Individual Project
Final Report

ID Number: F122367

Programme: Mechanical Engineering BEng

Module Code: 23WSC500

Project Title: Development of a Balancing Robot to remain stable on a moving sphere.

Abstract:

This project aims to design, build and implement a fully functional robot capable of balancing on a moving ball. This report details the development of a Ball Balancing Robot (BBR), encapsulating the integration of electromechanical components with a control system, using insights from previous research and design approaches to optimise the stability and functionality of the robot on a low budget.

The BBR uses omnidirectional wheels and is actuated by stepper motors, with the movement executed using a PID controller. Initial testing focused on motor responsiveness and sensor accuracy, validating the effective application of Madgwick's filter for precise orientation measurement. However, the full implementation of the designed controller was not achieved within the project timeline, significant progress was still made in the robot's design and initial testing.

Key challenges included the drift of sensor data and the complexity of control system tuning of dynamically unstable systems. Future work is recommended to focus on improving sensor calibration, the integration of a battery management system for improved mobility and refined control algorithms. This project lays the framework for future developments in Ball Balancing Robot systems, showing promising enhancements in robotic agility and for use in human environments.

Table of Contents

1 Introduction

Ball-balancing robots (BBR), commonly known as Ballbots, represent a complex engineering challenge in the field of dynamically unstable systems [1]. Unlike traditional robots with wheels or legs, Ballbots operate on a single contact point with the ground, enabling omnidirectional movement and high manoeuvrability in tight spaces[2]. Because there is only a single point of contact with the ground, it cannot maintain static stability, and can only be in dynamic balance[3] The system is mainly composed of three parts: the body, the control system, and the ball[1]. The most complex part is the control system which typically consists of 3 or 4 Omni wheels driven by motors and a sensor that reads the orientation and acceleration of the robot and moves the wheels accordingly to dynamically balance the system. This system follows the principles of the inverted pendulum system, needing continuous motion to actively balance the pendulum and prevent falling. The development and study of Ballbots are driven by the aim to place robot workers in human environments, this is due to these types of robots being tall and skinny, having smaller footprints and having the ability to accelerate/decelerate quickly[4], [5]. They are ideal for human environments they have no minimal turning radius and can roll in any direction without having to yaw to change direction[1].

This project aims to design, build, and implement a fully functional robot capable of balancing on a moving spherical surface. It includes the integration of electromechanical design, control systems and sensor technology to achieve dynamical stability. Through comprehensive research and engineering, this report details the progress of the development of the BBR, highlighting key design decisions, technical challenges, and methodologies employed to overcome them. The following sections will include a comprehensive literature review that researches existing solutions of the BBR, and critiques methods used. A detailed design and development section describing the engineering process of the project includes the selection of components and design decisions. A methodology and testing section contains insights into the experimental approach, ensuring the performance and reliability of the robot under varying operating conditions. A results section analyses the data obtained from testing to verify the control design specifications of the robot. This report describes the development of a BBR in detail and all methodologies used

2 Literature Review

2.1 Introduction

In the recent progress of the development of dynamically stable robots capable of navigating human environments, BBR has emerged as a strong point of research due to the high robotic agility possible with this type of robot[4]. This literature review gives a detailed explanation of the various designs of BBR, exploring critical components and aspects of the BBR, including design, control systems and the technologies that enable their functionality.

The first instance of a BBR was found in 2005 in “A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive” at Carnegie Mellon University, it presented a pioneering development in the field of dynamic stability and mobile robotics, This work introduced the concept of the Ballbot, desirable due to its tall and slender characteristics, high manoeuvrability, and omnidirectional movement capabilities[5].

The Ballbot's innovation lies in its ability to move directly in any direction without needing to reorient itself, this is a significant development over earlier two-wheeled designs, (e.g. Segways). The design challenges addressed in developing the ballbot included creating a system that could maintain dynamic stability with a high centre of gravity and a compact footprint.

