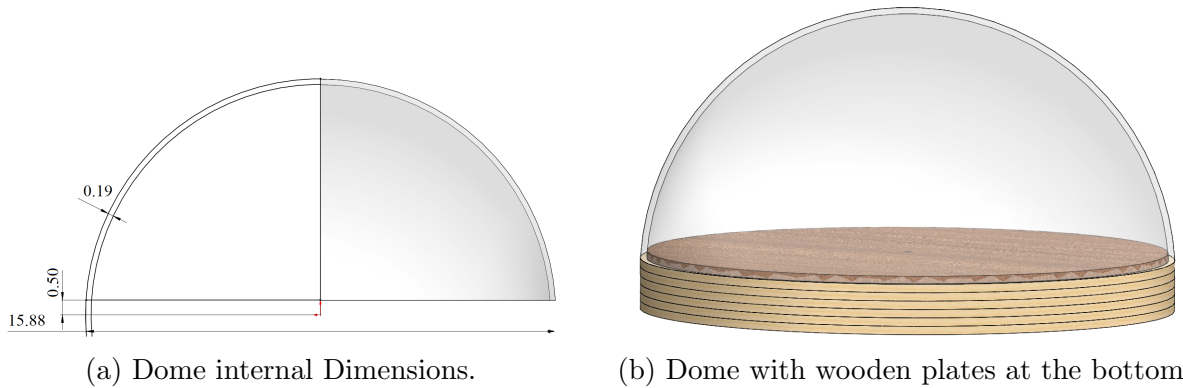


Figure 3.9: Negative mold for manufacturing the carbon fiber shell. Note that due to manufacturing defects the outer diameter is 15.88 instead of 16 inches. The height is increased by adding wooden plates at the bottom.

sphere to ensure stability during rolling motion from small disturbances. In essence, the resulting shape is a sphere with a short tube at its center with the sides cut off, as shown in Figure 3.8. The shell needs to be easy to take apart. Forming the robot as two separate domes and then attaching them near the seam is an obvious design choice. A dome of 16 inches diameter was used as the mold for making the outer shell, shown in Figure 3.9a. The dome has a 16 inch diameter, but is only 7.5 inches in height due to manufacturing constraints. Since the shell requires two domes, one can be made of height 7.5 inches and another elongated one inch length with a cylinder. Since the dome is only 7.5 inches high a wooden platform was built to increase the height, as shown in Figure 3.9b.

For this project the hand-layout method was used. Carbon fiber cloths come in various weights and fiber layouts. The strength of the final design depends on the orientation of the fiber to the load path [40]. The weight of the fiber in the cloth plays a role in the strength of the design. Heavier carbon fiber cloths are hard to work with as they will not conform to the mold easily. The ability of a fabric to conform to a shape is also known as its drapability. As Plain weaves “are not very drapeable” [40] compared to Twill weaves, for this project a Twill weave was chosen. If the carbon fiber material is bi-directional then it becomes even

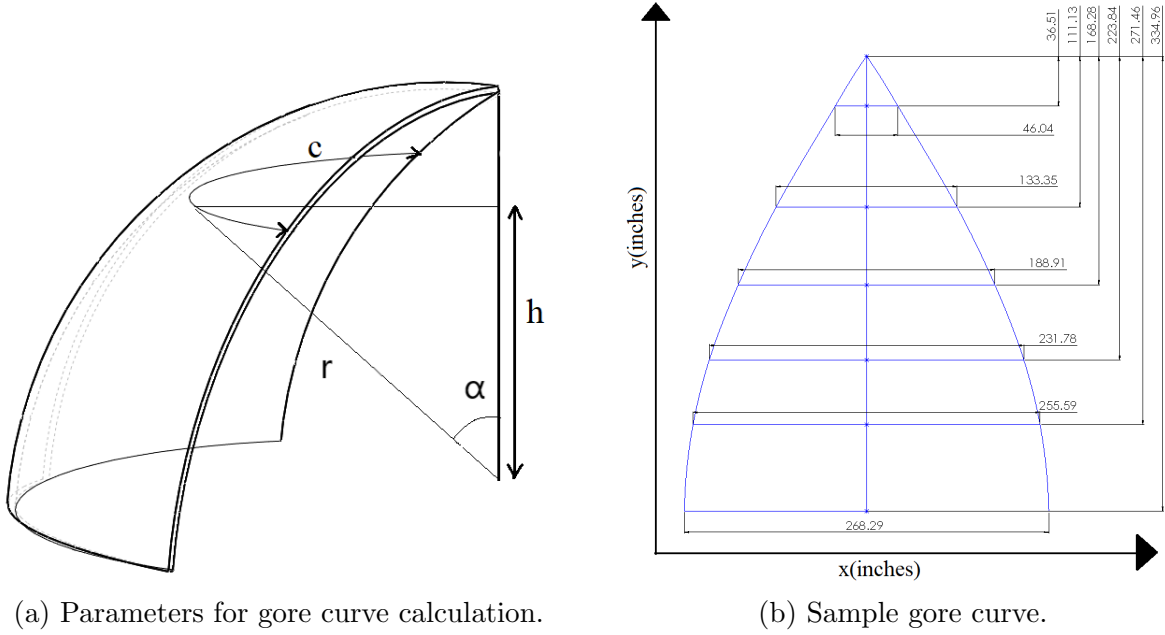


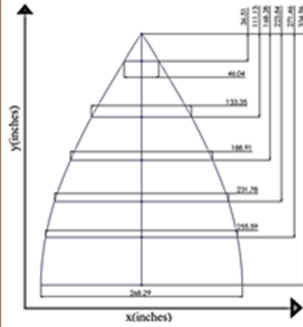
Figure 3.10: Gore Curves. (a) At different height, h , the circumference is calculated. (b) The calculated circumference at each height is then plotted and a spline is fitted to make a proper gore curve.

less drapeable. Bi-directional carbon fiber was also used to handle multi-directional loads [40]. The best way to solve the drapability issue is to cut the carbon fiber in such a way that the material has to stretch minimally to conform with the mold.

The construction of the DragonBall dome follows the basic strategies described above. Unlike [38] where the shape of the dome was proper, the top part of the shell is open. Also, in [38] the dome was constructed using “peel-like” shapes to form the dome. The resulting layout has more seams near the bottom of the dome than the top. Having too many seams is not ideal in a layer of carbon fiber layout. Since the base of the dome is constructed from the peels connecting together, it might be weaker than the top. So it was necessary to develop an alternative layout. Gore curves are conventionally used for fabricating a dome out of fabric or even parachute cloth [41], see Figure 3.10. A similar approach was used to form the dome. While reducing the numbers of seam is the goal, the constraint becomes

1. Determine:

- (a) Segments: Number of segments that will be cutout.
- (b) Num_Coordinates: Number of coordinates calculated to draw the curve.



2. Calculate

- a) $Circumference = 2\pi r$
- b) $BaseWidth = \frac{Circumference}{Num_Segments}$
- c) $Height = \frac{Circumference}{4}$
- d) Circumference



3. Plot:

- (a) Draw an inverted T shape with base width of BaseWidth.
- (b) The height of the T shape is Height.
- (c) Draw lines on the inverted T shape at specific heights, as shown in Figure 3.9b.
- (d) Fit two polynomial curves through the line ends, as shown in Figure 3.9b.

Figure 3.11: Steps in the fabrication of a gore curve.

dealing with the drapability of the fabric. The fewer the pieces of material the less the likely the material is likely to conform to the dome shape. The basic idea behind the use of gore curves is to divide the dome into sections and to calculate the circumference of the curve at different heights of the hemisphere. Each height defines a specific circumference value, as shown in Figure 3.10a. These values are then plotted on a vertical line, as shown in Figure 3.9b. Fitting a polynomial line through the ends of the lines gives the desired gore curve. The steps taken for gore curve calculation are described in Figure 3.11.

One issue with cutting larger sections of the carbon fiber fabric is that the fabric might not even conform to the dome shape. With experimentation it was discovered that a cutout of five sections was able to conform to the dome shaped mold. One section out of the five was fabricated, using two layers of twill weave and one layer of $\pm 45^\circ$ weave. After curing the



Figure 3.12: The gap between layers is marked on the mold. The gap was approximated by an exponential function.

carbon fiber piece was found to have a weight of 73 gm and 1 mm thickness. The cured piece was placed on the mold and the outlines were marked near the seam of the next section. Then the cured piece was moved to the next section. Each time the seams were marked with a marker. Near the seam of each outline, a gap was noticed. The gap was due to the lack of stretch in the fabric. This gap had a maximum value of 0.45 inches near the center, as shown in Figure 3.12. From the basic outline of the gap it was determined that the shape of the gap could be modeled as an exponential function. From the model's width additional width can be added on both side of the gore curve to compensate for the lack of drapeability in the carbon fiber fabric. Two exponential functions were tested by cutting cloth and putting them on the mold.

$$y = ab^{-x} + c \quad (3.1)$$

$$y = \frac{b}{e^{-ax} + c} + d \quad (3.2)$$

Equations 3.1 and 3.2 were used to generate curves that describe the cutout. Here x is

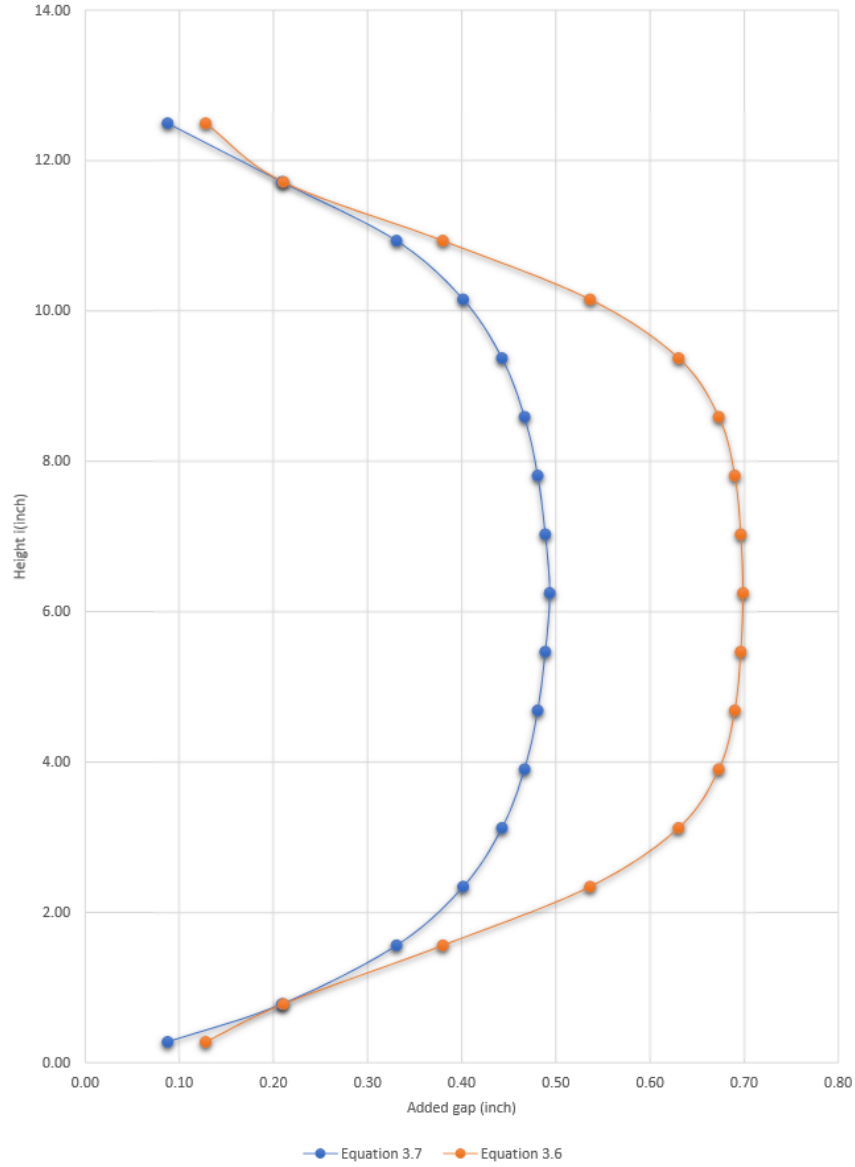


Figure 3.13: Graph of Equation 3.1 and 3.2. The width of the gore curves had to be modified in order to compensate for the gap. Above graph shows Equation 3.1 and 3.2 plots, where the vertical axis is the height in inch and the horizontal axis is the width to be added on each side of the gore curve. Each of these additional values were added to the width side of the gore curve then tested for drapability.

the height of the gore cutout and y is the width to be added to the cutout. The variables a , b , c and d in Equations 3.1 and 3.2 were determined experimentally. The width in the gore curve calculations were then adjusted accordingly. A piece of cotton cloth was laser cut to

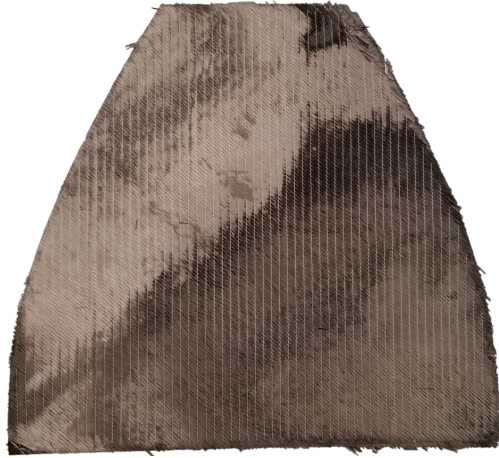


Figure 3.14: The $\pm 45^\circ$ fabric was cut with a laser cutter.

obtain each of the five pieces needed for the dome. Although the drapability of the cloth is not identical to the carbon fibre plies, they were found to be close enough for this purpose. The laser cut pieces of the cloth were placed on the dome to determine how well they meet near the seam. An example of the fit for a given piece as approximated using Equations 3.1 and 3.2 is shown in Figure 3.13.

Two twill weave carbon fiber fabric were recommended by the seller based on pricing and finish. Twill fabric of 5.8 oz and another of 7.3 oz weight were used as well as a bidirectional $\pm 45^\circ$ fabric of 6 oz weight. Cutting the cloth with the laser cutter was tried initially, but for both the twill weave fabrics the strands did not adhere to the fabric after cutting. Given this a hand method was used to cut the twill fabrics. Two MDF board templates were laser cut, and the carbon fiber fabric was clamped between them while the outline was cutout using scissors. The $\pm 45^\circ$ fabric was cut using the laser, as the strands did not fall apart after cutting, as shown in Figure 3.14.



(a) A negative mold where the carbon fiber plies are placed inside the dome.

(b) A positive mold where the carbon fiber is put on top of the mold.

Figure 3.15: Negative and positive molds. (a) Taking a piece out of a negative mold would be harder as another setup would be required to pull it out. (b) Since there is a hole in the top of the carbon fiber piece makes it easier to just push it down once it is cured.

3.2.3 Making the mold

The mold for the robot shell is a positive mold, in which the top is fabricated out of acrylic and the bottom is fabricated from MDF boards. Acrylic is a good material choice for the mold as it does not stick to carbon fiber epoxy. MDF boards are harder to work with. While it is easy to cut the material using a laser cutter, obtaining a polished surface on them is hard quite difficult. A sealer was used to seal the edges of the MDF board and then they were spray painted to obtain a polished finish.

One of the main concerns after making a mold is taking the piece out after it has cured. The reason why a positive mold was chosen instead of negative is due to the ease of taking the piece out after curing. Figure 3.15 shows comparison between a positive mold and negative mold. From the first test piece it was observed that extracting the piece from the acrylic mold was difficult. Pulling the piece out from a negative mold would have been much harder than pushing the acrylic from the top hole while sitting the base. Pushing the piece from the top while sitting on the bottom is a much easier approach. The MDF boards used to increase the length of the piece come off easily due to the limited surface area in contact. In order to further facilitate the removal of the piece from the mold release plies were used



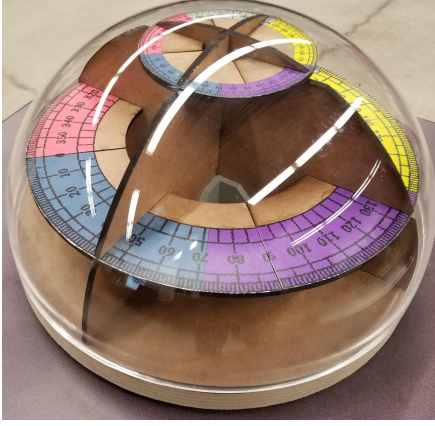
Figure 3.16: Laser cut of the release ply.

(Figure 3.16). The release plies were laser cut, the shape is similar to the gore curve cut before except that they extended above and below to help pull the piece out of the mold, as show in Figure 3.16. Another method to help ensure that the carbon fiber piece comes off the mold involves PVA spray and wax. Waxing the mold creates a layer above the mold which comes off when pressure is applied. The PVA, or polyvinyl alcohol, is a spray that creates a thin layer of film which can resist the epoxy but is water soluble, washing or submersing the setup later helps the PVA film to dissolve away and the piece can come out. For this project to ensure that the piece comes out of the mold, several different measures has been

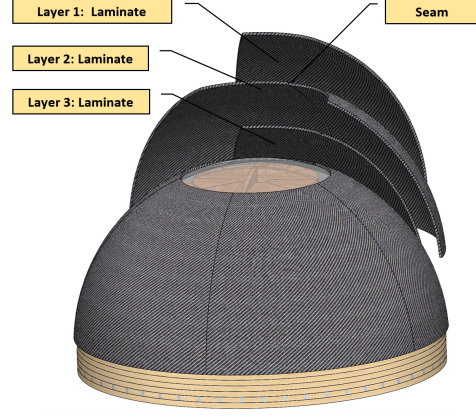
taken, both the acrylic and the edges of MDF board were polished with 6 layers of wax and 4 layers of PVA release.

While making the test piece, it was clear that it is necessary to have some sort of jig to help orient the carbon fiber. A wooden frame was designed to go inside the acrylic dome, as shown in Figure 3.17a. The structure was made up of horizontal and vertical ribs. Colored measuring labels were attached to the vertical ribs to help while placing the fabrics.

Each of the carbon fiber fabric pieces (*ply*), has the same orientation over the entire cut out piece. Several different types of ply stacked and bonded together is called a *laminate*. Since the composite is an anisotropic material, when multiple stacked ply cures, there may exist residual stress after the curing process. The residual stress commonly occurs during the heated cure process, due to various causes such as lay-up, tooling material, different thermal expansion properties etc. While it is hard to predict thermal expansion of a ply, a similar amount of type of residual stress can be expected from plies with similar properties but opposite orientation. Hence a balanced laminate can be made, which would have a $+45^\circ$ and a -45° ply to balance each other out. It is important to balance the residual stress, otherwise “macro-warpage will certainly occur” [42]. To make a balanced laminate, every direction of ply needs to have an opposite direction ply [42]. For this project this was achieved using bidirectional and $\pm 45^\circ$ ply. Also the mold is not cured using heat. The laminates are made of two bidirectional ply with a $\pm 45^\circ$ ply sandwiched between these two layers for symmetry. Three layers of laminates are used in making the shell. Each layer is composed of five laminates placed side by side covering the entire dome. When placing the second layer, the center of the cutout is placed at seam of the previous layer, as shown in Figure 3.17b. Doing this ensures the strength of the dome near the seam. The steps for manufacturing the shell are detailed in Figure 3.18.



(a) Mold with aligning frame inside.



(b) Layup method

Figure 3.17: Each layer is made of three types of carbon fiber fabric, two twill with a bidirectional carbon fiber fabric in between them.

The physical properties of a carbon fiber piece depends on the relative volume of carbon fiber to epoxy [40]. All types of forces exerted on the carbon fiber piece pass through both the epoxy and carbon fibers. The relationship between the modulus of elasticity of the resulting piece, carbon fiber and epoxy is $E = E_f V_f + E_m V_m$ [40]. Here E is the modulus of elasticity of the cured piece, E_f is modulus of elasticity for carbon fiber and E_m is the modulus of elasticity for epoxy matrix. The modulus of elasticity of epoxy is less than that of carbon fiber, so ideally the volume of fiber should be higher than epoxy. Unfortunately the *hand layup* method does not produce the best fiber volume ratio (typically around 60% [40]).