2.2 Electromechanical Design

2.2.1 CMU Ballbot (2005)

In the development of Carnegie Mellon University’s BBR, the design aimed to approximate human dimensions with a height of 1.5m, a diameter of 400mm, weighing 45kg, for operation in human-centric environments. This robot featured retractable landing legs, enabling it to remain upright when powered off, improving operational readiness and safety. It is powered by a 48V Lead acid battery, chosen for its weight and long operational battery life (several hours), crucial for prolonged autonomous tasks. The BBR computational needs are handled by a 200MHz processor, ensuring robust data processing capabilities onboard. Communication was facilitated through a wireless link, providing reliable data transmission within research facilities[5].The robot ball drive

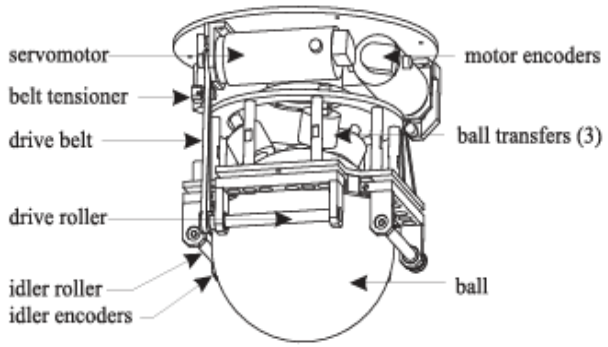


Figure 1- Ballbot inverse mouse-ball drive mechanism.

system is an inverse of a mouse ball drive, as shown in Figure 1- Ballbot inverse mouse-ball drive , where instead of the mouse ball driving the mouse rollers to provide computer input, rollers drive the ball to produce motion. The ball selected was a 200mm hydroformed steel shell covered in a urethane outer layer, this is to increase the friction to make it easier to drive the ball and its good durability characteristics. The ball is driven by the stainless-steel rollers placed orthogonally around the equator of the ball, with these driver rollers connected to high-torque DC servomotors with a pulley. The mechanism also included 2 spring-loaded idler rollers, opposite the driver rollers that apply force at the ball's equator to maintain contact between the driver rollers and the ball. the system stability was measured by a 6DOF IMU, and high-resolution encoders were placed on passive rollers that measured the ball rotation[5].

2.2.2 BallIP (2008)

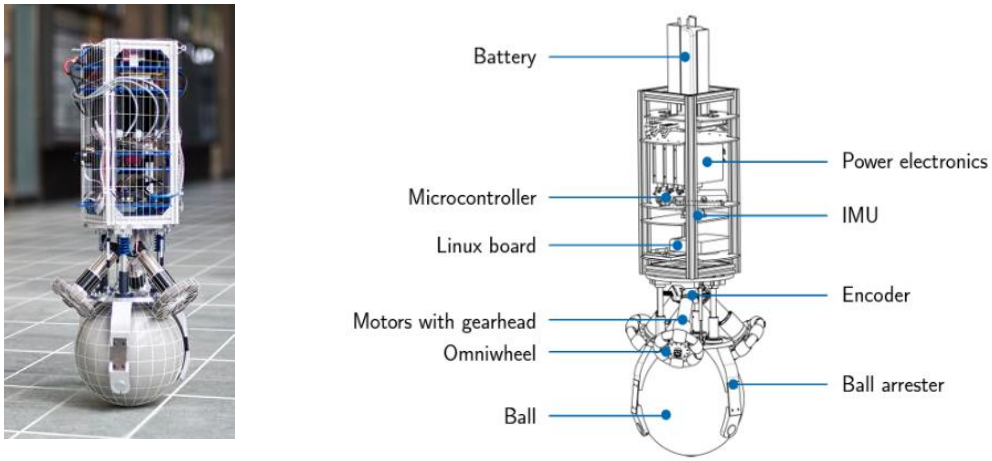
In a paper from Tohoku Gakuin University, it goes into to the components selected for their BBR, it was designed to be more compact than the previous design only 500mm in height and 11kg in weight. The ball selected was a rubber-coated bowling ball (approx. 200mm in diameter), this reduced height and size suggested a focus on stability and manoeuvrability. The ball drive system, shown in Figure 2(b), operates using 3 stepper motors, placed 120 degrees apart from each other, linked to omnidirectional wheels, these were selected due to their single contact line feature which facilitates movement on uneven surfaces. Stepper motors were selected over DC servomotors due to a preference for higher torque capacities (1.2Nm) and open-loop control, this may improve the robot's ability to handle the added weight without the need for complex feedback systems. The motors selected also didn't need reduction gears, simplifying the robot system with the shaft of the motor reinforced with thrust bearings between the motor and the wheel. A cover was also placed on the Omni wheels to prevent dirt from building up and prevent debris from clogging the wheels and reducing the functionality of the wheels. In electronic components to design the control system, it included MEMS attitude sensors (which contained a rate gyro and accelerometer) and a 16-bit microcontroller, powered by 7.2V Ni-MH batteries. Also incorporated a micro-step controller to allow for finer control of motor steps aimed at achieving smoother movement of the wheels, this is useful as it could improve the ability to carry cargo when navigating. This design of Ballbot with 3 omni wheels placed 120 degrees apart has become very common in more recent designs of Ballbots[2].