West Marine 105 epoxy and 205 hardener was used in this project. The specific gravity of the cured epoxy is 1.18, therefore the density is $1.18g/cm^3$. The total weight of the cured carbon fiber piece of the short side is 998g and the weight of the plies were measured to be 555g. The weight of the epoxy is 443g, therefore the volume is;

$$V_m = \frac{443g}{1.118g/cm^3} \approx 375cm^3$$

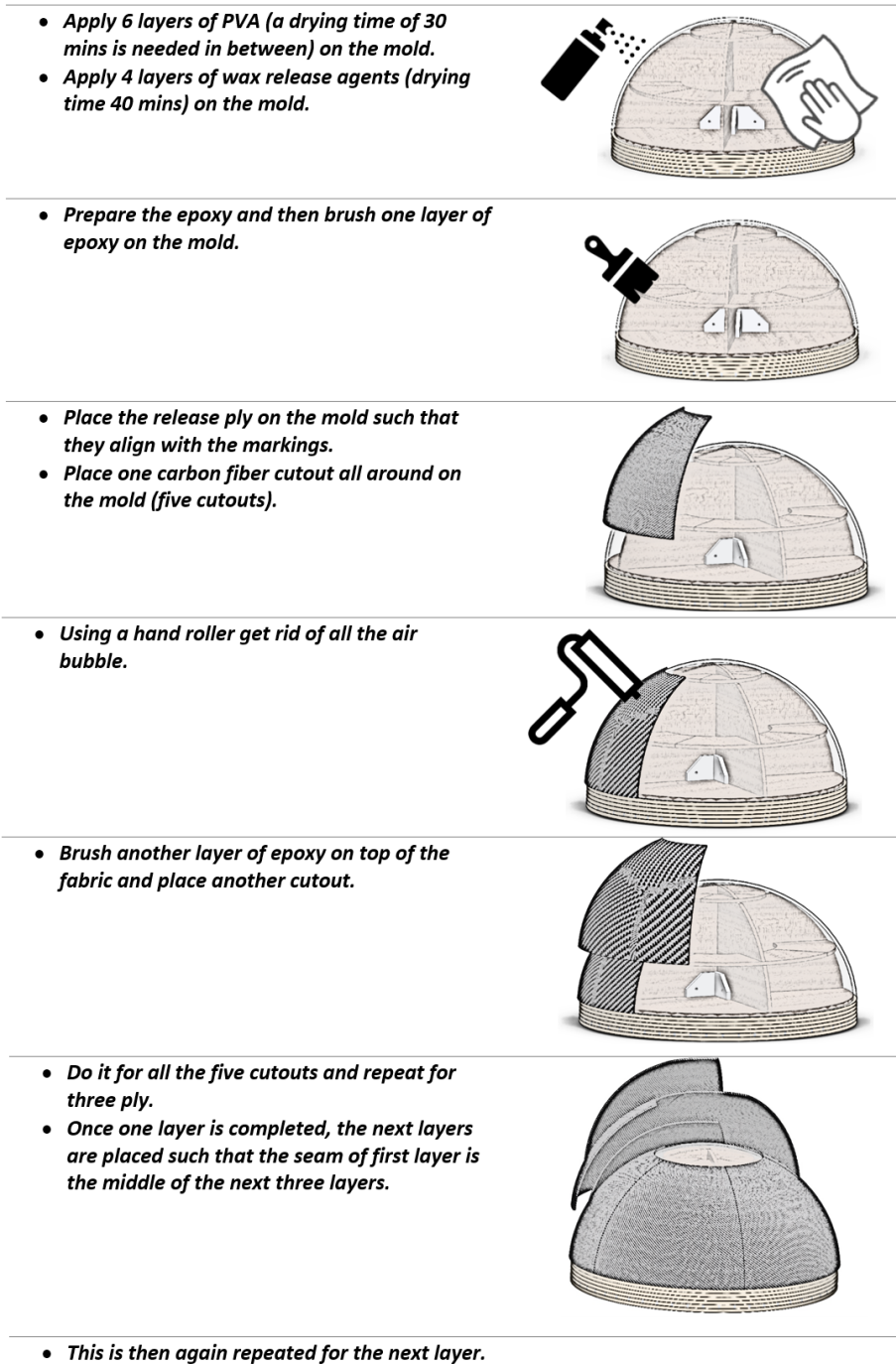


Figure 3.18: Manufacturing process for the outer shell of DragonBall.

The area of each ply is approximated to be 557cm^2 . The thickness of each ply is around 0.02 cm and therefore the volume of each ply is 11.14cm^3 . The shell is constructed of three laminates, each laminates has three plies, hence nine plies in total. As there are five sections in the sphere there are 45 plies in total. The volume of all the plies comes to 501.35cm^3 . Therefore the total volume of the short dome is 876.44cm^3 . That means the V_f is 57%, which is expected of hand layup method. Calculating the theoretical strength of the cured piece is not within the scope of this project. Since the V_f is within the expected range of the layup method means a standard mechanical properties of a carbon fiber piece can be expected.

3.2.4 Shell design

For manufacturing ease, it was an obvious choice to make the dome in two parts. This leaves the challenging problem of combining the two domes. Various complicated systems were explored to come up with an easier solution. Given that the robot will not be disassembled often, an obvious but inelegant solution was to put hands from both side inside the robot and fasten then shells to the track support.

A method of attaching to the aluminium track support to the shells was needed. Also the attachment should be water tight. Two commonly used methods of sealing a system involves either using a gasket or an O-ring. Given that the surface of the track support is flat and has a large surface area, a gasket approach was used. Before going into the details of how the seal is designed it is important to understand how the two shells were attached together. The two shells are attached together as shown in Figure 3.19. It can be seen that there are three seams in the entire design that need to have water tight seals, the two domes on the sides and the main seal in the center. For a gasket seal, two flat surfaces areas are required. In order to provide this surface an acrylic plate was used.

A jig was designed so that the acrylic plates were co-centric, parallel and also at the proper height. The jig was fabricated from MDF board in three parts; a top MDF board, a bottom MDF board and a threaded rod. The bottom acrylic plate is fastened to the bottom MDF board and the top acrylic plate with the top MDF board. The bottom plate is designed such that once it is attached with the acrylic plate, it forms a groove for the epoxy while covering the areas where the screws are placed. Figure 3.20a shows the bottom plate, and Figure 3.20b shows how everything will be fit together. Both of these plates were then passed through the threaded rod and fastened using screws. Adjusting the screws distances allowed to change the distance between the plates. The distance between them is obtained from the CAD design of the robot. The top plate can be lowered and epoxy could have been applied on the groove between MDF board and the carbon fiber shell. Doing this would made a epoxy layer same height as the MDF board, which is only 0.25". Unfortunately a much thicker layer of epoxy is desired.

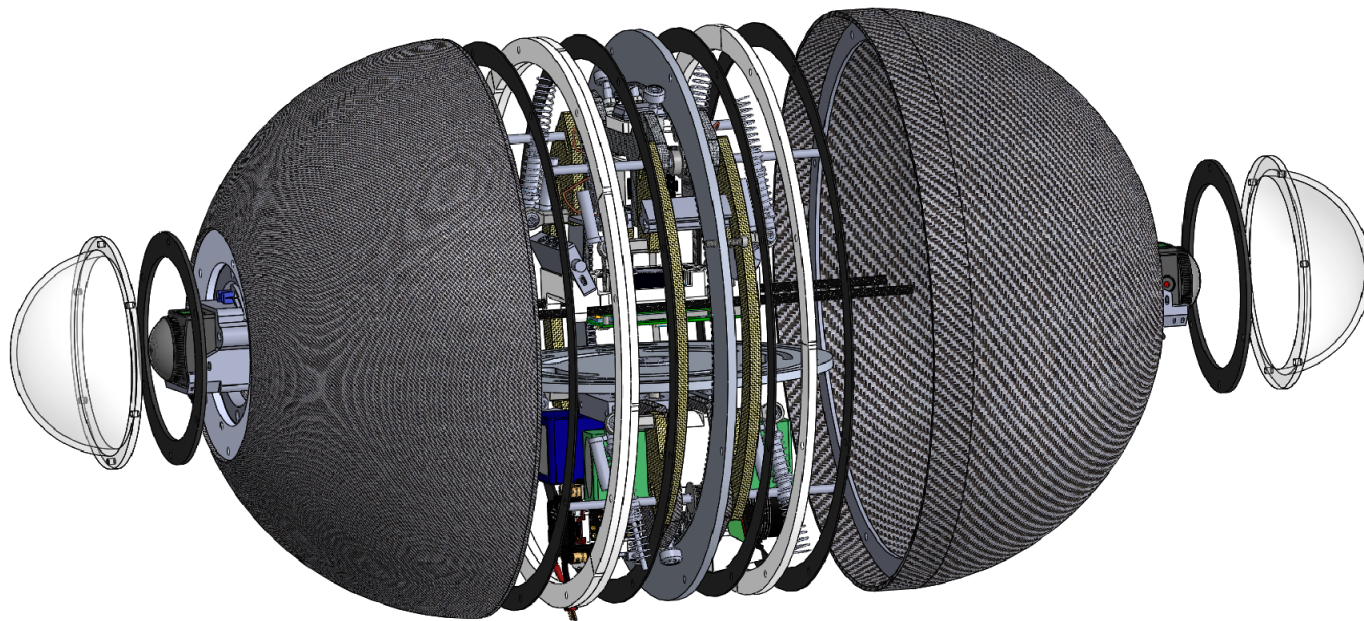
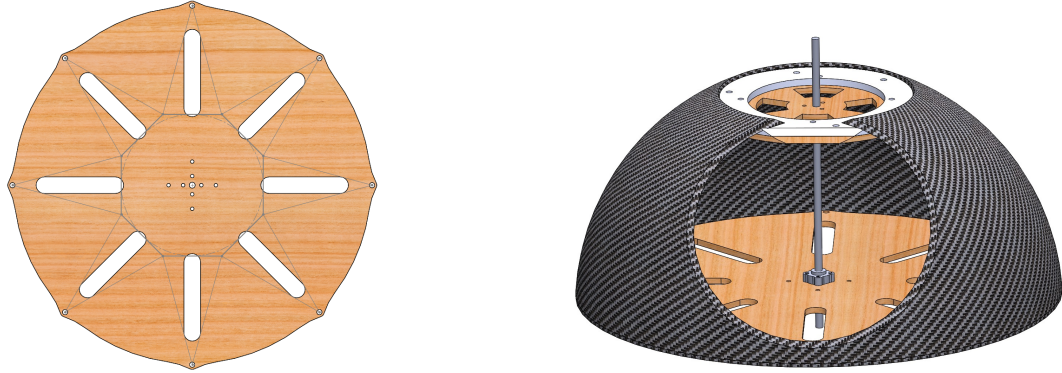


Figure 3.19: Overall assembly of DragonBall.



(a) Bottom MDF board of the jig assembly. (b) Cut through view of the inside of the carbon fiber shell with the MDF jig.

Figure 3.20: Epoxy jig. (a) The bottom board is designed such that the screws can hold the acrylic circular plate underneath. The lines on the board are laser etching which are used later to break the board out the carbon fiber shell. (b) The complete setup of the jig. First the bottom acrylic plate is joined using glue, then the top acrylic part is done.

$$z = \frac{1}{2} \frac{\omega^2}{g} r^2 \quad (3.3)$$

It is well known that rotating container with liquid, makes the liquid's surface take a parabolic shape. Equation 3.3 shows the function of the surface of a liquid in a rotating container [43]. Here z is the equation of the curve, ω is the angular velocity, g is gravitational constant and r is the inside radius of the container. It should be noted that the shape of the curvature is proportional to the angular velocity of the container. Increasing the velocity increases the epoxy height. Using this technique, the height of the epoxy was increased and since the resulting surface shape is a curve the resulting epoxy filler is shaped like a chamfer.

In order to create this curvature a setup made from a continuous rotating servo, supporting collars for the setup to rotate, a marked knob and an Arduino Nano was built. At first a smaller quantity of epoxy was rotated to measure the increase in height of the epoxy for a given rotational speed. The entire jig was then flipped upside down and the bottom

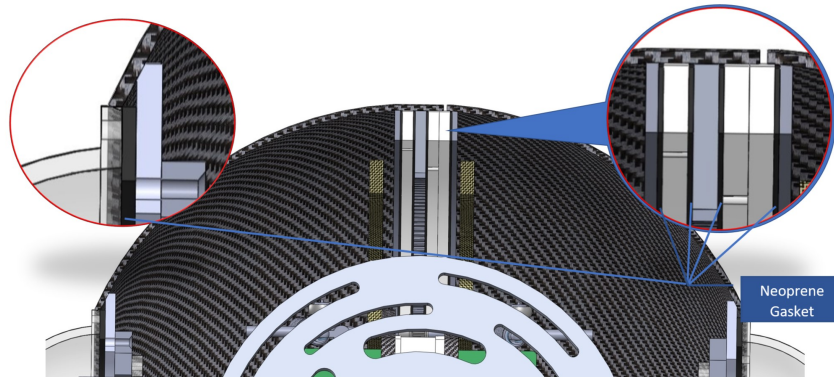
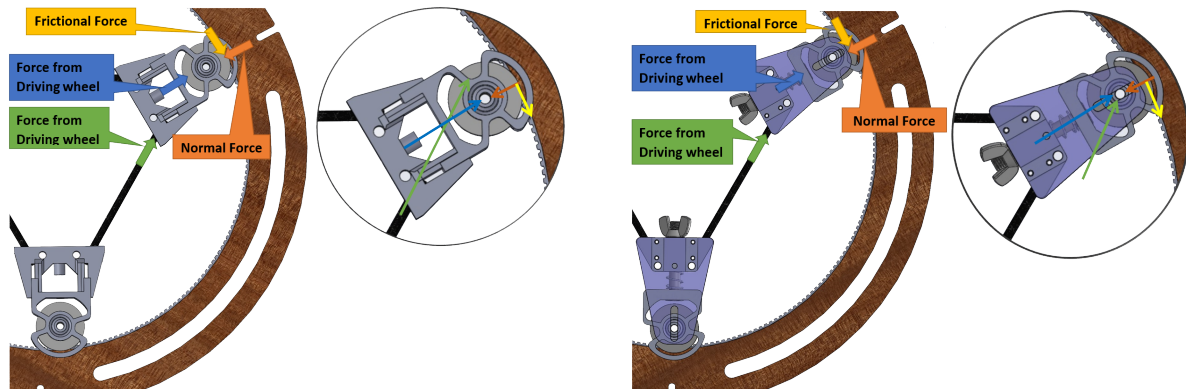


Figure 3.21: DragonBall has six gaskets to seal it from outside environment. Four at the center and one on each side.

plate held in place by using silicone caulking. Then the top plate was lowered so that epoxy could be applied to fix the bottom plate in place. Then the threaded rod was attached to the continuous rotating servo, which was set to turn at the same speed obtained earlier. The entire setup was then left to be cured. Once cured, silicone caulking was applied to the top acrylic plate. Then the bottom MDF board was broken apart so it could be removed and the top MDF board was taken away. The epoxy was then applied to the top acrylic plate from the bottom such that it adhered to the carbon fiber shell. In the DragonBall all the seals are made of 40 Durometer rubber. The seam at the center required four gaskets seals which were cut out using a laser cutter. Figure 3.21 shows how all the gaskets are placed inside the robot.

3.3 Drive system

Spherico provided an essential test bed for the design of the DragonBall. Spherico used a track drive system to manipulate the centre of mass of the vehicle. Amongst the many things that were learned while designing and building Spherico, the drive system was perhaps



(a) Initial wheel holder design where the force from the driving wheel does not pass through the axle.

(b) Update design, the force from the driving wheel passes through the axle.

Figure 3.22: Wheel holder design for Spherico

the most crucial system. The build of the track with a timing belt worked as expected on Spherico. Although this proved effective, it was found that sharp jolts of the drive mechanism could damage the drive wheels relative to the track. So for DragonBall a track drive system was adopted but a more effective (and rigid) track support was used. In Spherico two idler rollers were used in the track system for support and one for driving. The design of the wheel holder was challenging for Spherico, as it required designing a system where the force from the driving wheel goes through the axle while allowing the axle to move up and down for shock absorption. In the initial Spherico design, as shown in Figure 3.22a, the force from the driving wheel passes through the wheel holder where the spring pushes the wheel to the track. The force of friction also acts at the bottom of the wheel. Both the friction and driving force create a torque around the wheel holder. This resulted in the wheel holder bending and Spherico showing poor performance. Recognizing this, in the subsequent design the force of the driving wheel passes to the axle through the acrylic cutout holding the axle of the wheel, as shown in Figure 3.22b. This small change enhanced Spherico's performance significantly.

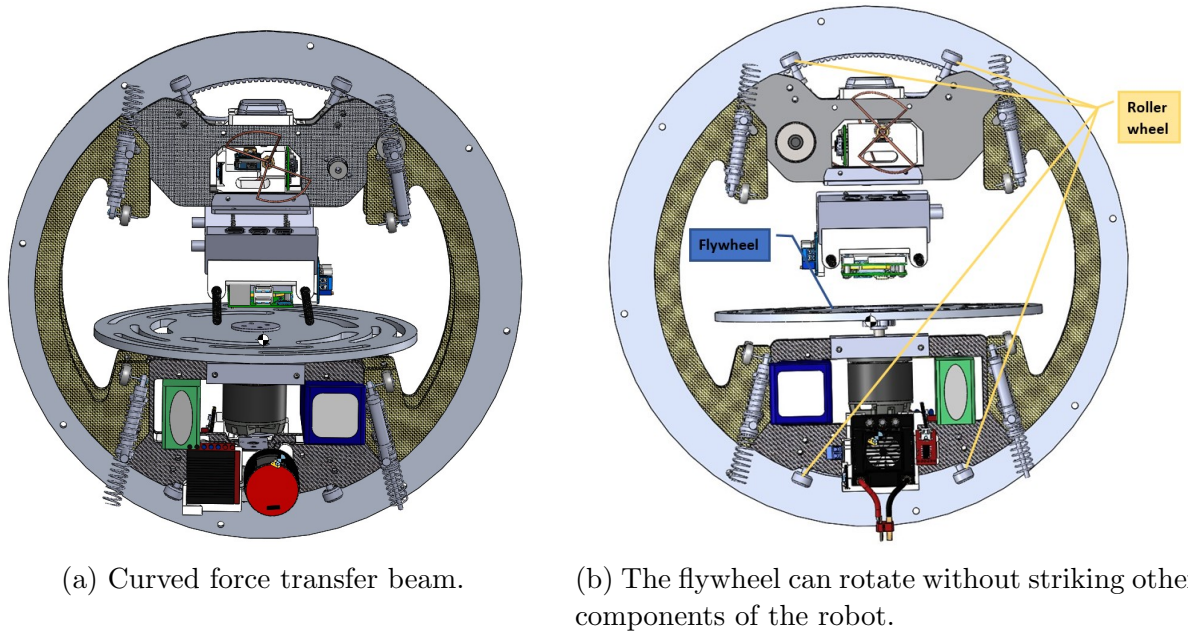


Figure 3.23: Force transfer beam design for DragonBall.

A similar design approach a mechanism where a shock absorbing mechanism is separated from the driving force coming from the drive wheel was followed for the DragonBall. Given Spherico's mass it was decided that four idler rollers would be used, in contrast to the two used in Spherico. Instead of having two tracks, a single track with an aluminium ring was used to support the weight of the system. Internally Spherico used a simple triangle structure to hold the wheels and to transmit force from one wheel to the other. While constructing such a triangle is simple, it restricts the available volume inside the robot. A structure that does not restrict space inside was needed. So instead of straight beams connecting one wheel hub to another, a curved beam that has same internal radius as the track was designed, as shown in Figure 3.23a. This allows the mass shifting mechanism to rotate inside of the robot, as shown in Figure 3.23b.