(a) Ballbot (b) driving mechanism diagram

Figure 2 – BallIP[6], [7], [8].

2.2.3 Project Rezero (2010)

This robot, designed at ETH Zurich, stands 1 metre tall and weighs 14.5kg which was optimized to ensure the robot is both agile and robust to handle various environments. It's composed of 3 main parts: the ball, the propulsion system, and the body, with the body carrying the electronic components. The ball drive system consists of 3 Omni wheels driven by brushless DC motors with gear heads with a maximum torque of 5 Nm, with each motor having an encoder attached to measure the speed and position of the motors. The ball selected was an aluminium hollow sphere to reduce the load and inertia of the robot compared with a ball of the same size, increasing the agility. The ball is coated with polyurethane to increase interaction between the ball and Omni wheels, with a ball arrester also to push the ball towards the Omni wheels to increase the contact force. The sensor measurements are done by an IMU which measures all three Euler angles and angular rates of the body, with an ARM 7 microcontroller implementing the control algorithms. the set points are sent to the microcontroller by a Linux board over RS-232. A diagram of the robot is shown in Figure 3.



(a) Ballbot (b) Project Rezero system description

Figure 3 – Project Rezero[7]

2.2.4 Comparative Analysis

This comparative analysis of the electromechanical designs of the previously described Ballbots aims to show the engineering choices made to achieve performance goals through components, ball drive mechanisms, and ball selection. Each design's choices significantly affect each robot's mobility, stability, and ability to overcome external disturbances.

The CMU ballbot is designed to approximate human dimensions, with a height of 1.5m and a weight of 45kg. This larger size increases the inertia of the robot, requiring more complex control systems to ensure stability and handle increased momentum in comparison. However, due to its large size, it has limited manoeuvrability in confined spaces. Conversely, BallIP is considerably smaller (0.5m in height and 11kg), giving it much higher agility due to its lower inertia. This also allows for quicker acceleration and deceleration and better responses to changing of direction, improving its ability in confined spaces to the previous design. However, due to its smaller base, it has reduced stability, making it harder to maintain dynamic equilibrium under external disturbance. Project Rezero's design uses a hollow ball, decreasing the inertia per unit mass, and

allowing for faster and more precise control due to lowering the torque required for movement. CMU ballbot also uses a hollow ball, however, BallIP uses a bowling ball which is not hollow, hindering its ability to be controlled. All BBR's Balls are coated in forms of urethane to increase friction and prevent slip from the ball drive systems[2], [6], [7].