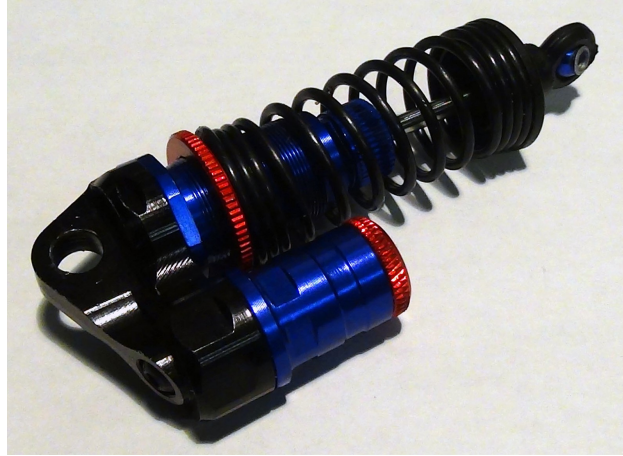


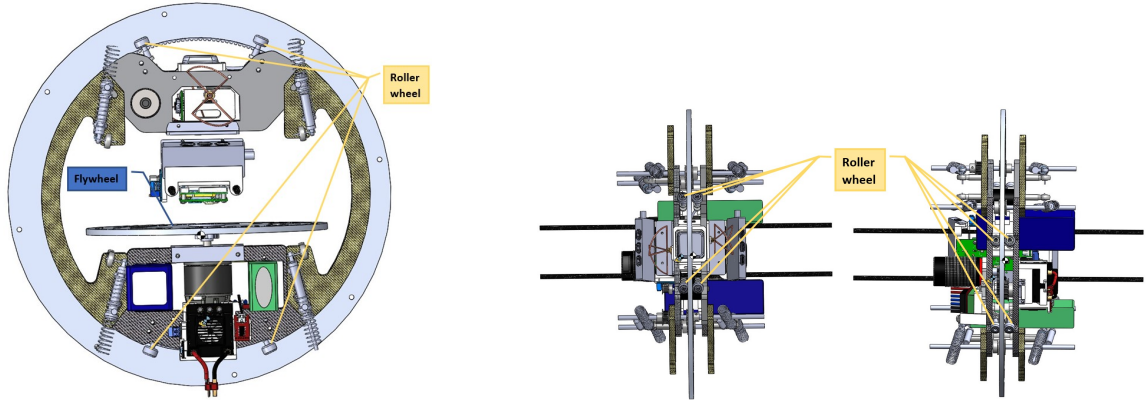
Figure 3.24: RC shock absorber.

3.3.1 Shock absorbers

Shock absorbers are critical for rugged terrain vehicles in order to reduce shock impulses from the ground. Manufacturing custom shock absorbers is difficult and time consuming. Instead of dealing with this, commercial off the shelf RC car shock absorber were used. These are highly sophisticated devices and are not just a simple spring system but a complete oil filled spring dampener system, as shown in Figure 3.24.

3.4 Turning

While testing Spherico it was observed that the strategy of shifting the mass across the mid-line of the vehicle produced a surprisingly small amount of body tilt. The short side of the DragonBall shell weighs 998g and the long side weighs 1.1kg. The weights of the internal parts are also much more than that of Spherico. It was clear that extra mass would be needed to tilt the robot using an approach like Spherico. Many other possibilities were considered and evaluated including adding a lead weight. But more than 3 kg would be needed. Adding



(a) Four roller wheel installed on both side. (b) Top and bottom view of the installed rollers.

Figure 3.25: Rollers are installed in DragonBall to transfer the torque force from the flywheel to the body.

extra weight for tilting results in the robot carrying extra weight, which consumes more power. Clearly a better system was required for turning. Given these concerns, for the DragonBall a system based on the conservation of angular momentum was adopted.

Several spherical robots have been built using systems that use conservation of angular momentum to change the direction of the vehicle, as discussed in Chapter 2. The DragonBall also uses this approach, as shown in Figure 3.25. In order to support the flywheel, DragonBall required rollers mounted on the side of the curved force beams, as shown in Figure 3.25. This allows the curved beams to roll inside the aluminium support plate while absorbing torque from the flywheel motor. The flywheel is designed with slots that allows lead wire to be wrapped onto the flywheel to increase its weight if needed, as shown in Figure 3.26.

Commonly used turning systems for spherical robots are the reaction wheel, momentum wheel and control moment gyroscope (CMG) [1]. The reaction wheel uses the reaction torque from a flywheel when it is accelerated to turn the vehicle. The reaction wheel is normally not rotating and only rotates when the vehicle itself is rotating. This requires a motor with high torque. In the momentum wheel system uses a motor to continuously rotate

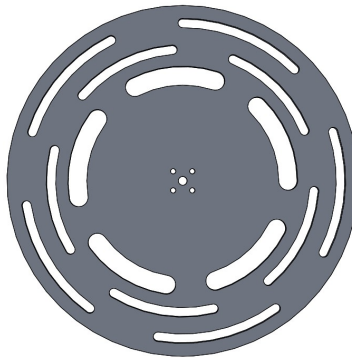


Figure 3.26: The flywheel design for DragonBall was fabricated using waterjet cutting. The slots allow lead wire to be wrapped around the flywheel to increase its mass.

the flywheel at high speed. A braking system is then used to stop the flywheel quickly to exchange momentum [1]. While an additional braking system is required in this system, the motor does not need to have high torque, hence a smaller motor can be used. Control moment gyroscopes (CMG) are widely used in satellites and spacecraft. Unlike the previous two system in a CMG system the flywheel's speed remains constant and instead precession torque generated by the flywheel is harnessed. As discussed in detail in Chapter 4 the DragonBall uses the reaction wheel method.

3.5 Summary

The design of DragonBall required validation of concept, for which a prototype was build. Various important lessons were learned from the prototype called Spherico. Manufacturing Spherico leveraged the use of 3D printing and laser cutting to rapid prototype a spherical robot. A judicious choice in parts design exploited the relative advantages of the available components. The use of OTS parts (such as electromechanical devices) helped to reduce the total manufacturing time. Among the lessons learned from Spherico, the most crucial one was designing the IDU.

A proper method for manufacturing the carbon fiber shell has been developed given the limited resources. Various manufacturing issues such as drapability of carbon fiber fabric, alignment, cutting carbon fiber etc. had to be solved. Also a method for sealing the robot seams was required. Using rubber gasket cutouts solved the issue of sealing the vehicle.

Chapter 4

Electronics system design of the DragonBall

4.1 Introduction

The previous chapter reviewed the mechanical structure of the Dragon Ball and its evolution from Spherico. This chapter discusses the design of the electro-mechanical and communication infrastructure of the vehicle. Although currently the Dragon Ball operates primarily in a teleoperated fashion, in the long term it is intended to operate the vehicle autonomously. For this reason, direct radio control was avoided in the system design.

For any project, it is crucial to identify how many parts can be bought instead of manufacturing them directly. Any engineering student can make their own shoes but the time that will be invested on making a pair of shoes could be spent to make money to buy the shoes and many other things. For larger or more complicated projects using prebuilt items can

reduce the risk of project failure. In terms of the electro-mechanical design of the Dragon Ball this is an important lesson. In recent years hobby electronics has become more and more sophisticated, especially in the quadrotor space. The Dragon Ball is not a quadrotor but many of the parts from a hobbyist quadrotor can be re-purposed here. One reason for using hobbyist parts is because they offer much better pricing options than commercial components and/or making components in house. Another advantage is the vast number of part options available. It should be mentioned that while part prices can be very low, documentation for hobby parts can be missing, inaccurate, or incomplete.

4.2 Electronics

Designing the electronics architecture of a robot is crucial for its operation as performance will rely on properly allocating tasks to each component. Perhaps the most critical decision is the choice of primary on-board computation. Given the desire to provide support for ROS, the obvious choice is some sort of Linux-based machine as ROS support is most mature for that platform. Spherico utilized a Raspberry Pi microcomputer as its primary computational resource. Although this was sufficient for Spherico the limited data bandwidth of the processor was a concern. There was also the desire to be able to perform high performance image processing operations on board the robot. Given these constraints the Dragon Ball uses the NVIDIA TX2 [44] as it's primary computer. Note that the TX2 only performs the minimum amount of low level tasks required with most low level tasks distributed to dedicated micro-controllers. Arduino micro-controllers are used as micro-controllers due to their availability and popularity.

The NVIDIA TX2 is a quad core ARM 64-bit CPU, with a small footprint of 87 x 50

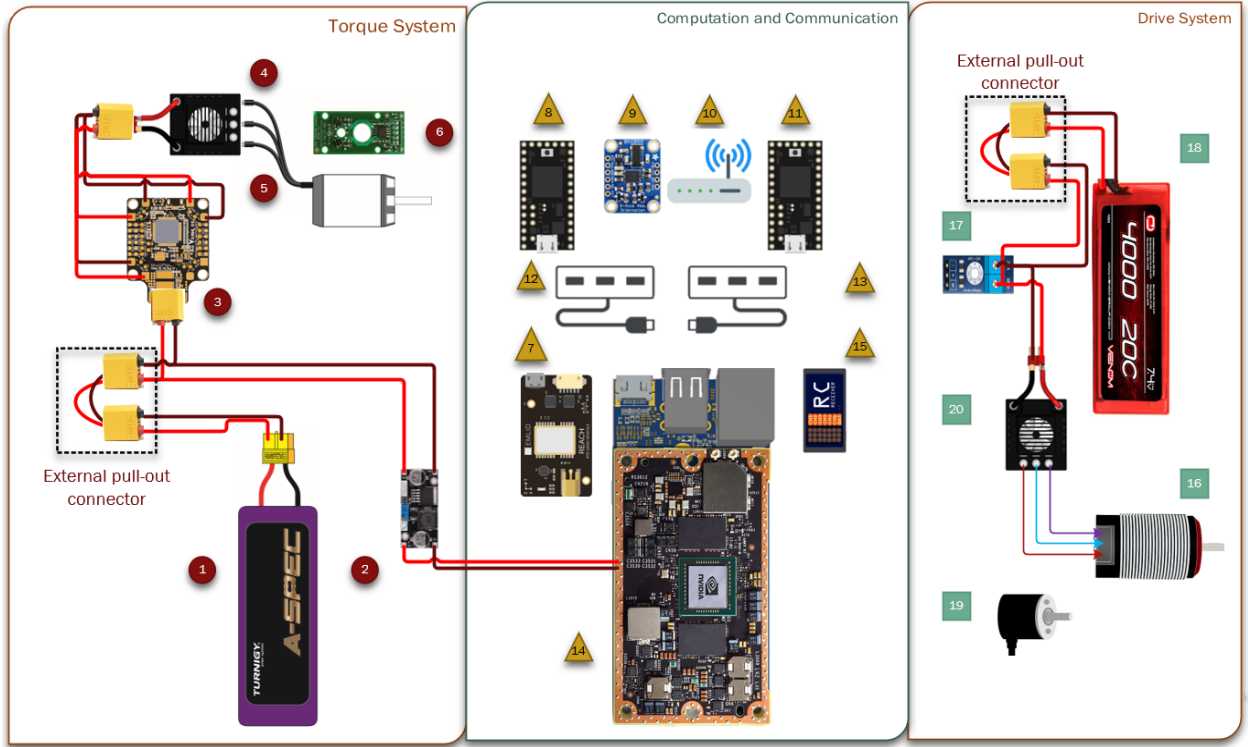


Figure 4.1: System architecture of the Dragon Ball. 1- 5200 mAh Lipo battery. 2- Voltage Regulator. 3- Matek Controller. 4- 90A ESC. 5- Brushless DC motor. 6- Torque Sensor. 7- Reach EMLID RTK GPS. 8- Arduino Teensy. 9- BNO055 IMU. 10- TL-WR802N WiFi Router. 11- Arduino Teensy. 12- USB Hub. 13- USB Hub. 14- TX2 on an Auvidea J120 board. 15- RC Receiver. 16- Brushless DC motor 120A. 17- OSEPP Voltage monitor. 18- 4000mAh LiPo Battery. 19- Positional Encoder. 20- 120A ESC. Area marked as “External pull-out connector”, is a wiring setup such that if the wire is pulled out it will disconnect the battery from the rest of the circuit.

mm and weighs only 85 grams [44]. With such a small footprint and weight, the TX2 is an ideal choice for this applications. A peripheral board was required to connect to other external devices. AUVIDEA offers a carrier board for the NVIDIA TX2 called the J120 board [44]. The J120 board offers one Ethernet, two USB3, mini HDMI out, and a range of other options for connecting to external devices. It has a footprint of 50 x 110 mm, and weighs only 40 grams. One limitation of the AUVIDEA board is that it only offers two USB 3.0 ports. This is not ideal as there are more devices in the robot that requires a USB port. Two powered USB hubs were used to address this issue. One other notable feature of this

hardware that the J120 board offers is the M.2 port, which allows for the installation for a SSD for high speed data collection. The Dragon Ball uses a 120 GB SSD for on-board data storage.

4.2.1 System architecture

The Dragon Ball electronics architecture can be divided into three subsystems as shown in Figure 4.1. In the first section, the torque system incorporates a battery, voltage regulator, Matek Flight Controller, an ESC (electronics speed controller), the BLDC (brushless DC) motor and it's encoder. The second section which supports computation and communication within the vehicle, consists of the TX2 (on a J120 board), a Tp-Link router, the EMLID GPS modem, a RC reciever, two Arduino Teensy micro-controllers, the BNO055 IMU and two powered USB 3.0 hubs to support communication from the TX2 and external USB power. The last section is the drive system, which is made up of the drive motor, a corresponding electronics speed controller (ESC), a voltage reader, the drive encoder, a decoder, and a battery. The reason for dividing the electronics architecture in this manner is due to the power requirements of each system and also due to the specification of the ESCs. Note that the computational component utilizes a separate power regulator system from the other systems. This helps to regulate the power available to the TX2 which can be sensitive to power fluctuations.

The first Arduino (numbered as item 8 in Figure 4.1) has the task of decoding the flywheel speed and simultaneously publishing the decoded data in ROS. The encoder used for this task is the *WW-12 WheelWatcher Encoder* (item 6 from Figure 4.1). It is a low resolution optical encoder. Unlike tasks such as tracking a robot's motion where odometry data is crucial and a high resolution sensor is usually used, here a low resolution sensor can be

used as the profile of the speed is more important than the exact position of the flywheel. This Arduino also takes input from OSEPP voltage monitor (item 16 in Figure 4.1) from the Drive system and publishes the data. Both of the BLDC motors are controlled by this Arduino through the ESC (item 4 and 20 in Figure 4.1). The PID control loops for both motors also runs in this micro-controller.

The second Arduino (item 11 in Figure 4.1), is connected to the BNO055 IMU (item 9 in Figure 4.1) I2C port. BNO055 runs the sensor fusion algorithm from it's internal IMU and magnetometer. The Arduino publishes the result of this process in ROS as shown in Figure 4.16. Data from this IMU is critical for estimating the robot's current pose. The RC receiver is also connected to this micro-controller. It decodes the signal and publishes it in ROS. The Phidget encoder (item 19 from Figure 4.1) is connected to the rotating rim of the robot to determine the distance travelled by Dragon Ball. The encoder data is used for monitoring the state of the gear train requires a high resolution for state estimation of the robot. The micro-controller also decodes the encoders data and publishes in ROS.

The Matek board [45] (item 3 in Figure 4.1) provides IMU and battery voltage data. The IMU from Matek is taken only as a reference, as the quality and calibration of the IMU are not ideal. The main reason for using the Matek board was to take advantage of the battery sensor, piezo buzzer, and power distribution capabilities of the board. The other features of the Matek board, such as IMU and current sensor are added benefits.

4.2.2 Power management

During typical operation of the robot it can be expected that the robot will tend to move forward, rather than turn. Hence the power consumption of the torque system will be low

Power, voltage and current consumption of devices			
Device	Voltage (V)	I_{max} (A)	Power Consumption (W)
LiPo 1			
ESC 120A + Motor	30.00	120.00	3600.00
Arduino Teensy 3.2	5.00	0.4	1.25
Phidget Decoder	5.00	0.20	1.00
WiFi Router	5.00	1.0	1.25
LiPo 2			
ESC HobbyKing + Motor	22.50	90.00	2025.00
Nvidia TX2	10.00	1.20	12.00
Reach Emlid	5.00	0.50	2.50
Arduino Teensy 3.2	5.00	0.4	1.25
Pixpro 4K Right	5.00	1.00	5.00
Pixpro 4K Left	5.00	1.00	5.00

Table 4.1: Dragon Ball devices and their estimated power consumption.

and the power supplied by LiPo 1 (item 1 from Figure 4.1) can be mostly used in powering the TX2. Similarly, it can be expected that the forward running motor will consume more power, so the larger battery, LiPo 2 (item 17 from Figure 4.1), is allocated for that purpose. The estimated maximum power consumption of the devices is given in Table 6.1. It is notable there are a few items missing from this table, as it is hard to obtain good estimates of their power requirements, such as the Matek board. The power rating for the ESCs is estimated at their peak drainage moments. It should be also noted if the torque motor is not used the total current consumption is only 2.3A, a LiPo battery of 5.8Ah should be able to power it for 2 hours. Also the power consumption of drive motor is reduced much lower value, as it is not required to run on full velocity. The target run time for DragonBall is around 40~50 minutes.

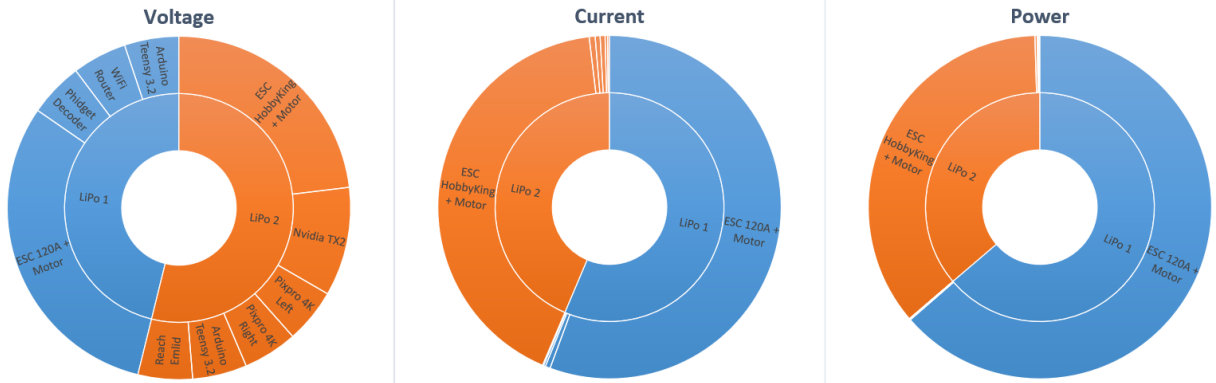


Figure 4.2: Sunburst graph of voltage, current and power consumption of each part of the robot.

LiPo batteries can easily be damaged if operated below their normal operating voltage. This makes it crucial to monitor their power state at all times. In the torque system, the Matek boards supply the state of the battery to the TX2. The Matek board is also equipped with a piezo buzzer which has been set to alarm at 3.2V, as that is the minimum safe operating voltage for LiPo. In the drive system, the OSEPP voltage sensor is connected to the Arduino to monitor the LiPo power. The Arduino will notify the software system if the voltage drops below 3.2V. A separate voltage monitor is connected to the balancer output of the LiPo to monitor and alarm the user by making a high pitched noise through a piezoelectric buzzer should the voltage fall below 3.2V. Figure 4.2 shows the voltage, current and power drainage of each piece of equipment from their designated LiPo. As the case for the previous power circuit it should be noted that the power requirements of a few of the parts is not listed here as no good estimates were available.

Sensors		
Internal sensors	Robot specific	Data collection
Current sensor	Odometer sensor	235° Camera
Voltage sensor	Torque	RTK GPS
Temperature sensor	IMU	

Table 4.2: Sensors in the Dragon Ball

4.3 Sensors

Robots operate in an inherently uncertain world. Even if the world itself is certain and stable, estimation of the state of the world is an uncertain process. For mobile robots such uncertainties arise from poor modeling of the system and the environment, noise in sensors and actuators, and the relation of dynamic processes. Integrating sensors in a robot reduces these uncertainties. Even a simple sensor, such as a current sensor on a motor, can provide clues to the state of the outside world. Proper integration of internal and external sensors is crucial. While some sensor integration is very standard for all robots, other sensors can be robot specific or even task specific. This makes it hard to generalize for all robots classes. Here we consider the integration of sensors specific to a spherical robot performing video data collection in an outdoor environment. Table 4.2 shows how sensors are allocated to specific tasks. Proper integration of sensors requires both proper sensor selection and effective software integration. Selecting the right sensor for the task is crucial for task performance. Other factors in selecting the right sensor includes budget limitation and availability. In the following sections, the sensors in Table 4.2 are described as well as how they are integrated within the robot.

Pixpro 4K camera Resolution		
Resolution	Aspect ratio	Frame rate
2880 x 2880	1:1	30p
2048 x 2048	1:1	30p
1440 x 1440	1:1	60p
1440 x 1440	1:1	60p
1072 x 1072	1:1	60p
1072 x 1072	1:1	30p

Table 4.3: List of available resolution and frame rate for Kodak 4K camera [46]

4.3.1 Cameras

Based on results from Spherico, the Dragon Ball also uses two Pixpro 4K camera to collect data of the surrounding environment. The 235° field of view of these two cameras envelopes the entire surrounding of the robot. The cameras are mounted on the roll axis. The overlap of their field of view happens 15 inches away from the front of the robot, as shown in Figure 4.3. This creates a blind region near the robot.

The Kodak Pixpro 4K camera comes with Linux support. This makes developing a ROS node very convenient and straightforward. The Pixpro 4K camera offers a range of resolution and frame rate options, as shown in Table 4.3. Chapter 5 explains further how data from these cameras are integrated into DragonBall. As the camera geometry is fixed it is straightforward to generate an approximate radial combined map of the cameras as described in the next chapter.

4.3.2 RTK GPS

For pose estimation it is important to have a low noise odometry data [47]. In appropriate environments RTK GPS positioning can be used to accurately track the robots motion. Precise GPS data coupled with odometry, control command and vehicle plant model can be used to produce an accurate estimate of the vehicle's position. While RTK GPS can be effective for estimating the position of the vehicle it also imposes certain restrictions on the vehicle's operational terrains. For getting an ideal RTK GPS lock, it is suggested to operate it away from building, hills, or power electronics system. The DragonBall robot uses the REACH EMLID RTK GPS [48] due to pricing and their small form factor. The device is shown in Figure 4.4.

4.3.3 Odometry

The odometer in a conventional wheeled vehicles is integrated into the wheel. For a spherical robot the odometer needs to be installed between the IDU and the external shell. The *ISC3004 Optical Rotary Encoder* [49] was selected for this task on the Dragon Ball, shown in Figure 4.5a. This sensor outputs two phases of a square wave A and B, where A leads B during clockwise motion and when rotating anticlockwise B leads A. The encoder can handle up to 500,000 rpm with a minimum of 5V input. 500,000 rpm corresponds to 18775.63 m/s of ground speed which should be more than sufficient for the DragonBall robot.

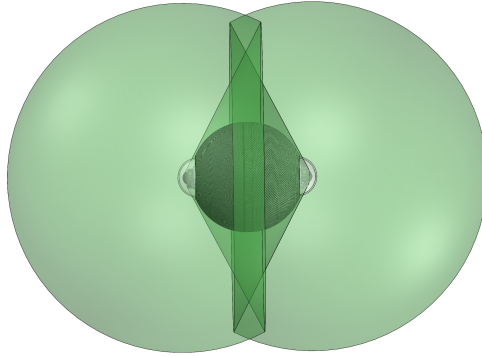


Figure 4.3: Field of view for the Dragon Ball. The two cameras have an overlapping field of view approximately 15 inches from the DragonBall external shell

4.3.4 Torque sensor

The controller driving the flywheel requires the speed profile of the flywheel more than the position of the flywheel, in order to be used as a CMG. While the *ISC3004 Encoder* offer great resolution and accuracy they are expensive and the flywheel does not require precise position data. The *WW-12* [50] is an inexpensive encoder from Nubotics that offers high resolution and high speed at the same time, as shown in Figure 4.5b. It relies on an optical disk, as shown in Figure 4.5c to monitor the axis of rotation. The disk needs to be installed on the wheel and an encoder board needs to be mounted right under that. The board has two photo sensors and a LS7084 quadrature decoder IC. The count data is send through the I2C bus to an additional micro-controller, item 8 in Figure 4.1.

4.3.5 IMU

For pose estimation of a mobile robot an IMU plays a significant role as it can be used to monitor state or change of state. The *BNO055* [51] from Adafruit offers a huge range of features that are vital and desirable even at a low price. It couples a 9-DOF IMU sen-

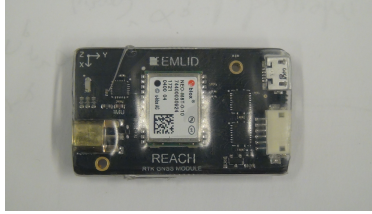


Figure 4.4: REACH EMLID GPS.

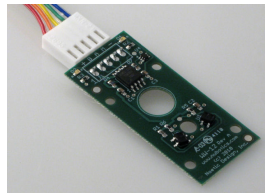
sor together with temperature sensor, compass, and an ARM Cortex-M0 based processor. The processor is used for running an on-board sensor fusion algorithm. Connection to the *BNO055* is through a USB cable. This makes it easy to integrate the device to the Dragon Ball.

4.3.6 Current and voltage

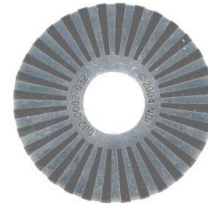
The Dragon Ball has two battery sets, electronic speed controllers (ESC) systems and motors. Both battery sets are separated to simplify the design. Sensing battery voltage is very critical not only for the operation of the robot but also due to the fact that LiPo batteries can be damaged permanently if operated above a certain power drainage. On the DragonBall, each battery set has its own power monitoring system. One of the sets uses the *OSEPP* voltage sensor (Figure 4.7a) together with an Arduino to read the battery voltage. This set also has an external battery monitoring system (Figure 4.7b). It comes with a buzzer which notifies the user if the voltage falls under a certain level. The second set uses the *Matek FLIGHT CONTROLLER F405-CTR* to monitor the battery voltage. The *F405-CTR* also incorporates a buzzer for low voltage notification. The voltage and current data can be obtained from the *F405-CTR* from a USB connector.



(a) The ISC3004 Optical Rotary Encoder. Images reprinted from Phidgets/Encoders.



(b) WW-12 WheelWatcher Encoder. Images reprinted from Nubotics/Encoders.



(c) Optical disk for torque encoder.

Figure 4.5: Optical sensors in the Dragon Ball

4.3.7 Temperature sensor

While the role of temperature inside the robot is not expected to be that significant, excessive temperature within the robot may cause the main computer to shut down. The Dragon Ball uses a *Jetson TX2* as the main computer, the carrier board used for this is the *Auvideo J120* board. The board has an integrated temperature sensor which provides straightforward monitoring that the *Jetson* is running within the prescribed temperature range.

4.4 Motor selection

BLDC motors (brushless DC motors) are one of the most popular motors for hobby RC applications. There are several advantages of using a BLDC motor over a conventional brushed DC motor. The main advantage of brushless motor is related to damage to the commutator over time. In a conventional brushed DC motor the commutator is in contact with the motor as it rotates and this wears off over time. As a consequence other problem

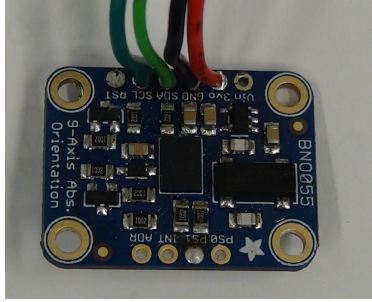


Figure 4.6: Adafruit BNO055 IMU.

arise over time including metal dust, sparks, and decrease in motor efficiency. Brushless motors offer a solution to all those issues by not having a commutator in the motor. Instead, a separate control circuit drives phases of electric current through the coils to drive the rotor which holds the permanent magnets. An electronic speed controller (ESC) controls the speed and torque of the motor.

4.4.1 Drive motor

The current fastest human running speed is at 12.5 m/s over a 100 m race within 9.58 seconds. The average velocity is much lower, that is 10.43 m/s and average acceleration is only 1.089 m/s². Performance of a typical robotics research is less than this. As it would be ideal if a person can run up to the robot if something goes wrong, it was decided that a top speed of 15m/s and acceleration of 11 m/s² would be sufficient for the Dragon Ball. Converting the velocity to inches gives 590 in/s. Given the 16 inch diameter of the dragon ball an angular velocity ω required is 73.75 rad/s. The gear train used in Dragon Ball was determined through a long process of designing and availability constrain. The resulting design had a gear ratio of 15, which reduces the angular velocity. Therefore the motor should be able to provide 15 x 73.75 rad/s, i.e 1106.25 rad/s. Many motors and batteries can meet this requirement hence more constraints for selection were needed. Another constraint is

the torque requirement for Dragon Ball. Although there are many advantages for using a hobbyist motor, the documentation is usually incomplete and many parameters are not given. One of such missing parameter is the max torque, which is very important for the design of this robot. When the torque parameter is not given in the datasheet of the motor, it is possible to calculate it in other ways. If a speed controller (ESC) for the motor and a battery are selected then the maximum torque can be calculated. Hobby electric motors typically provide a KV value which specifies how many turns a motor will turn per volt. For example, if the KV value is 200, that means the motor will turn 200 RPM for one voltage. From the battery specification the maximum voltage can be obtained and also the maximum possible current. Therefore the maximum possible power can be calculated, i.e. current x voltage. The power of a rotating body can be calculated as;

$$P = VI = \tau\omega. \quad (4.1)$$

Here V is maximum voltage, I is the maximum current, τ is the maximum torque and ω is the angular velocity. Rearranging gives

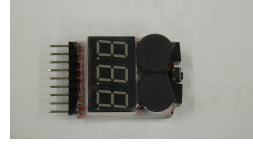
$$\tau = \frac{VI}{\omega}. \quad (4.2)$$

One notable thing here is that the torque calculated is not the final torque value as the motor turns the track through a gear and tack system, which results in higher torque and reduced angular velocity. In order to select a motor, ESC, and battery, with the right torque, it is necessary to determine the torque requirement.

In order for the body to accelerate forward, the total torque from the motors must overcome the moment of inertia of the body and the torque from the friction force from



(a) OSEPP Voltage Sensor Module. Image reprinted from OSEPP.



(b) LiPo voltage indicator and monitor. It has builtin peizo electric buzzer to notify if the voltage in any cell is below the threshold value, i.e. 3.2V.

Figure 4.7: DragonBall voltage sensors.

the floor $\tau_{motor} \geq I_{shell}\alpha_{shell} + \tau_{friction}$. Also, the torque from the motor must be greater or equal to the torque required to rotate the offset mass, hence $\tau_{motor} = I_{pendulum}\alpha_{pendulum}$. The weight of the mass offset includes the mass of the motor, the ESC and the battery. The average weight of each part is used to approximate the weight of the offset mass, calculated to be 4.00 kg. The moment of inertia of a pendulum is mr^2 . Substituting $a_{linear} = r\alpha_{motor}$ in the previous Equation, τ_{motor} approximately equates to 8 N.m . Due to the increase in torque by gear ratio the required torque from the motor is $8/15 = 0.533$ N.m.

These parameters were used to select a set of ESC, motor, and battery. A list of available of motors with corresponding ESC and battery was made. The final selection was made using criteria including price, availability and product review. The resulting BLDC motor was a Turnigy XK-4074 2000KV Brushless motor [52] (Figure 4.8b), a 120Amp ESC [53] and a 4000mAh battery [54] of 65C continuous discharge.

4.4.2 Turning motor

There are three turning strategies that can be implemented in the DragonBall; reaction wheel, momentum wheel, and the CMG system. In order to select the turning motor only the requirement of the reaction wheel was considered. Equation 4.3 relates the motor torque, the



(a) PROPDRIVE v2 5060 270KV Brushless Outrunner Motor. Image reprinted from Hobbyking.



(b) Turnigy XK-4074 2000KV Brushless Inrunner BLDC motor. Image reprinted from Hobbyking.

Figure 4.8: DragonBall motors.

flywheel and the motor. A desired maximum angular speed is required in order to determine the required torque. It was decided that the robot should take a maximum of three seconds to make a full revolution. This corresponds to an angular velocity ω of 2.1 rad/s and an angular acceleration of 0.7 rad/s. To simplify the model we consider friction to be zero, the moment of inertia to be $\frac{1}{2}mr^2$ and the total weight of the body to be 6.23 kg. Then from Equation 4.3, $\tau_{motor} = 0.09$ N.m. Here the angular velocity can not be used as another search criteria for an appropriate motor, as the amount of torque is the determining factor. For selecting this motor price and size was considered as a second and third criteria. Also the torque is obtained without accounting for friction which changes from surface to surface, so increasing the required torque by a factor of 5 was necessary to ensure proper performance. The selected motor was *Propdrive v2 5060* [55] as shown in (Figure 4.8a), it has a torque value of ~ 3.1 N.m, which is much higher than this requirement.

$$\tau_{motor} = I_{body}\alpha_{body} + \tau_{friction}. \quad (4.3)$$



(a) Basher 4000mAh 65C discharge battery.
Image reprinted from Hobbyking.



(b) Zippy 5800mAh 60C discharge battery.
Image reprinted from Hobbyking.

Figure 4.9: DragonBall batteries.

4.5 Battery safety

Prior to dealing with control mechanisms that enable the robot to move it is important to recognize that the control systems are also required to prevent damage to the mechanical parts of the robot or to its LiPo batteries. Both of the motors are controlled by the first Arduino (item number 8 in Figure 4.1). Safety measures ensure that the motors do not drain the LiPo batteries or actuate the motors in a manner that might damage them or other mechanical parts within the robot. Extreme power drainage from either battery could temporarily shut down the ESCs. This would be extremely bad for the torque system as the battery is shared between the computational and torque systems. Shutting down the computational system would require restarting the entire system including re-arming the motors. Also if the drive motor accelerates too fast the drive track might be damaged.

The two batteries within the Dragon Ball are shown in Figure 4.9. The Basher battery [54], shown in Figure 4.9a, has a 65C discharge rate. A rating of 65C means that it will be able to discharge 65 times the current rating of the battery, i.e. $4000mA \times 65 = 260A$. Similarly for the Zippy battery [56], shown in Figure 4.9b has a discharge rate of 60C. This “C” rating of battery allows system to drain higher current than it’s normal operating range for less than 10 seconds. The discharge rating is critical for transient power consumption,

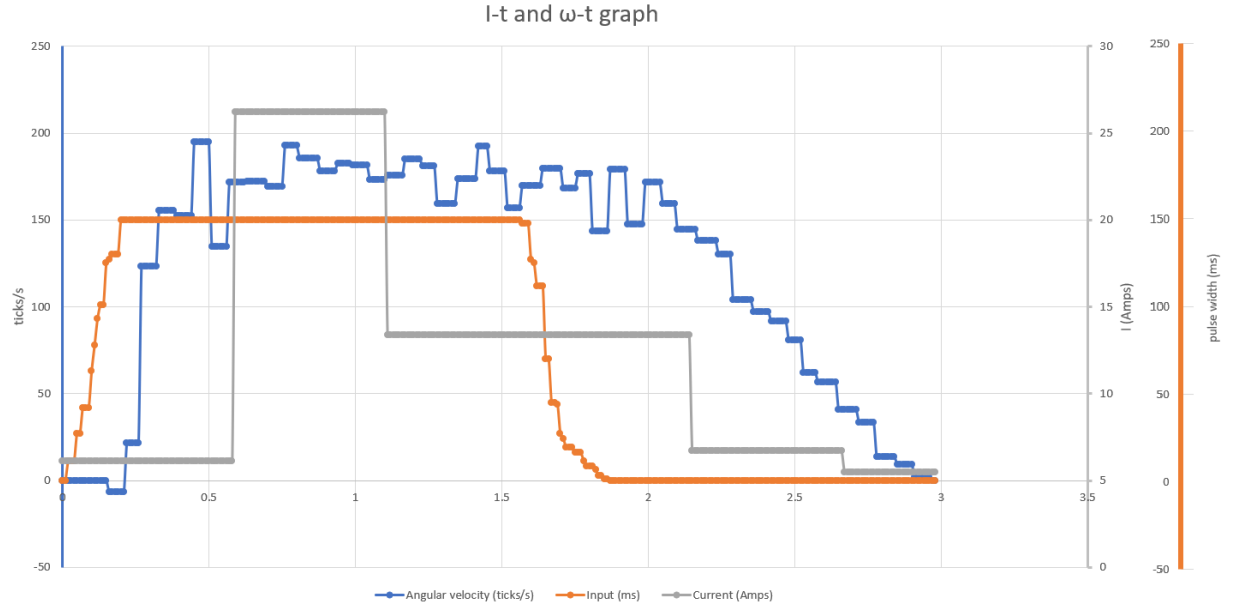


Figure 4.10: Torque motor current drain test.

especially when using motors. The current drawn by a motor when starting from a static state is much higher than that of it when just increasing speed from a dynamic state. This current draw is called *inrush* current [57]. The inrush current for the drive motor is less of a concern than that of torque motor. As can be seen from Table 6.1, the power from the Basher battery only powers the drive motor whereas the Zippy battery powers both the turning motor and the computer. The turning motor needs to be able accelerate rapidly which means that it is critical to ensure that the inrush current of this motor is less than the maximum current discharge of the battery together with the maximum power drainage of the TX2.

In order to understand the inrush current drainage of the turning motor, a test setup was made using the Matek board to measure and log the current drainage of the turning motor while reading the odometry data from the *WW-12* encoder. The motor was controlled manually with a joystick to the full power and then stopped. The data from the *WW-12*

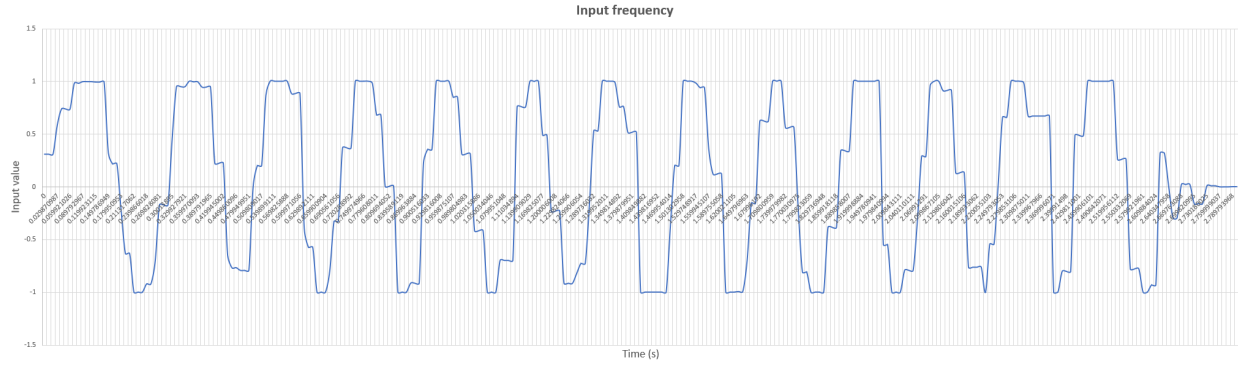


Figure 4.11: LiPo voltage indicator and monitor. It has builtin peizo electric buzzer to notify if the voltage in any cell is below the threshold value, i.e. 3.2V.

was read by the Arduino (item 8 in Figure 4.1), which calculates the instantaneous velocity and publishes it in ROS. Figure 4.10 plots the instantaneous velocity against time in blue. The current reading data from the Matek board is also plotted in grey. The orange line is the input from the joystick which controls PWM in microseconds. The current drainage increases with the increase in velocity value, but when the velocity value stabilizes current drainage reduces. It is clear that the amount of current drainage is related to the change in velocity (the acceleration value). It should also be noted that there is a lag between the current drainage and instantaneous angular velocity. This might be due to sampling bias in the measurement blocks. A maximum current drainage of 27.2 Amps was recorded during this experiment.