Regarding ball drive mechanisms, the CMU ballbot utilises an inverse mouse ball drive with high torque DC servomotors, allowing for smoother movement and being more suitable in human environments. However, this is at a much higher power cost, and it can't move in yaw due to the way the mechanism is placed around the equator in parallel to the ZX and ZY axis. BallIP on the other hand, uses a much simpler ball drive system with the stepper motors allowing for less complex control systems to implement due to its open-loop characteristics, reducing the costs of this system. However, this open loop control may limit the robot's performance in more dynamically challenging settings, such as inclines. The use of omnidirectional wheels in BallIP has become very common in more recent adaptations of the Ballbot, due to its ability to be able to move in any direction, also shown in Project Rezero. Rezero incorporates the use of brushless DC motors, achieving higher precision combined with the use of encoders, however at higher costs in comparison to BallIP[8].

In summary, each Ballbot's design reflects a unique approach to the electromechanical design of a BBR to help achieve the goal of dynamic stability by providing a framework to implement control systems.

2.3 Controller Design

Control systems play a pivotal role in maintaining the stability and movement of dynamically unstable systems like Ballbots. This section will focus on two main types of controllers used in BBRs: PID (Proportional-Integral-Derivative) and LQR (Linear Quadratic Regulator) controllers. These controllers are designed to maintain the balance and positioning of the robot through mathematical models to predict and react to the falling of the robot to maintain balance. These differing strategies impact the performance of the robot's ability in terms of stability and responsiveness. This section will explore the mathematical foundations and practical implications of these control systems.

2.3.1 PID (BallIP)

A PID controller adjusts the controller output based on error values (difference between desired and actual state). The proportional term gives an output proportional to the error, this helps in Ballbots to counteract deviation from the desired angle of inclination. The integral term gives an error over time which can smooth the response for offsets and the derivative term reacts to the rate of change of error, helping to improve the system's stability by damping oscillations.

BallIP is modelled as a 3D inverted pendulum in the ZX and ZY planes and controls the motors to oppose motion in these planes through derivations of the position and velocity of the robot. It provides a derivation of optimal wheel speeds including the zenith angle ϕ, which is the angle between Omni wheels and the vertical axis of the ball. A diagram of the relationship between the ball and wheels is shown in Figure 4. The Zenith angle ϕ determines the ratio of wheel to ball rotation and affects the robot body support. This was then used to derive the velocities of each wheel in the XY plane, to work out the combination of the wheels to move the robot around the X and Y axes respectively. This later would become the omnidirectional ball drive system that is the output of the control system.

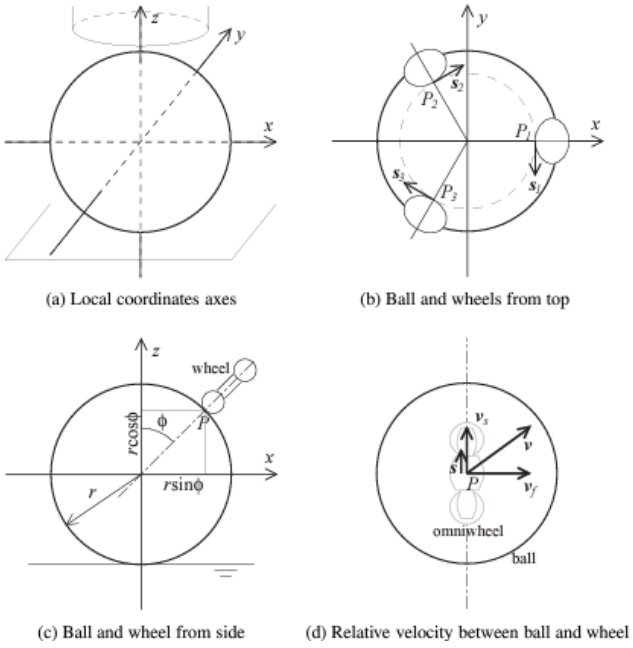


Figure 4 - Axis definition and relationship between ball and wheels[8].