The max current drainage is well below the maximum current drainage rating of the Zippy battery, i.e. $60 \times 5.8 \text{ Amps} = 348 \text{ Amps}$. The ESC's are designed to be operated through commands from an analog joystick. The joystick is operated by human, so the rate of change in input is limited by the operators hand motion speed. The rate of change from computer command can be much higher, as computer commands can go from 0 to 1 in nanoseconds. The DragonBall is designed to operate in a fully autonomous manner in the future, where the control commands will come from a computer. While the battery will

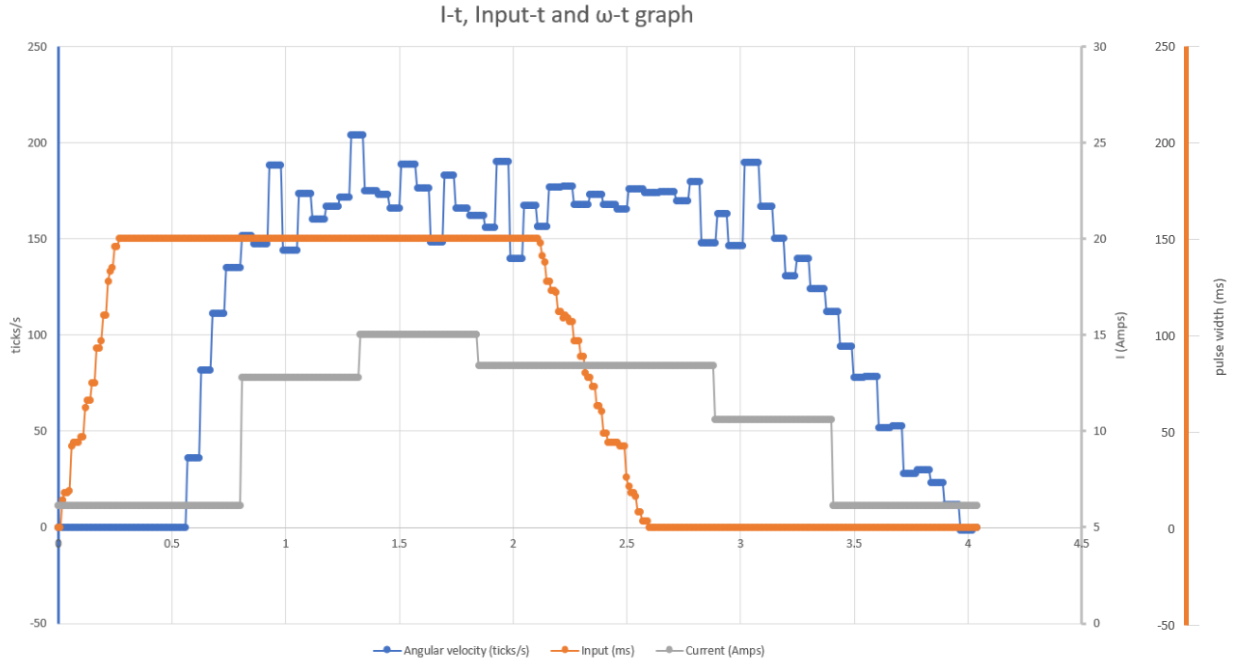


Figure 4.12: Torque motor current drain test after implementing filter.

be able to provide such power for less than 10 seconds the ESC's are known to burn out in such instances. It is important to ensure that this drainage is limited, so that the ESC is not damaged.

The main controller could be programmed to not change control command faster than a given rate. In order to understand how fast a human could send motion commands to the system a setup was built to receive joystick commands from a user and log them. Three participants, A, B and C were asked move one joystick as fast as they can and their data was recorded. Figure 4.11 shows participant A's input log plotted against time. From the graph the participants average time period between max to min command was calculated. From the time period average frequency can be determined.

A comparison graph of all the user and their average is shown in Figure 4.13. The lowest

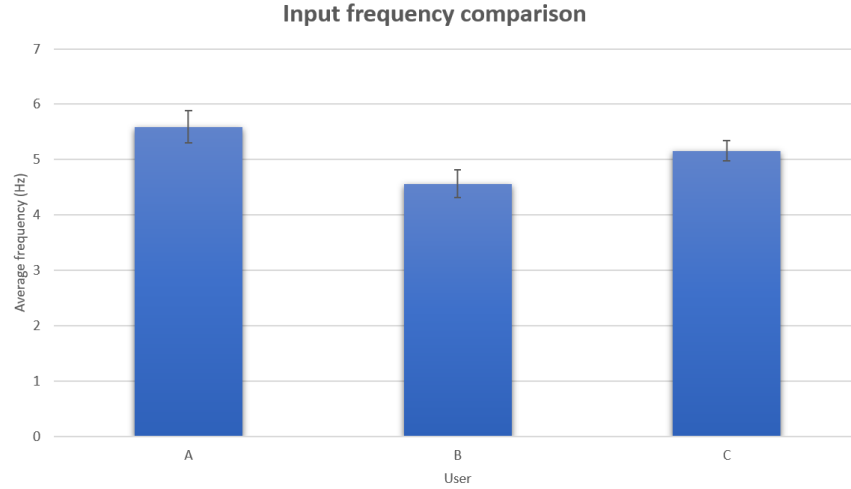


Figure 4.13: Three user input comparison.

frequency, from Figure 4.13 is 4.42Hz. A simple way to ensure that the ESC never receives frequency higher than 4.42Hz is by implementing a low pass filter to its input. A Butterworth filter with a cutoff frequency of 4.4Hz was applied to the input line of this ESC to ensure it's safety. High speed can be obtained but acceleration is limited. The Arduino ensures that if the main computer is not properly communicating through ROS then both motors will be shut down to ensure the safety of the robot.

4.6 Communication

Proper control of the robot by a human operator requires an effective human-robot interface and control system design. Also the perception of robot's environment is critical for the user to determine whether the next move is plausible. As mentioned in Chapter 1, common issues that are faced with HRI includes communication delay, sensor integration issues, and limited data bandwidth. Another challenge commonly faced with teleoperating a mobile robot is providing effective communication of the robot's orientation relative to the user. A

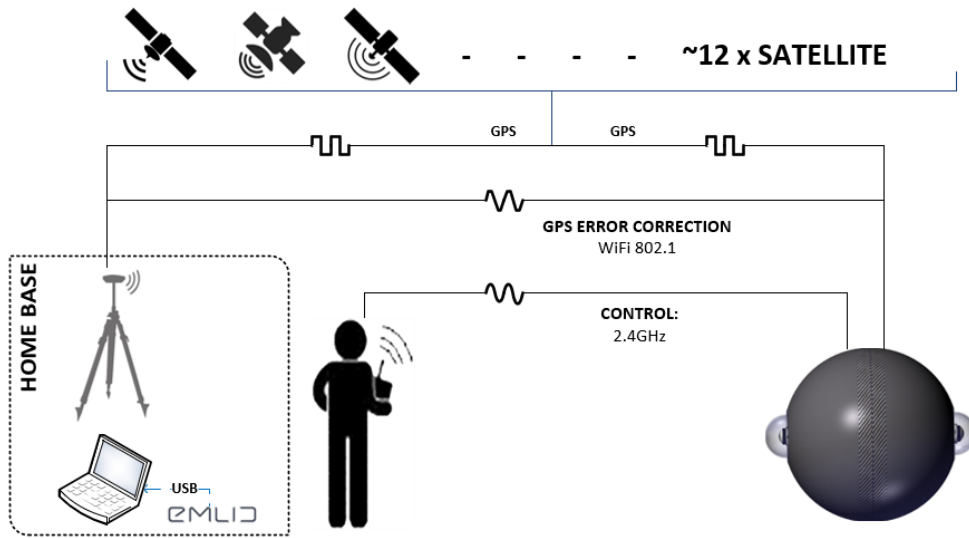


Figure 4.14: Communication architecture. Video, GPS and ROS data is received through WiFi. Control data is send via RC remote control.

typical example of such situation is when flying an quadcopter, if a quadcopter starts to yaw and the user losses the sight of the front end of the quadcopter, pressing the joystick button forward may not result in forward motion of the quadcopter as defined by the user. DragonBall, when far from user, can be difficult to keep an eye on to determine the front end of the vehicle (the robot being spherical does not help).

Many of the issues associated with teleoperation communication delay can be reduced if a coordinated teleoperation system is implemented. All control loops reside inside the robot's computer, therefore input from external sources are only needed when a change in state is required. This eliminates the risk of unwanted behaviour from the robot due to communication delays. Also, all the sensors for the safety of the robot, such as current or voltage is integrated within the control system of the robot. This means that the operator does not require to monitor such warnings as the robot will react to them by itself.

Before choosing the method of communication it is important to understand the terrain on which the robot will be tested. The DragonBall is primarily intended to be used outdoor in



(a) Flysky FS-i6X 2.4GHz remote control and receiver. Image reprinted from Amazon. (b) RC receiver modded to connect an antenna further away from the body.

Figure 4.15: RC transmitter and receiver and the modified receiver.

land-based experiments. For the scope of this experiment, it can be expected that the robot will operate within a WiFi zone. If long range is required than only control data will be transmitted. Keeping this strategy in mind communication was divided into two parts; one via WiFi and the other via a RC remote. Figure 4.14 shows how communication is provided to the DragonBall. In order to reduce data bandwidth usage video transmitted from the robot is scaled to 640 x 480. Only the ROI is transmitted through WiFi communication. Video teleoperation is further explained in the next Chapter. Control data is sent through the RC remote control to the Dragon Ball.

Carbon fiber is known for its RF shielding. While it is lighter and stiffer than glass-fiber, installing communication devices inside is challenging. For both the WiFi and the RC remote modification to the unit was needed to overcome this situation. External antennas were soldered to the internal circuit board with coaxial cable to extend them. Extending the antennas allowed them to be placed on each side of the windows on the longitudinal axis of the robot. Having two antennas on both sides ensures neither will be shadowed by the body of the robot. A similar modification was needed for the WiFi router inside the robot to have two antennas on both sides of the robot.

4.7 ROS

ROS provides an excellent means to control various system on various devices in a synchronized manner. ROS also allows the user to divide tasks easily between devices, with the capability to monitor them in real time. Figure 4.16 shows how the tasks are divided and allocated to each system for the Dragon Ball. The DragonBall uses a simple ROS architecture for control. Eight nodes are created; one for the IMU, Arduino, odometer, controller, joystick, Matek board (mavros), video publisher, and user view. The system controller node takes information from almost all nodes. The video publisher only takes input from the joystick to control the virtual pan and tilt system as described in Chapter 5.

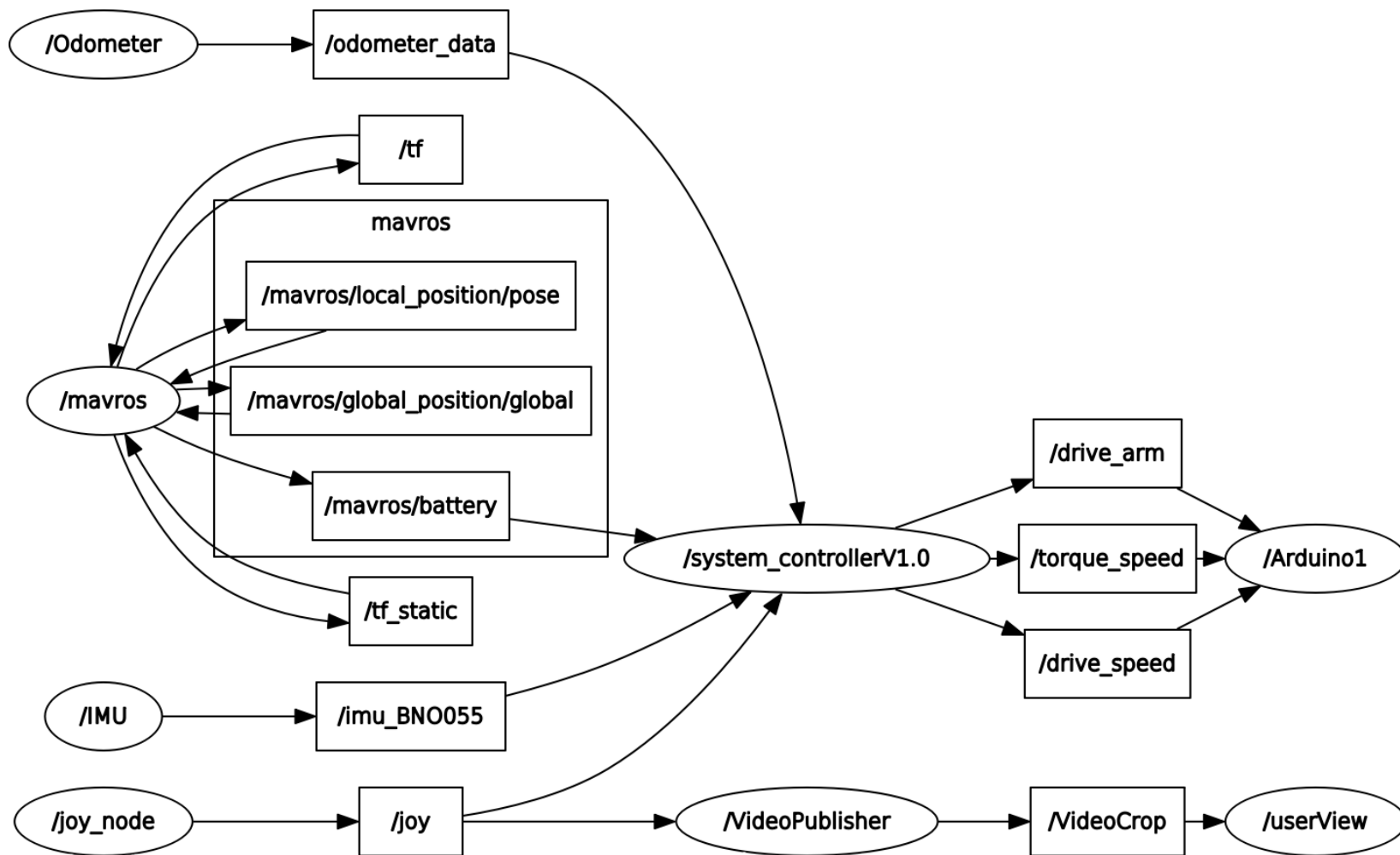


Figure 4.16: ROS architecture of the Dragon Ball.

4.8 Summary

This chapter describes the low-level hardware integration of the robot. The low-level control design of DragonBall requires the proper design in the integration of sensors to the robot. The control of the robot depends on how well the interface is designed and also on how well the control loops are designed. The integration of sensors also means the proper design of power and communication. Power consumption is divided into two sections, the LiPo 1 and 2. Due to high power drainage the LiPo 2 powers the drive system only. The other LiPo powers the rest of the system. Sensors are categorized between internal, robot specific and data collection sensors. All internal sensors are in the integrated within the control system of the robot so that the operator does not need to monitor them all the time.

Chapter 5

Control and visual teleoperation of the DragonBall

5.1 Introduction

The design of the control system for a robot defines its performance. Control systems can be divided into two general approaches; open-loop, and closed-loop control. The main difference between these approaches is that open-loop systems do not employ feedback while closed-loop systems do. In an open-loop design processes are connected to one another to produce the desired output directly without incorporating feedback to the system. Whereas in a closed-loop system the feedback is compared to the desired output so that necessary modifications can be made to the system to account for any error in the output. The DragonBall robot has two main control systems; drive and spot turning. The spot turning system is an open-loop system that enables the robot to turn while it is stationary. The drive system is a closed-

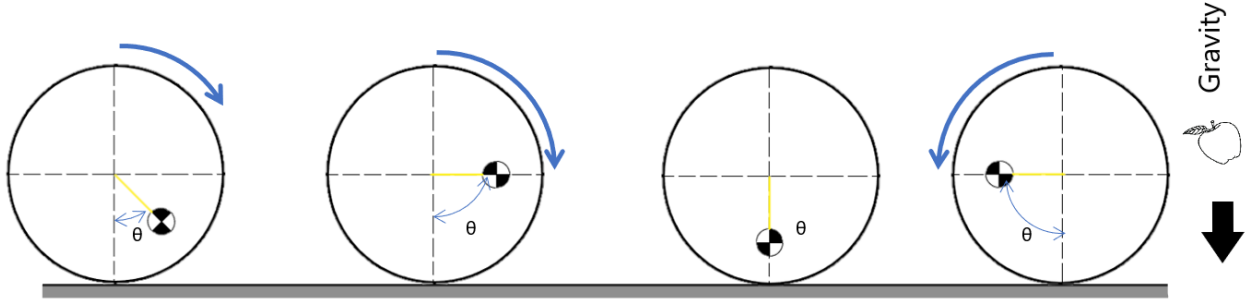
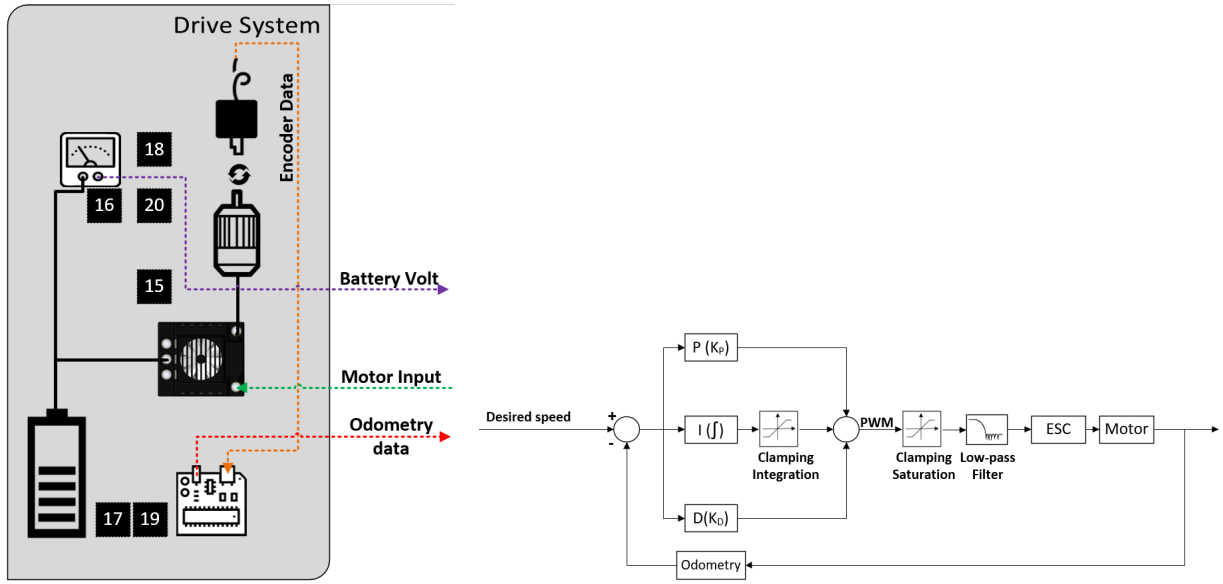


Figure 5.1: Effect of IDU angle to the drive direction. From left to right, increasing θ in positive direction (clockwise) the torque generated makes the robot move in forward direction. When θ is 90° maximum torque is generated. At 0° there is no torque and the vehicle is steady. If θ is negative the vehicle rolls backwards.

loop system. The drive system allows the robot to drive forward or backward. The following sections describe how each of these control systems were designed and implemented.

5.2 Control

Designing the control systems of the DragonBall required proper understanding of the hardware involved. The motor and the torque motor together with their speed controllers were fully tested before designing the control system. Certain important characteristics of the hardware were noted and these observations were critical in the design of the control systems. For example, the drive motor requires at least 8 seconds of stop before reversing direction. Also unlike a conventional servo motor the drive motor does not rotate within the range of 1460 to 1540 milliseconds pulse width modulation. The range of pulse width the torque motor requires is between 1350 to 1650 milliseconds. Also, as discussed in Chapter 4, the acceleration phase for the torque motor must be limited to a maximum of 10 seconds to satisfy battery power drainage limits. The control system also needs to reside in the Arduino to ensure smooth and safe operation, as discussed in Chapter 4. With these and



(a) Symbolic diagram of the Drive system.

(b) Closed-loop control system for driving the robot.

Figure 5.2: The drive system. (a) Electro-mechanical parts of the drive system. (b) Control system diagram of the drive system.

other constraints in mind, the design of the control system of the DragonBall is described in the following sections.

5.2.1 Drive system theory

Chapter 2 described torque generation methods for a spherical robot that operates by controlling the C.G. of the IDU with respect to the gravity line. Figure 5.1 illustrates the effect of the robot's C.G. (defined by θ) on the robot rolling direction. From Chapter 2, the driving torque is proportional to the θ angle. Starting at zero torque at 0° and going to maximum torque at $\pm 90^\circ$. Controlling this angle with respect to the gravity line essentially controls the throttle of the robot. A PID controller maintains the torque that drives the robot. Note that $|\theta| > 90^\circ$ reduces the driving force and eventually causes the robot to flip direction at

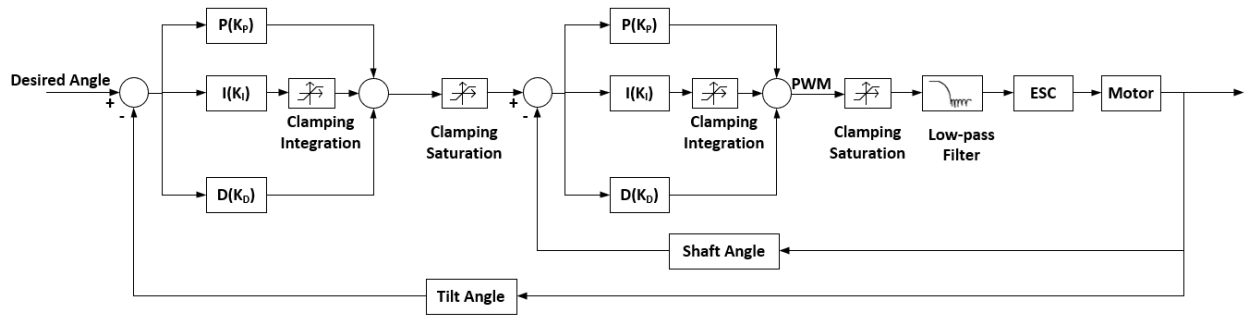


Figure 5.3: Cascaded PID control design for DragonBall.

$|\theta| = 180^\circ$. The θ angle determines how much torque will be provided by the IDU. Controlling this angle is essentially controlling the rolling acceleration of the robot.

5.2.2 Drive system implementation

The components involved in the drive system are shown in Figure 5.2a and the PID implementation is shown in Figure 5.2b. Given the user's desired speed for the IDU the *Controller* process passes this input to the PID. From the PID section, the ESC receives the PWM signal to drive the motor. Feedback from the ISC3004 Encoder provides the system with angular velocity data. A low-pass filter and clamp for saturation is implemented before the ESC to ensure the safety of the ESC. The saturation clamp clamps down the signal to the operational frequency range of the ESC. The low-pass filter smooths the input signal.

A cascaded PID control loop [58] is used to account for both the speed of the motor as well as the angle of the IDU. The inner block determines the motor's rotational speed and the outer block determines the mass orientation with respect to the gravity line. In the outer loop feedback comes from the BNO055 IMU. Figure 5.3 shows the entire cascaded PID block for controlling the drive system of the robot. The user inputs the desired angle which

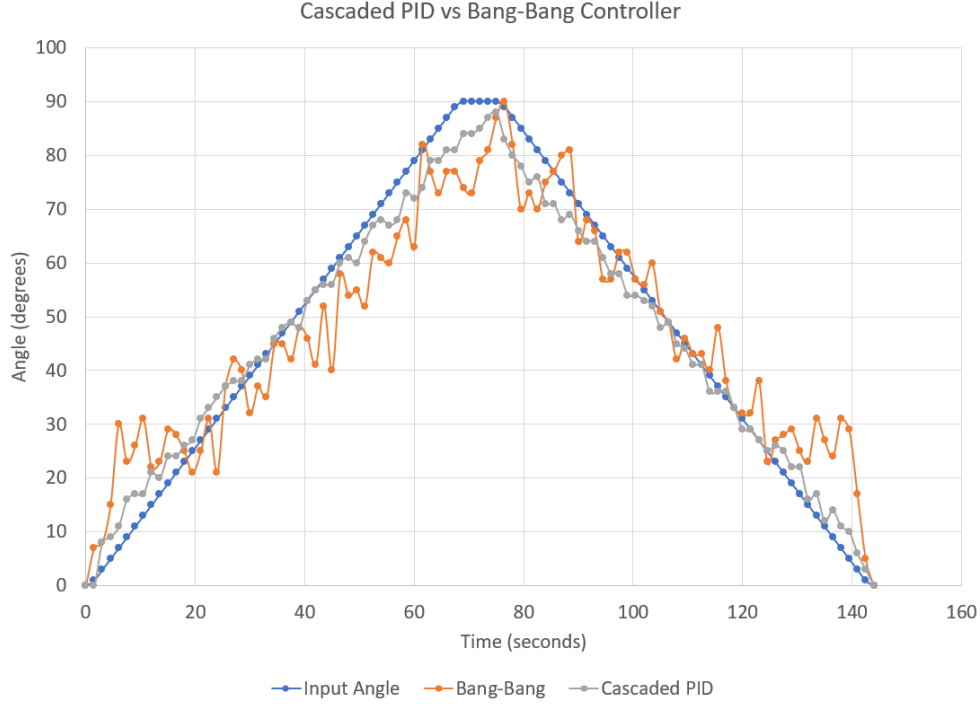
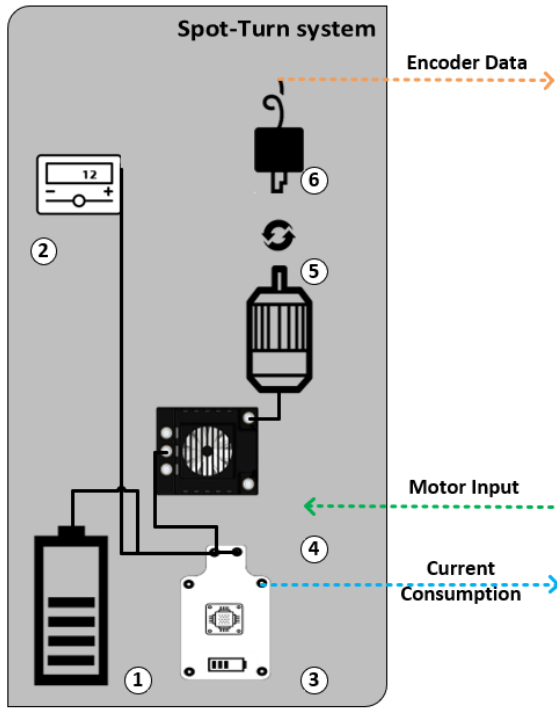


Figure 5.4: Cascaded PID control vs Bang-Bang control of DragonBall. The Bang-Bang controller shuts off the motor if the angle is higher than the input.

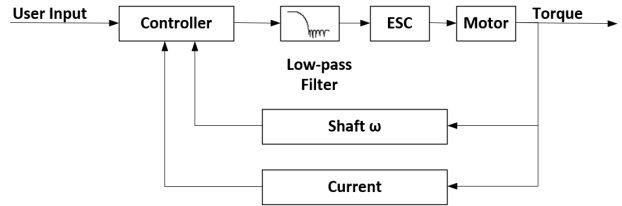
controls the outer loop. Then the outer loop sends the desired speed input to the inner loop. This cascaded control ensures the safety of the motor and inner ring. To tune the inner loop a testing rig was built. After tuning the inner loop the outer loop was tuned by driving the robot. Figure 5.4 shows a plot of user input and actual angle of the IDU with respect to the gravity line. For comparison a Bang-Bang controller graph is also shown in the same figure.

5.2.3 Spot-Turning system

When the robot is not moving forward turning can be performed using counter torque from the flywheel. A simple way of controlling the torque is to induce controlled change in the flywheel's angular velocity, $\tau = I\alpha$ where $\alpha = (\omega_{final} - \omega_{initial})/\Delta time$. By controlling the input of the ESC we have control over the velocity of the motor. A simple “bang-bang” type



(a) The Torque system.

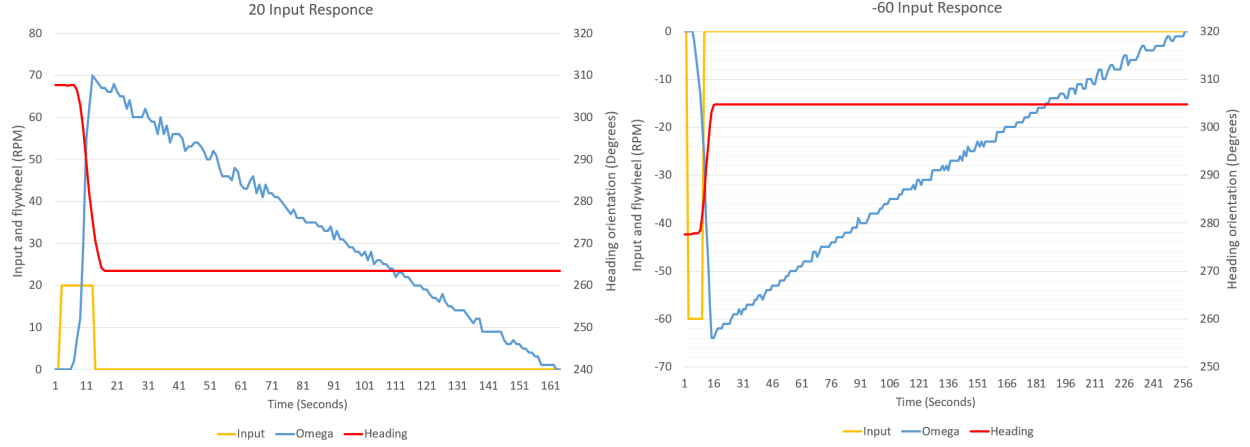


(b) Open-loop control system for turning the robot.

Figure 5.5: System control. (a) The parts in spot-turning system. (b) Control system diagram of the turning system.

of controller is used to control the orientation of the robot. The basic idea is to accelerate the flywheel using maximum power from the motor and then to slow down the wheel rotation at a slower rate as shown in Figure 5.6a. This creates a torque pulse which turns the robot. Repeating this process is used to rotate the robot to a desired amount.

Understanding the motors characteristics is important for designing the system. The BLDC motor with the flywheel accelerates slower than without the flywheel. Also, the motor has a lower acceleration in one direction than in the other. This is due to the fact that these motors are normally used for RC cars and most RC cars designed to go slower backwards. As shown in Figure 5.6, when a target speed of 20 RPM is set for the flywheel, in the clockwise direction, it overshoots and the body turns 45° . Whereas when rotating



(a) Input of 20 RPM causes the robot to rotate $\approx 45^\circ$.

(b) Input of -60 RPM causes the robot to rotate $\approx -27^\circ$.

Figure 5.6: Spot turning example of inputs 20 RPM and -60 RPM. Here “Input” is the user input, “Omega” is the angular velocity of the flywheel and “Heading” is the orientation of DragonBall. (a) A target velocity of 20 RPM is set on the microcontroller, shown as the yellow line. The resulting angular velocity of the flywheel and the robot is shown in blue and red respectively. (b) Similarly the effect of inducing a -60 RPM to the microcontroller.

the flywheel in the counter clockwise direction the acceleration is slower and the body only turns 100° . In order to account for this, the user input from joystick is treated differently for each direction as explained in the following sections.

5.2.4 Spot-Turning implementation

There are five processes related to spot turning as shown in Figure 5.5b. The first process, *Controller* takes input from the user, the flywheel encoder and the current drainage of the ESC to determine the number of possible acceleration and deceleration pulses. The controller monitors the current drainage value to ensure that it stays within a 10 second window. Each torque pulse is generated by sending the max value to the ESC while noticing the rise in flywheel speed, as soon as the speed reaches the target value the power is cut off. The

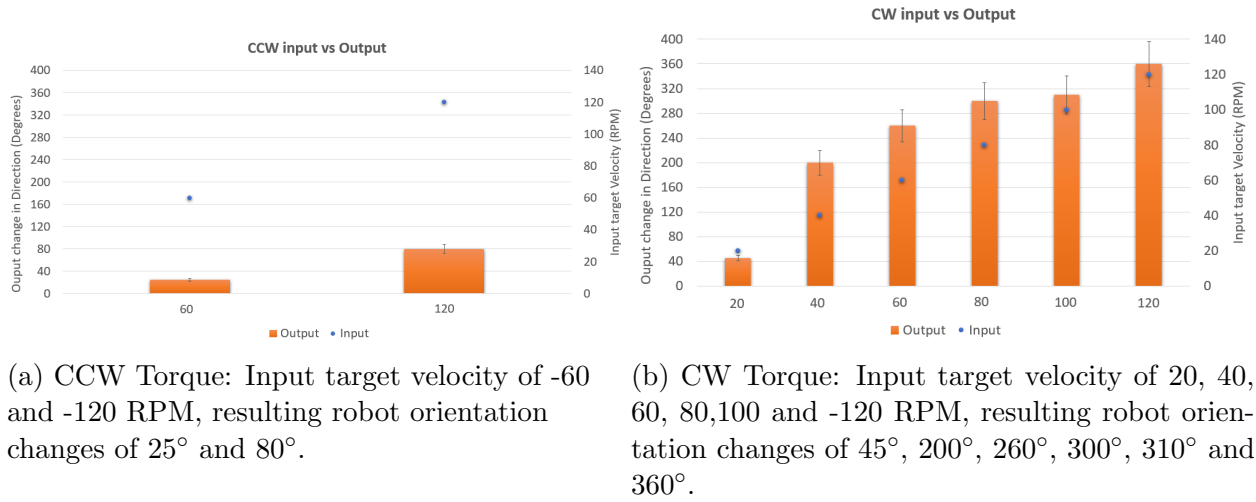
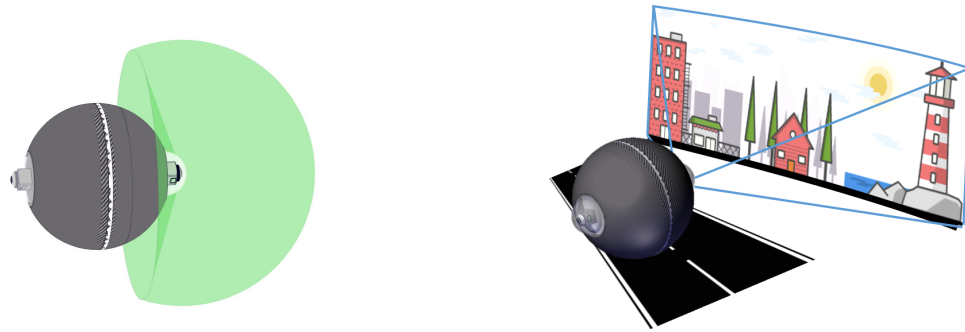


Figure 5.7: Bar chart of input target velocity to output orientation change of the robot. Each input was tested 10 times. Standard deviations are shown with the error bars.

second process, the *Low-pass filter* ensures that only low frequencies can pass through, thus preventing damage to the motor or ESC.

After multiple experiments with short torque pulses on a cement floor, it was estimated that an angular velocity difference of 60 RPM in the flywheel causes the robot to turn 25°. Shorter torque pulses were not as useful as they do not generate enough torque to either overcome the static friction or offer sufficient degrees of rotation. The maximum RPM of the flywheel is 160 RPM. A safer limit of 120 RPM was chosen to use in the DragonBall. For turning clockwise, the target input velocity range was divided into six segments and two segments for left. The user input selects a velocity difference (20/40/80/100/120 RPM or -60/-120 RPM) using the joystick. For clockwise turns the velocity overshoot reduces with higher target input velocity. Figure 5.7 shows a bar chart of input target velocities and the resulting change in orientation of the robot measured over 10 repetition for each input. From Figure 5.6, it can be seen that for clockwise turns the final velocity overshoots. A more gradual change in output angles can be obtained by properly adjusting the final velocity of the flywheel. For this project the set of output angles were chosen as shown in Figure 5.7.



(a) Spherical Robot with omni-directional camera on both sides.

(b) The camera captures views while the robot moves forward.

Figure 5.8: Camera positions in DragonBall.

It should be noted that in order to make turns on cement floors (a surface with low friction) rotations can be performed with fewer torque pulses whereas when operating in a grass field or in a sand pit or similar environment for DragonBall will require more pulses and their range would be different than that of cement. This can be considered as a closed loop control as the current drainage due to output is considered, but since the robot's actual angle is not considered, this control system design should be considered as an open loop system. The user determines the number of torque pulses sent to the robot, hence the user is controlling the robot's rotation.

5.3 Image processing

The DragonBall monitors its environment through two Kodak 360 cameras mounted on the rolling axis, as shown in Figure 5.8. When the IDU rotates the cameras rotate with the IDU. When sufficient torque is generated the body begins to roll and the IDU becomes stabilized with the gravity frame. When the robot moves at constant velocity the camera's view is approximately constant with respect to the ground and only changes during transients when

the IDU is accelerating or decelerating. The camera and BNO055 IMU both are attached to the same structure. Teleoperating the robot from such rotating views is difficult. It is desirable to stabilize this image view. The change in the tilt angle of the IMU due to the robot's motion can be obtained from the IMU data. The same data can be used to compensate for the roll in the camera views. The IMU has a refresh rate of 100 MHz whereas the camera runs only 30 frames per second. The data from IMU can be used as ground truth for this application due to such a high data rate. Also, change in tilt reduces as the robot starts to run at a constant velocity.

The Kodak cameras produce fisheye images. Operating the robot using these image stream directly will also be difficult due to the fisheye distortion and the need to refer to both images. To operate the robot properly a flattened and integrated view from the two cameras is desired. The following sections describes the image processing performed in the DragonBall to accomplish this.

5.3.1 Image stabilization

As illustrated in Figure 5.9 the camera views are not captured in a gravity centric frame. Correcting for the vehicle motion (roll) involves rotating the image around the vehicle's rolling axis. Here we utilize OpenCV [59] to perform this rotation. OpenCV supports a vast library of functions for image processing. An image is in essentially a matrix. Rotating an image is thus equivalent to rotating a matrix. Image rotation operation from OpenCV is done in three steps. The first step is to define the center of the image. The second is to determine the rotational matrix for a given angle and the matrix center. The third and final stage is to rotate the matrix using the rotational matrix.



(a) Unstabilized.



(b) Stabilized.

Figure 5.9: Unstabilized vs Stabilized. Series of images in the left column are taken without stabilizing the camera. Right column images are taken after stabilizing the images.

The rotational angle needed to determine the rotation matrix comes from the BNO055 IMU. The IMU supports sensor fusion, which includes a filter for noise suppression [51]. Since both the camera and the IMU are defined in the same inertial frame the tilt angle and the rotational (roll) angle are linked. The only deviations between the two frames are due to temporal offset and minor frame misalignment. Here we ignore these two issues but these could be established more precisely through appropriate calibration.

It is crucial to keep in mind the time requirements for each operation for image processing, since for teleoperation we need a frame rate that is adequate for driving in real time. Hence all the processes were measured to determine the cost of each function. For example the time required for rotating two 1440 x 1440 size images is 0.01 seconds.



(a) Raw image captured using Kodak 360 camera.



(b) Image after dewarping.

Figure 5.10: Camera views in DragonBall.

5.3.2 Dewarping and stitching

In order to present the operator with a coherent image, the left and right images needed to be combined (stitched) together. Conventional image stitching methods rely on extracting features from images to be stitched and matched. Feature extraction algorithm such as SIFT [59] are used to define these features and provide matching matrices. Iterative methods can be used to identify incorrect matches and remove them [60]. Large overlapping area reduce the number of robust incorrect matches and increases the number of matching features from the images. From these matching feature, a suitable mapping, such as a homography matrix can be computed to map the images to a common reference view and then stitch them together. Smaller overlap regions can lead to incorrect matches and thus failure in stitching. Ho [60] introduced an approach to stitch images from wide angle (fisheye) cameras. His stitching method incorporates four steps; intensity compensation, fisheye unwarping, two-step alignment and blending. Intensity compensation may be required as the intensity of a fisheye camera image can be reduced near the periphery. Unwrapping the images is accomplished using a simple algorithm to convert spherical images to a 360° projection and from there to a 2-D square image. Blending the images in the overlap region produces a



(a) Raw image captured from left and right Kodak cameras. (b) Image after dewarping and blending.

Figure 5.11: Example of blending dewarped left and right images.

clearer image. Image alignment is accomplished using lens misalignment compensation and refined alignment. While Ho's work focused on producing proper image stitching our focus is on usability of the image for teleoperation. Keeping in mind the need for low latency real-time processing a simplified version of Ho's algorithm was adopted for this project. Specifically we exploit the unwarping and blending steps and optimize the algorithm for the required image processing. DragonBall's version of Ho's algorithm is described below.