After deriving the speed at which the wheels relate to their respective axis, a control system was derived. A block diagram of the system described is shown in figure 5, It uses its control input as acceleration, which is unlike typical inverted pendulum systems, which would use torque 3esor force. The use of acceleration provides robustness against changes in inertia parameters with feedback based on robots' inclination and wheel travel. The data was measured by combining the data from an accelerometer and a rate gyroscope to measure the robot's attitude, being directly implemented into the control loop. The system's stability was confirmed using Nyquist diagrams

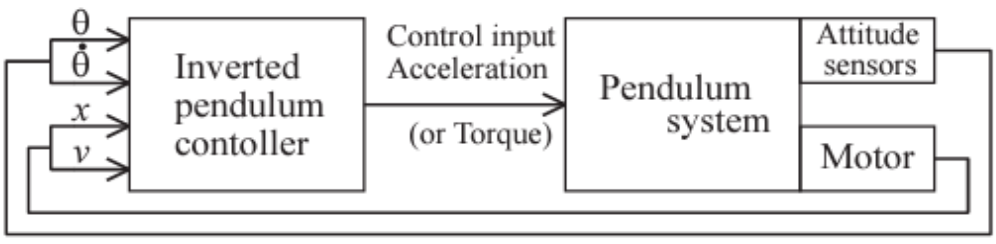


Figure 5 - BallIP Block Diagram[8].

showing effective control over varying inertia loads. The control system was implemented onto the microcontroller using C-code, which made the implementation difficult. It could handle a tilt of up to 8 degrees without slipping, In position tests, the robot could maintain station and traverse effectively and could manage rotations around yaw under controlled tests. There are also images of the robot being used to carry a concrete block, and cooperating with a human to carry a large wooden frame without toppling. While acceleration input simplifies some aspects, it requires precise calibration and tuning of control gains to achieve optimal performance[8].

2.3.2 LQR (Rezero)

An LQR controller is a full-state feedback optimal control law, u=-Kx that minimises a quadratic cost function to regulate a control system. This controller uses many parameters of the dynamics of robots to accurately model them.

Rezero uses an LQR aimed to achieve high-performance motion and dynamic stability and various speeds and manoeuvres. It develops a full 3-dimensional model of the Ballbot to capture the dynamic interactions and coupling effects between system components. It converts these dynamics found into a state space format to apply LQR effectively, with the design focusing on balancing the state deviations of position and velocity and control efforts of the motor torques. It utilises quadratic terms for both state errors to derive its optimal control law for the Ballot. The controller is implemented through MATLAB, ensuring it is robust and responsive, which adjusts the torques generated by the motors to control the ball's motion directly using inputs calculated from the model. The sensor data on this robot was measured using a 6DOF IMU that calculated the Euler angles and angular rates of the body, and the data was filtered using a Kalman filter with low and high passes to estimate the gyroscope values more accurately for use in the control system. This system is validated through a range of tests in simulations and the real world showing its effectiveness in managing complex trajectories and dynamic tasks. It has enhanced ability and precision compared to the previous controller and offers more robustness against the nonlinear dynamics of the system. However, an LQR controller requires more costly components due to the detail of control, it has high needs in terms of computational requirements and tuning. Also, the effectiveness of the LQR model heavily depends on the accuracy of the underlying model[7].

2.3.3 Comparative Analysis

In comparison to the PID and LQR systems, PIDs are generally simpler and easier to implement, making them suitable for environments with predictable dynamics, whereas LQR controllers offer better performance in handling more complex environments at the cost of high computational power. Both controllers play roles in advancing ballbot technology by enhancing its ability to navigate and perform tasks in dynamic settings.