First for each point in the original image a projected point in a spherical coordinate system is calculated. All these points are then projected on a square plane. Image alignment and overlap regions are identified manually. Linear image blending is performed using OpenCV equation $g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$. Where $\alpha = 0.5$, f_0 is left pixel and f_1 is the right pixel. Since a high video frame rate is desirable it was important to design an efficient image processing pipeline for the DragonBall. The computation described above is implemented as a look-up table from the merged 360° image. This takes the left and the right

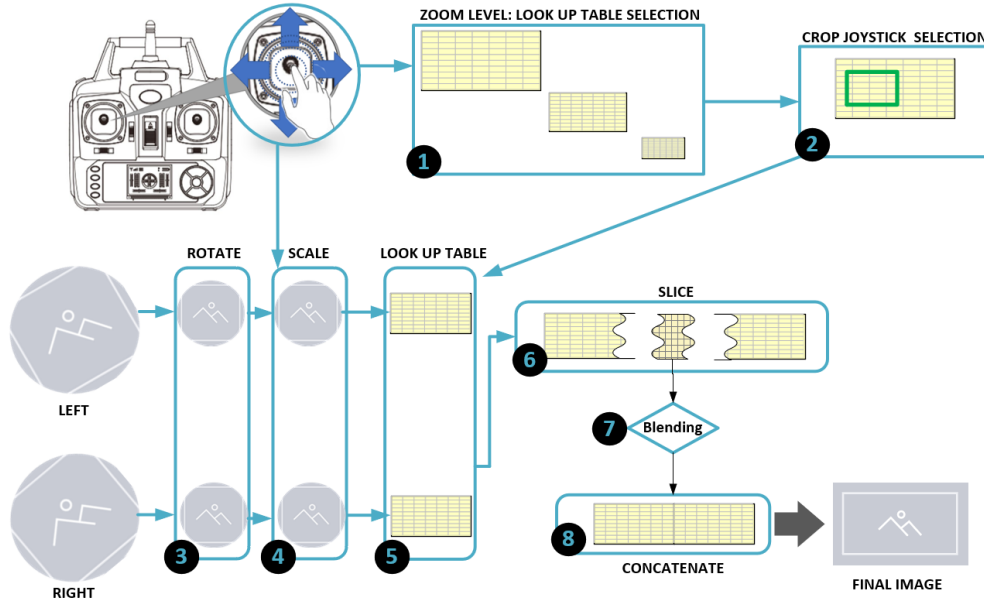


Figure 5.12: Image processing pipeline for the DragonBall. The user first selects the zoom level and the ROI. From the zoom level a scale factor is used to resize and rotate the incoming images from the two camera. Also a corresponding look-up table is selected. Using the ROI the each look-up table is cropped to required dimension. Then this look-up table is used to processes each image which results in two 300 x 300 pixel image. The blending region is sliced and blended using linear blending. All three are then blended and concatenated to create the final image.

images and blends them at the region of intersection. Although it is possible to recompute the full field as of view, more typically a virtual Pan-Tilt-Zoom system is deployed which enables the user to move a simulated camera centered to DragonBall's center to view defined directions. The process of converting the spherical image to 360-degree projection then to 2-D square image could be done using a loop over each pixel. An element wise operation is quite time consuming. In order to avoid this cost a look-up table was created and used for all incoming images.

For the virtual pan-tilt-zoom camera only a limited region of interest of the entire spherical image is selected for processing and is shown to the user. For the lowest zoom level the entire image is processed. Higher zoom levels require a smaller part of the image to be

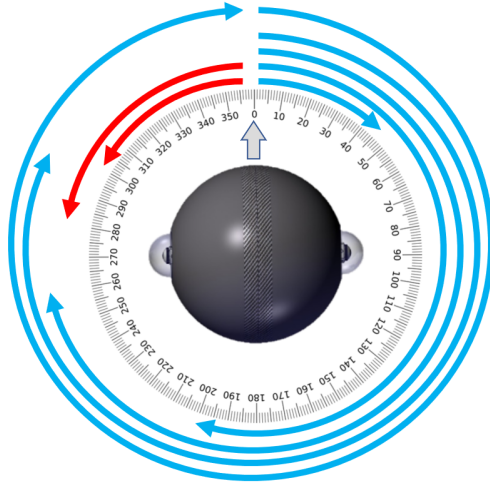


Figure 5.13: DragonBall possible turns with spot turning.

processed. To optimize performance each zoom level has separate look-up tables for each camera and also one table for the overlap/blending function. Figure 5.12 summarizes the image processing pipeline.

5.4 Teleoperation

The DragonBall is teleoperated using a radio controller to send commands and a WiFi connection to receive digital video from the robot to a laptop. The radio controller used is a modified RC remote controller, while the RC input is received in a traditional way but the signal goes to the Arduino instead of the ESC. A laptop runs a GUI for the DragonBall which includes the output of the virtual pan-tilt camera and internal robot's state. The following sections describe how teleoperation was designed and implemented for DragonBall.

As discussed previously, the DragonBall decomposes turns into six primitive right turns and two primitive left turns. The input RPM for left primitive turn are 20, 40, 60, 80, 100 and 120 RPM. The corresponding changes to orientation are 45° , 200° , 260° , 300° , 310°

and 360° respectively. For left turns there are two primitives, -60 and -120 RPM and their corresponding change in output heading are -25° and -80° . As a consequence counter clockwise turn steps are much smaller than clockwise turn steps. A combination of these primitive turn operations are used to perform a required change in orientation. For example if 20° clockwise turn is required then the motion can be divided into two steps. First a 45° clockwise turn and then a 25° counterclockwise turn are executed, resulting in a net 20° clockwise turn. Figure 5.13 summarizes the available primary turns.

Unlike a traditional wheeled robot controlled “on the plane” in the motion of a rolling robot there is a disconnect between the motor (actuator) control and the resulting vehicle control. A joystick is a linear input device, so input from the device was modified to use in teleoperation. Inputs from the joystick are divided into eight sections. Six sections are used for clockwise turns and two sections for counterclockwise turns. When the user moves the joystick nothing happens unless the user stays in one of the sections for more than 8 seconds. In the GUI a prompt indicates the selected zone. This way the user can go from zero to 360° selection without worrying about selecting another input between the sections. Also when a section is selected the user can only go back to zero, as the flywheel needs to stop for the next torque pulse.

During forward motion, the flywheel is static. This ensures that the robot does not tilt due to precession. Hence the joystick axis, controlling the spot turning, is locked when moving forward. A user prompt is shown in the GUI for the user to remind them of this. Similarly, while spot turning, the robot can not move forward and the joystick axis for controlling forward motion is locked. This ensures that no incorrect input is given to the robot during teleoperation.

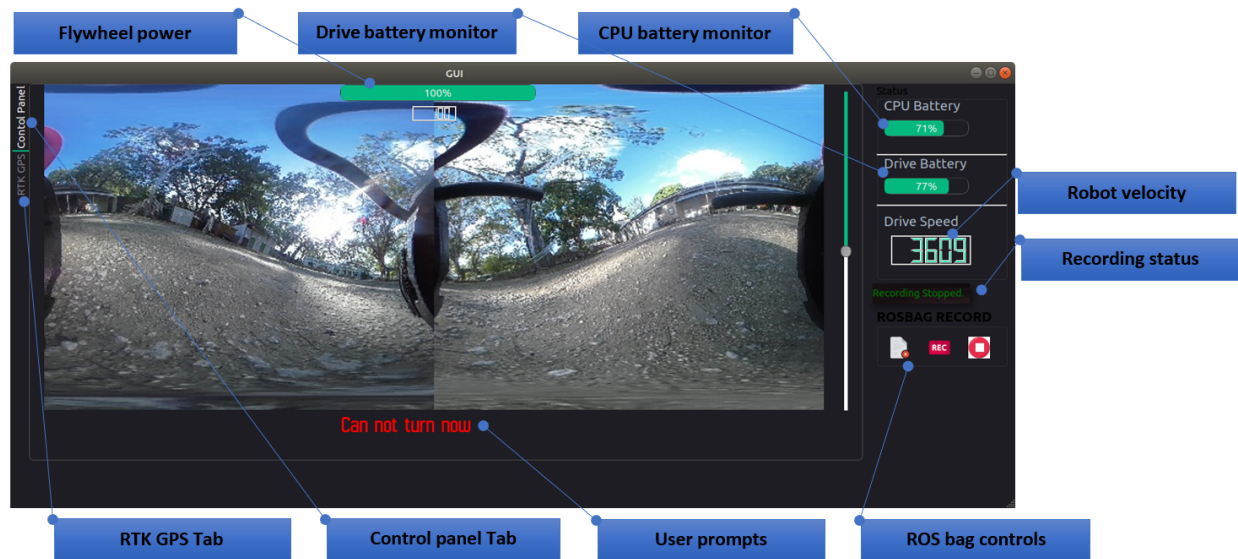


Figure 5.14: DragonBall GUI

5.4.1 GUI

Figure 5.14 shows the GUI of the DragonBall teleoperating system. The GUI has two tabs to show information about the robot. The first tab, the Control Panel, shows the camera output of the robot and user prompts from the robot. Prompts for selecting turn commands or when the command is not available is shown at the bottom of this tab. Figure 5.15 shows the GPS tab. All other information, such as battery remaining voltage, and ROS status are shown on a side panel of the GUI. At the top center of the display video images from the robot, a progress bar and the flywheel remaining speed are shown. The user needs to wait for the flywheel to stop for the next command, without seeing the status of the flywheel it can become frustrating. The progress bar and the number panel provides the percentage of RMP currently used. The highest RPM is 120 RPM. The moment the flywheel starts to rotate the progress bar and number panel monitor what percentage of the flywheel's rotation is being reduced. The user simply needs to wait until the progress bar goes back to 100 percent. The

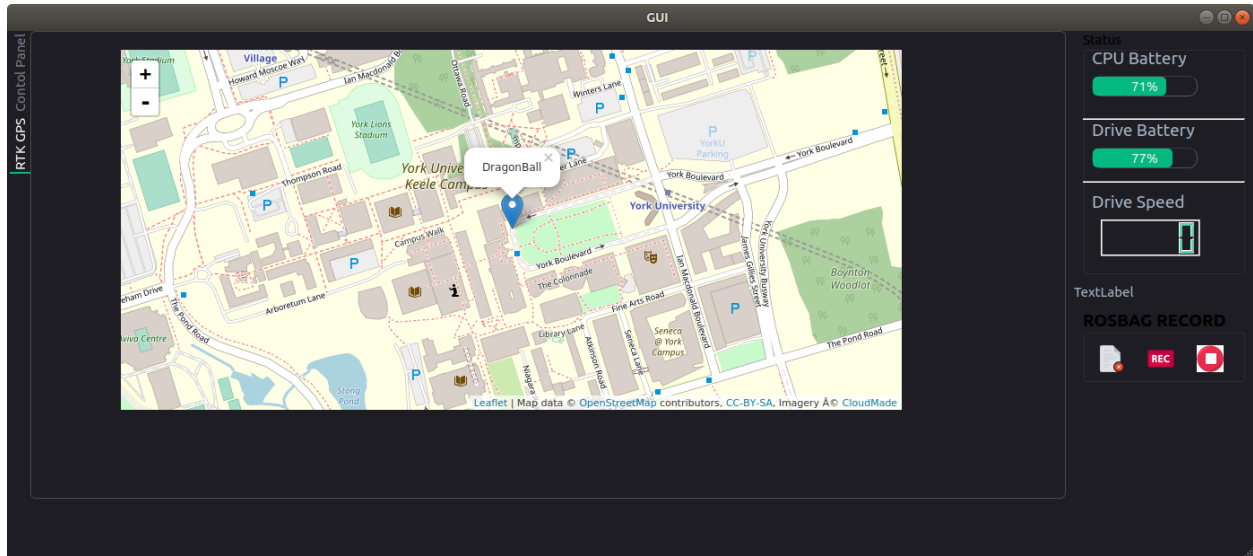


Figure 5.15: DragonBall GPS tab

inspiration for this user interface is taken from the “life bar” common in many video games.

5.5 Example usage of the teleoperation system

The purpose of this test was to validate the teleoperation system by driving the robot over a complex track. A figure eight track was designed to evaluate the teleoperation system. The figure of eight track includes four right turns and four left turns, as shown in Figure 5.16. The DragonBall starts from the center and moves around the track as shown in Figure 5.16. A run begins with the robot in the center then completing the figure eight path, finally coming to rest at the center.

During each run of the track, the user input and robot state data were logged. This data is related to the control of the robot and includes the latency of the response, the user’s perception of the outside world from the robot, the position of the robot and its internal

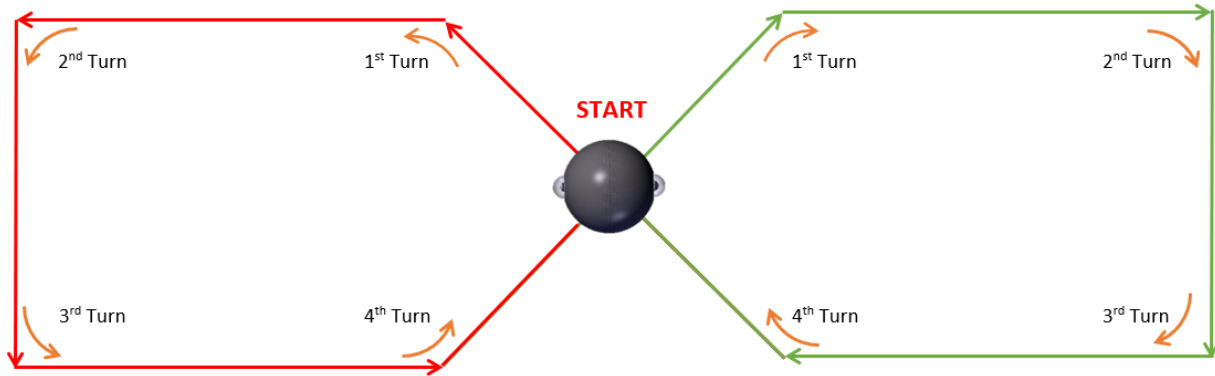


Figure 5.16: Figure eight track

state as a function of time. This data helps identify the validity and reliability of the control systems

Due to COVID-19 pandemic it was not possible to do a study with a large number of participants. Only the author was able to run the study. The study involved driving the robot through the predetermined track using the joystick and the display. The display shows the robots view of the outside world. Performing this experiment required the DragonBall, controller joystick, RTK GPS, a laptop with DragonBall GUI, and a pavement with the marked track. Figure 5.16 shows time-lapse image of DrgonBall moving in a figure eight track. The robot starts from the center of the track, while the user is seating at far corner operating the robot using RC controller and video from a laptop. A camera was installed on corner of the track, which allows viewing the path the robot took during the trail runs. Lane markers made of paper are placed around corners to help the operator keep the robot on track. Each side of the corner has a dimension of 120 x 120 inches.

5.5.1 Results and discussion

The DragonBall completed the track approximately 11 minutes on average over seven runs. Figure 5.17a shows time-lapse footage of the track. The 90° clockwise turns were performed by using two 20 RPM clockwise pulses. And for counterclockwise turn two -60, one -120 and 20 RPM input was used, the resulting turn was approximately 85° . The other turns required were 45° counterclockwise and clockwise. For counterclockwise turns two -60 RPM input was and one 20 RPM input used for clockwise. It should be noted that the counterclockwise turns are not precise to what is required, as two -60 RPM input produces 50° and not 45° .

5.6 Summary

The visual teleoperation of the DragonBall required integrating all the driving systems together with the image processing in the DragonBall GUI. Out of three available maneuvering (Drive forward, spot-turn, Turn) two were implemented due shortage of time. Developing the drive control required implementing a cascaded PID control. A open loop control system was implemented for spot-turning. Image processing required multiple steps, best summarized in Figure 5.12. The GUI utilizes all the above mentioned systems to provide the user with a means to control the DragonBall. Due to COVID-19 a full user study was not possible so a demo test was performed to validate the system. The next chapter summarizes entire work and future work.



(a) DragonBall following the figure eight track.



(b) DragonBall at the center of the track.

Figure 5.17: Teleoperating DragonBall

Chapter 6

Summary and future work

6.1 Summary

Mobile robots are emerging at the epicenter of unmanned hazardous environment monitoring research. Their ability to carry out complex autonomous and teleoperation tasks in a hostile environment makes them ideal for monitoring or guarding hazardous environments. A large number of mobile robots have been developed for hazardous environment monitoring. This work describes the development of the DragonBall spherical robot. The fundamental idea behind developing the DragonBall was to investigate the capabilities and realities of deploying a spherical robot in hazardous environments. In a hazardous environment monitoring task, two of the biggest challenges are how to remediate the vehicle after deployment and effective teleoperation in a complex environment. Cleaning the robot has to be sufficiently straightforward so that deploying the robot in a hazardous environment does not make the

robot hazardous afterward. A properly designed spherical robot should address these challenges. The shell of DragonBall provides a sealed barrier from the outside environment, keeping the internal electronics safe from external contaminants. The primary advantage of having an internal drive unit (IDU) is that it enables mobility without the need for any structure protruding from the robot's shell. This is a significant advantage over other vehicle designs in terms of cleaning the robot. It is crucial that the robot is designed to minimize the possibilities that the vehicle can get caught, stuck or flipped over during operation. A spherical robot design mitigates these concerns due to their mode of locomotion and shape. A true holonomic spherical robot has the capability of maneuvering through extremely difficult environments within which traditional vehicles may fail. Also since there is no "top" to the vehicle, it is not possible to flip this robot upside down. It also lacks a stable "inverted" state from which it can not recover. Although the DragonBall is quasi-holonomic it is capable of turning on a spot, making tight turns, etc. A limitation of spherical robot is the non stationary sensing platform associated with the design. DragonBall utilized two axial ports and wide field of view camera to provide a visual sight for teleoperation. IDU stabilized imagery addresses, at least in part, the difficulties associated with the rotating sensor frame.

The goal of this project was to develop a platform that can be used to validate a teleoperated spherical robot and develop an effective visual teleoperation system. In order to operate in outdoor environment, it was necessary to develop a robust platform that can withstand outdoor weather conditions such as rain, sand, mud, etc. This imposes certain design constraints such as, sealed shell, scratch and impact resistance, long operations, etc. Typically spherical robots have adopted a clear acrylic shell to provide a clear view of the outside world and gives shape to the robot. Acrylic shells typically fail to provide sufficient strength for outdoor operations. Also with time, a clear shell can become scratched leading to blurry images from within the shell. For the DragonBall, a carbon fiber shell was

fabricated as the primary shell of the vehicle. A hand layout method was used to make the carbon fiber shell. Many construction challenges were resolved with rapid prototyping capabilities, such as laser cutting, 3D printing, etc. The shell also has two windows on each side with transparent dome ports for cameras. The lightweight shell provides DragonBall with the spherical shape and strength that it requires.

The choice of an internal drive unit was a crucial decision in the design of DragonBall, as the performance of the robot was highly depended on this decision. A practical and methodical approach to this selection was taken in making this decision. First, most of the current IDUs designs were evaluated in terms of their performance, manufacturing ease, robustness, etc. Then a house of quality was established to evaluate their performance relative to the needs of this project. Based on this evaluation an IDU was designed and prototyped to evaluate the design choice. The prototype, Spherico, provided with much valuable information for developing DragonBall. The IDU design of Spherico was improved upon for the DragonBall.

The electromechanical design of any robot is key to it's performance. The increased popularity of robotics in hobby world has enabled access to an extensive market of electronics components for usage in the DragonBall. Utilizing that market space, a range of electronics components were adopted for use in the DragonBall. The DragonBall also incorporates custom devices. Proper integration of such electronics and sensors is crucial.

Integration of internal and external sensors together with a control system enables a robot to perform its task. Poor sensor integration with a control system may allow it perform its task, but the performance will not be adequate. The DragonBalls sensor and control system were designed considering outdoor experimental and operational issues. Although DragonBall is designed to be teleoperated, the control system and the sensor are designed

Expensive parts in DragonBall	
Item	Price
TX2	\$600.00
Kodak Cameras	\$500.00
RTK GPS	\$700.00
Shells	\$800.00
Total	\$2,600.00

Table 6.1: List of most expensive parts in DragonBall

to enable autonomous operation in the future.