2.4 Robust Control Techniques

This section will discuss robust control techniques used to enhance the performance and stability of Ballbots. It will discuss the combination of various control systems to improve the adaptability and effectiveness of the controllers. Robust control is a necessity in environments where operational conditions can vary drastically and unpredictably, robust control systems are designed to handle uncertainties and external disturbances effectively, ensuring consistent performance.

Fuzzy logic control strategies will be discussed where they deal with reasoning that is approximate rather than fixed and exact, this is done to mimic human decision-making and can be effective in control systems where fixed inputs are hard to obtain. Reinforcement learning will also be discussed, which is a type of machine learning where an agent learns to behave in an environment by performing actions and seeing results. This type of control is especially useful in dynamically balancing applications, for tuning control policies based on feedback received from the robot's interactions with the surrounding environment. This section aims to show how more advanced techniques for robustness can be implemented to develop more sophisticated and adaptive control systems for Ballbots.

2.4.1 Fuzzy Logic

Fuzzy logic is designed to solve problems by considering all available information and making the best possible decision given the input, this is especially useful when combined with a more conventional controller like the PID. This is done in a paper which uses fuzzy logic control to self-adjust the PID parameters of a ballbot based on real-time error feedback which enhances the adaptability and response of the system. It describes a ballbot with a previously determined PID controller implemented and adds in a system that uses predetermined fuzzy sets and rules to interpret the error and its rate of change, leading to adjustments in controller parameters. Using a fuzzy logic system reduces the overshoot of the control system and stabilises the robot more quickly than the conventional PID controller. This method was only validated through simulation studies, however designing, and implementing a fuzzy logic system requires careful consideration of the system's operational dynamics and the desired performance criteria. The complexity of this model also adds to the computational load, which could impact the controller's performance and could also be expensive[3].

2.4.2 Reinforced Learning

Reinforcement learning (RL) is a type of machine learning that enables a digital agent to learn in an interactive environment through trial and error. This can apply to the BBR system as when building a complex control model for the robot it can be difficult to model a robot to be dynamically stable in any environment. RL can greatly improve the adaptability of the robot so it can endure the toughest environments. This agent follows the Markov Decision process, where at each timestep the agent is in a state and takes an action, the environment provides a scalar reward and the next state. This behaviour is determined by a policy which maps these states to a probability distribution over actions. Policy learning can be divided into two categories: Stochastic and Deterministic. Stochastic policy learns the conditional probability of taking an action given that it is in a state. Deterministic on the other hand, does the opposite, where it learns the action as a function of the state, this means that it gives a deterministic action at each state. The agent sims to learn a policy to maximise the expected return, this was paired with a conventional model linearised controller that was able to balance a BBR, with the RL method used to improve the robot's ability and functionality in conditions it was not possible to model in the conventional model. This was shown to be effective, shown in the maximum tilt angle that could be controlled without toppling improving from 5 degrees in the conventional controller to 8 degrees in the RL compound controller. Results for showing the speed of the three motors were also displayed, showing smoother movement from the motors also reducing the overshoot from the implemented compound controller.

3 Design and Development

3.1 Introduction

This section showcases the design and development of a ball-balancing robot, to operate in dynamically unstable environments. Focusing on theoretical principles and innovations from the literature review, the design phase integrates the electromechanical and control systems to enhance the robot's stability.

This section outlines the steps taken from the initial component selection through to the final assembly of the robot. It highlights how each engineering decision was informed through the utilization of previous research conducted (in the literature review), in addition to further research and practical considerations. This section will be used as a design specification for constructing the robot, detailing the iterative development process.

The most critical part of BBR systems is the ball drive system, which is responsible for actuating the robot on the ball to maintain balance. This mechanism needed to be decided before conducting the component selection due to the significant variation displayed in the Literature review. The chosen ball drive system was one using three motors and omni-wheels placed equally around the robot's Z-axis like BallIP and Rezero's design[2], [7]. This also featured the zenith angle, ϕ, which is the angle at which the wheels are mounted to the ball, shown in Figure 4(c). This angle plays a critical role in the dynamics of the robot. This chosen arrangement has proven effective in achieving the desired functionality, manoeuvrability, and dynamic stability of the robot.