Teleoperation of a robot requires sensor capture and presentation of this information to the user. DragonBall has an enveloping field of view around itself. The fisheye images from DragonBall’s cameras provide full views of the external world from the robot. For the human operator a flattened and stabilized view of the outside world is required for teleoperation. This required flattening the images from both cameras and stabilizing them using IMU data. The GUI had to combine all the above mentioned information in a manner such that the user can drive the robot easily. Inspiration were drawn from video games to design the GUI. To test the teleoperation of DragonBall a track was set to drive the robot while recording the user’s and robot’s data. A test with a bigger group would have been appropriate, unfortunately due to COVID-19 this was not possible.

DragonBall was mostly developed using retrofitting hobby RC electronics. Doing so kept the price level to be lower for most electronics parts. Few parts had to be purchased that was expensive such as the on board TX2 computer, cameras etc. Table 6.1 shows a list of most expensive parts in the DragonBall. Since the DragonBall was developed in York University most of the manufacturing costs were negligible as university facilities were used.

6.2 Lesson learned

The Carbon fiber shell provides DragonBall with structural integrity and protection from outside environment. Carbon fiber was chosen for its rigidity and strength. On the other hand fiberglass has a higher impact strength and is more cost effective. The reason for not choosing fiberglass is due to its lack of rigidity. Rigidity was important for DragonBall's design. However, the shape of each side of DragonBall obtains rigidity due to its shape. This may have allowed the use of Fiberglass instead of Carbon Fiber. As carbon fiber shields radio signals, using fiberglass would have been allowed the antennas to be placed inside the robot.

Both carbon fiber and fiberglass requires elaborate setup and a long time to manufacture. Conventional 3D printed parts are made of PLA or ABS, both are examples of thermoplastics material. Neither of these has the strength of a part manufactured with carbon fiber. Recent development in 3D printers enable them to print parts with a carbon fiber composite filament. The carbon fiber filaments are made of chips of carbon fiber and standard filament such as nylon. While this type of material is not as strong as carbon fiber or fiberglass, it has a flexural strength of 50 MPa. The body of DragonBall could have made of such material. If any part was broken such part can be easily replaced by 3D printing another.

One of the key lesson learned from DragonBall is the importance of evaluation rigs and module testing. Each part was rigorously tested before assembly, but when assembled not all part worked together. The synergy of all parts together opened up other issues that were not be encountered when parts were tested alone. Instead of assembling the robot immediately after testing all parts, it would been better to assemble individual modules and test each module in a test setup and then to move up to the next assembly. This way there would have been less parts to debug at a time and less failure points to test.



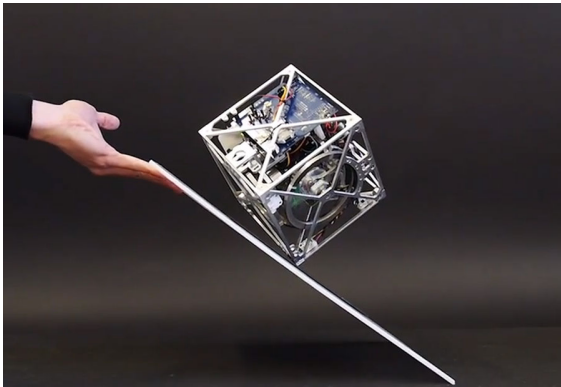
Figure 6.1: Super Ball Bot a Tensegrity robot. Image reprinted from NASA/SuperBallBot.

6.3 Future work

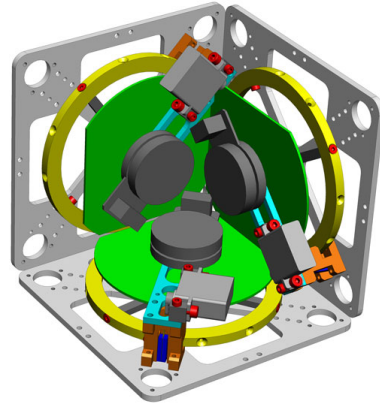
The DragonBall project has opened up possibilities for further studies in many aspects. The following sections describes a number of these topics and how can they can be studied further.

6.3.1 Shapes

In an ideal locomotion system the energy spent to move forward should be minimal while offering maximum maneuverability. On a flat surface a spherical shape exhibits those properties. The shortcoming of spherical shapes is evident in an uneven surface, where any bump on the surface will require the center of gravity to rise up and hence require extra energy. Tensegrity robots [61] offer a potential solution to such problems (see Figure 6.1). Research in Tensegrity robots have shown their energy requirement to be much lower over uneven environments. The control system of such robots is very complicated and certainly



(a) Cubli stabilizing on inclined plane. Image reprinted from ETH Zurich/Cubli.



(b) Cubli internals, showing all three reaction wheels. Image reprinted from ETH Zurich/Cubli.

Figure 6.2: Cubli Robot.

more complicated than that of DragonBall. A marriage between these two solutions might produce a more efficient system than either of them.

6.3.2 Maneuvering

Just as energy efficiency is important in driving forward, it is also important in maneuvering. Implementing a turning mechanism using CMG will help DragonBall maneuver more precisely and will be energy efficient than it does currently. This would require further research and study into the vehicle dynamics and CMG systems. Implementing this would require improving real-time control in the robot and also improve the encoders. Cubli [62] can be taken as an inspiration (see Figure 6.2), where a modified CMG control system is applied. Cubli is a cube which can jump, balance and move forward just by using three reaction wheels to control itself.

Another novel idea of improving maneuvering would be to change the shape of the robot. While spherical shapes have a low energy requirement to move forward, there are other

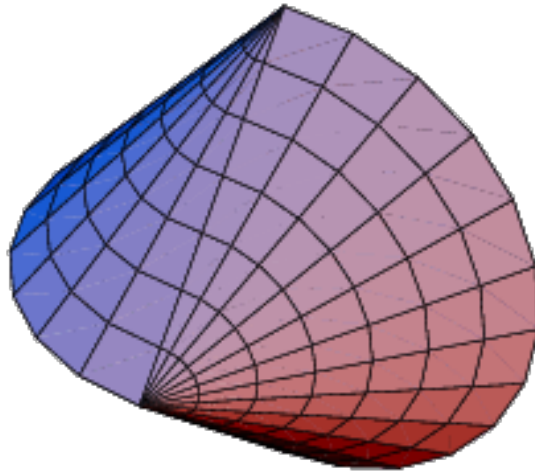


Figure 6.3: Sphericon external shell structure. Image reprinted from Mathworld Wolfram/Sphericon.

vehicle shapes that could be used as well, such as that used by Sphericon (Figure 6.3). Sphericons [63] rolls on an inclined plane just as a sphere but instead of rolling in straight path it takes a patterned path. Different shapes of sphericons offers different patterned path that can be exploited for maneuvering.

6.3.3 Image Stabilization

The IMU in DragonBall has a refresh rate of 100 Mhz and the cameras run at 30 fps. The image stabilization for DragonBall was made under the assumption that the IMU represents the gravity vector and thus this should be “up” for vehicle teleoperation. The IMU was used as it has a higher refresh rate than the cameras and as acceleration and deceleration phases of the robot were not considered to be crucial in terms of teleoperation. It was observed while running the DragonBall that the acceleration and deceleration phases were more frequent than originally assumed.

Stabilizing the images is quite difficult. The image is stabilized using the IMU data,

but the image processing requires time and the output image is delayed from the current IMU reading. This lag causes the image to not be stabilized during the acceleration and deceleration phases. Furthermore acceleration rotates the net acceleration vector away from the “up” defined by gravity. A system which can predict the next IMU reading to make the image rotate to such value will help with image stabilization. Such system could use a PID or Kalman Filter. Also another system is required to quantify the lag between the image output and IMU reading and might integrate visual information (e.g. the horizon) into defining the “up” direction.

6.3.4 Autonomous Operation

DragonBall was developed to be remotely operated while keeping in mind that future usage will be autonomous. The overall architecture was designed to be autonomous, meaning all the control goes through the computer. This allows to make the system autonomous since the on board computer has the connection to all the control system. DragonBall offers all the sensor input and control parameters that an autonomous needs. Developing an autonomous driving system for DragonBall should be fairly easy.

6.4 Summary

DragonBall is a fully enclosed spherical robot with enclosed actuation and sensor systems. In addition to IMU and GPS sensors, the robot also relies on two wide field of view cameras mounted in ports along the vehicle’s axis. These cameras provide a panoramic view of the space around the robot and can be used to provide a simulated Pan-Tilt-Zoom camera. This

simulated Pan-Tilt-Zoom camera coupled with a control system that allows a driver to choose a vehicle acceleration and turning rate is an effective control from vehicle tele-operation. Ongoing work with DragonBall will explore further development in various aspect of the robot, starting from controls to GUI development.

Bibliography

- [1] G. C. Schroll, “Dynamic model of a spherical robot from first principles,” Master’s thesis, Colorado State University. Libraries, Fort Collins, CO, USA, 2010.
- [2] V. A. Crossley, “A literature review on the design of spherical rolling robots,” *IEEE Transactions on Magnetics, Pittsburgh, PA*, pp. 1–6, 2006.
- [3] D. Balandin, M. Komarov, and G. Osipov, “A motion control for a spherical robot with pendulum drive,” *Journal of Computer and Systems Sciences International*, vol. 52, no. 4, pp. 650–663, 2013.
- [4] B. SHORTHOUSE, “Self propelling device,” Great Britain Patent US819 609A, May 1, 1906.
- [5] A. D. MCFAUL, “Toy,” U.S. Patent US1 263 262A, May 19, 1918.
- [6] J. L. TATE, “Toy,” U.S. Patent US508 558A, Nov. 14, 1893.

- [7] E. Satorius, P. Estabrook, J. Wilson, and D. Fort, “Direct-to-earth communications and signal processing for mars exploration rover entry, descent, and landing,” *IPN Progress Report*, pp. 42–153, 2003.
- [8] S. Lichiardopol, “A Survey on Teleoperation,” *Technische Universitat Eindhoven, DCT report. Eindhoven, Netherlands*, 2007.
- [9] N. W. Oumer, “Development of wireless control system for a spherical robot,” *Tech Report, Espoo, Finland*, 2009.
- [10] A. Behar, J. Matthews, F. Carsey, and J. Jones, “NASA/JPL Tumbleweed Polar Rover,” in *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*, vol. 1. Big Sky, Montana: IEEE, 2004.
- [11] J. L. Wilson, A. P. Mazzoleni, F. R. DeJarnette, J. Antol, S. Harris, G. Hajos, and C. Strickland, “Design, Analysis and Testing of Mars Tumbleweed Rover Concepts,” *Journal of Spacecraft and Rockets*, vol. 45, no. 2, pp. 370–382, 2008.
- [12] J. Suomela and T. Ylikorpi, “Ball-shaped robots: an historical overview and recent developments at TKK,” in *Field and Service Robotics*, vol. 25 (6). Springer, 2006, pp. 343–354.
- [13] Y. Sugiyama, A. Shiotsu, M. Yamanaka, and S. Hirai, “Circular/spherical robots for crawling and jumping,” in *Proceedings of the IEEE Int. Conf on Robotics and Automation (ICRA)*, Barcelona, Spain, Apr. 2005.
- [14] P. Harmo, A. Halme, H. Pitkänen, P. Virekoski, M. Halinen, and J. Suomela, “Moving eyeinteractive telepresence over internet with a ball shaped mobile robot,” *Automation Tech Lab, Finland, Oct*, vol. 2, p. 6, 2001.

- [15] R. Mukherjee, M. A. Minor, and J. T. Pukrushpan, “Simple motion planning strategies for spherobot: a spherical mobile robot,” in *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, vol. 3, Phoenix, AZ, USA, 1999, pp. 2132–2137.
- [16] F. Tomik, S. Nudehi, L. L. Flynn, and R. Mukherjee, “Design, fabrication and control of spherobot: A spherical mobile robot,” *Journal of Intelligent & Robotic Systems*, vol. 67, no. 2, pp. 117–131, 2012.
- [17] V. A. Joshi, R. N. Banavar, and R. Hippalgaonkar, “Design and analysis of a spherical mobile robot,” *Mechanism and Machine Theory*, vol. 45, no. 2, pp. 130–136, 2010.
- [18] V. A. Joshi and R. N. Banavar, “Motion analysis of a spherical mobile robot,” *Robotica*, vol. 27, no. 3, pp. 343–353, 2009.
- [19] T. Ylikorpi, “A biologically inspired rolling robot for planetary surface exploration,” *Automation technology laboratory, Espoo, Finland*, 2005.
- [20] D. Liu, H. Sun, Q. Jia, and L. Wang, “Motion control of a spherical mobile robot by feedback linearization,” in *2008 7th World Congress on Intelligent Control and Automation*, Chongqing, China, 2008, pp. 965–970.
- [21] M. Nagai, “Control System for a Spherical Robot,” Master’s Thesis, Lulea Univ. of Technology, Espoo, Finland, 2008.
- [22] W.-H. Chen, C.-P. Chen, W.-S. Yu, C.-H. Lin, and P.-C. Lin, “Design and implementation of an omnidirectional spherical robot omnicon,” in *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Kaohsiung, Taiwan, 2012, pp. 719–724.

- [23] A. Halme, T. Schonberg, and Y. Wang, “Motion control of a spherical mobile robot,” in *Proceedings of 4th IEEE International Workshop on Advanced Motion Control-AMC’96-MIE*, vol. 1, Mie, Japan, 1996, pp. 259–264.
- [24] R. Chase and A. Pandya, “A review of active mechanical driving principles of spherical robots,” *Robotics*, vol. 1, no. 1, pp. 3–23, 2012.
- [25] J. Alves and J. Dias, “Design and control of a spherical mobile robot,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 217, no. 6, pp. 457–467, 2003.
- [26] V. Kaznov and M. Seeman, “Outdoor navigation with a spherical amphibious robot,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010, pp. 5113–5118.
- [27] J.-F. Laplante, P. Masson, and F. Michaud, “Analytical longitudinal and lateral models of a spherical rolling robot,” Technical report, Univ. Sherbrooke, Sherbrooke, QC, Canada, pp. 15–22, 2007.
- [28] M. Kabala and M. Wnuk, “Design and construction of roball a spherical nonholonomic mobile robot,” *Unit of Foundations of Cybernetics and Robotics, Wroclaw University of Technology, Wroclaw*, 2004.
- [29] W.-H. Chen, C.-P. Chen, J.-S. Tsai, J. Yang, and P.-C. Lin, “Design and implementation of a ball-driven omnidirectional spherical robot,” *Mechanism and Machine Theory*, vol. 68, pp. 35–48, 2013.
- [30] Y. L. Karavaev and A. A. Kilin, “The dynamics and control of a spherical robot with an internal omniwheel platform,” *Regular and Chaotic Dynamics*, vol. 20, no. 2, pp. 134–152, 2015.

- [31] Y. Huang, G. Zhu, C. Wang, and H. Huang, “Structure design and dynamical modelling of a spherical robot driven by omnidirectional wheels,” in *MATEC Web of Conferences*, vol. 153, Guilin, China, 2018, p. 2003.
- [32] P. Mojabi *et al.*, “Introducing august: a novel strategy for an omnidirectional spherical rolling robot,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, Washington, DC, USA, 2002, pp. 3527–3533.
- [33] G. Shu, Q. Zhan, and Y. Cai, “Motion control of spherical robot based on conservation of angular momentum,” in *International Conference on Mechatronics and Automation*. Changchun, China: IEEE, 2009, pp. 599–604.
- [34] F. Franceschini, *Advanced Quality Function Deployment*. St. Lucia, Boca Raton: CRC Press, 2001.
- [35] A. Martinez and E. Fernández, *Learning ROS for Robotics Programming*. Birmingham, UK: Packt Publishing Ltd, 2013.
- [36] B. B. Dey and M. Jenkin, “Spherico: rapid prototyping a spherical robot,” in *2018 IEEE International Conference on Information and Automation (ICIA)*, Wuyi Mountain, Fujian, China, 2018, pp. 957–962.
- [37] R. A. Ibrahim, *Liquid Sloshing Dynamics: Theory and Applications*. Cambridge, UK: Cambridge University Press, 2005.
- [38] D. Sasser, “Structural Analysis of Carbon Fiber-Epoxy Composite Surfaces with Polar Stacking Sequences,” Ph.D. dissertation, University of Washington, 2015.
- [39] D. Gay and S. V. Hoa, *Composite materials: Design and Applications*. Boca Rotan: CRC press, 2007.

- [40] E. J. Barbero, *Introduction to Composite Materials Design*. Boca Raton: CRC press, 2017.
- [41] A. Witkowski, M. Kandis, J. Reuter, W. Machalick, R. Kelsch, and A. Sengupta, “Design of subscale parachute models for msl supersonic wind tunnel testing,” in *20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, Seattle, Washington, 2009, pp. 2988–2999.
- [42] F. C. Campbell Jr, *Manufacturing Processes for Advanced Composites*. Oxford, UK: Elsevier, 2003.
- [43] D. Kleppner and R. Kolenkow, *An Introduction to Mechanics*. Cambridge, UK: Cambridge University Press, 2013.
- [44] “Jetson tx2 module,” Apr 2019. [Online]. Available: <https://developer.nvidia.com/embedded/buy/jetson-tx2>
- [45] Mateksys, *Flight Controller F405-CTR*, 2017 (accessed April 3, 2018). [Online]. Available: <http://www.mateksys.com/?portfolio=f405-ctr>
- [46] “Kodak pixpro sp360 4k vr camera.” [Online]. Available: <https://www.kodak.com/gb/en/Consumer/Products/Digital-Cameras/Action-Camera/SP3604K/default.htm>
- [47] A. Rituerto, L. Puig, and J. J. Guerrero, “Visual SLAM with an omnidirectional camera,” in *2010 International Conference on Pattern Recognition (ICPR)*, Washington, DC, USA, 2010, pp. 348–351.
- [48] Emlid, *Reach EMLID manual*, 2015 (accessed February 3, 2018). [Online]. Available: <https://docs.emlid.com/reach/>
- [49] Phidgets, *Optical Rotary Encoder ISC3004*. [Online]. Available: <https://www.phidgets.com/?&prodid=403>

- [50] Nubotics, *WW-12 WheelWatcher Encoder*, 2010 (accessed January 3, 2018). [Online]. Available: <https://www.nubotics.com/products/ww12/index.html>
- [51] Adafruit, *Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout BNO055*. [Online]. Available: <https://www.adafruit.com/product/2472>
- [52] Hobbyking, *Turnigy XK-4074 2000KV Brushless Inrunner*. [Online]. Available: https://hobbyking.com/en_us/turnigy-xk-4074-2000kv-brushless-inrunner.html
- [53] —, *X-Car Beast Series ESC 1:8 Scale 120A*. [Online]. Available: https://hobbyking.com/en_us/hobbykingr-tm-x-car-beast-series-esc-1-8-scale-120a.html
- [54] —, *Basher 4000mAh 6S 65C Hardcase Pack*. [Online]. Available: https://hobbyking.com/en_us/basher-4000mah-6s-65c-hardcase-pack.html
- [55] —, *PROPDRIVE v2 5060 270KV Brushless Outrunner Motor*. [Online]. Available: https://hobbyking.com/en_us/propdrive-v2-5060-270kv-brushless-outrunner-motor.html
- [56] —, *ZIPPY Compact 2200mAh 4s 60c Lipo Pack*. [Online]. Available: https://hobbyking.com/en_us/zippy-compact-2200mah-4s-60c-lipo-pack.html
- [57] C. Trout, *Essentials of Electric Motors and Controls*. Burlington, Massachusetts, USA: Jones & Bartlett Learning, 2010.
- [58] A. Visioli, *Practical PID Control*. London, UK: Springer Science & Business Media, 2006.
- [59] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. Sebastopol, CA: O'Reilly Media, Inc., 2008.

- [60] T. Ho and M. Budagavi, “Dual-fisheye lens stitching for 360-degree imaging,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, Louisiana, USA, 2017, pp. 2172–2176.
- [61] J. Bruce, K. Caluwaerts, A. Iscen, A. P. Sabelhaus, and V. SunSpiral, “Design and evolution of a modular tensegrity robot platform,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, NJ: IEEE, 2014, pp. 3483–3489.
- [62] M. Gajamohan, M. Merz, I. Thommen, and R. D’Andrea, “The Cubli: A cube that can jump up and balance,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, 2012, pp. 3722–3727.
- [63] E. W. Weisstein, *Sphericon*. [Online]. Available: <https://mathworld.wolfram.com/Sphericon.html>