3.2 Component Selection

In the development of a ball-balancing robot, the careful selection of components is critical. Selecting the correct components not only ensures the operational efficiency of the robot but is also crucial for maintaining dynamic stability on a moving sphere.

The primary objective of the report – to demonstrate a working mechatronic system using an inverted pendulum model. This requires components that ensure high performance, reliability, and cost efficiency. The effectiveness of the control system, an integral part of the robot's functionality, depends significantly on the quality and suitability of these components.

For component selection, criteria were created and used to select each component which includes:

- **Performance** – Each component needs to meet performance criteria, including the ability to handle precise movements and rapid response times essential for balance control. This includes selecting sensors with high resolutions and motors with the necessary torque.
- **Compatibility** – Ensuring compatibility with selected components for seamless integration, ensuring each part can function cohesively without disruptions. Such as similar power requirements to simplify power regulation, making the motors and drivers compatible.
- **Durability** – Components should be able to handle operational stresses and have a long operational life to ensure the robot's longevity. Checking the load needed against the load handle by the motors and drivers.
- **Availability** – A preference was used for university-available parts due to budget constraints. Also, with time constraints in ordering parts to avoid delays to the project timeline.

- **Cost Effectiveness** – to stay under the budget of £100, utilising common components from the university wherever possible to satisfy this.
- **Scalability** – some components were also chosen with an eye towards future upgrades and to accommodate technological advancements. This would be such as a microcontroller upgrade, to faster process data in the BBR control system and higher computational load, or to increase the functionality by upgrading the motor or drivers.

To effectively manage project resources and budget, components such as omnidirectional wheels were sourced from the university's mechatronics lab, and repurposed from an old project. This approach not only saved costs but also expedited the assembly process. Additionally, a bowling ball, stepper motors and brackets necessary for mounting the components at a precise 45-degree angle were also obtained from university supplies.

Following the selection criteria and assessment of available resources, the components were categorised into four main groups to help structure the design process:

- Motors and Drivers – For actuation of the ball drive mechanism
- Sensors – Critical components of the control system to measure real-time data.
- Microcontroller – for implementing the controller designed later.
- Power supply – to power and regulate all components and ensure efficiency.

3.2.1 Motor and Driver

The selection of suitable motors was guided by a set of specific performance criteria essential for the functionality of the ball drive mechanism. These criteria were based on the need for:
High torque at low speeds: following the performance parameters outlined in BallIP, a holding torque was aimed for around 2Nm, improving on the 1.2Nm of BallIP's mechanism[8]. This torque ensures sufficient force to stabilise and manoeuvre the ballot on a spherical platform.

The NEMA 23 stepper motors provided by the university lacked a datasheet. Hence, research went through web archives to verify the holding torques, rated voltages and currents of the motors.

In searching for motors for comparisons it was decided to compare against the Project Rezero motor system, which included a Brushless DC motor combined with a planetary gear and encoder from Maxon, specifically made for the project. This is due to the functionality displayed in the robot, with high speeds achieved and the torque to handle large inclinations of up to 20 degrees. Also, compared is a Servo motor which can offer an integrated encoder and a more budget-friendly option [[9].

A stepper motor works through pulse inputs rather than analogue voltages; one digital pulse sent through a stepper driver causes one rotation, which is known as a step. This can be controlled by how much the motor moves through what's called a micro-step resolution, which is a setting micro-step drivers offer, allowing for greater control of the motor with a microcontroller. Additionally, it means an encoder to measure rotation isn't needed as stepper motors are considered precise enough to measure based on the input.

Characteristic	Stepper Motor	Brushless DC Motor
Motor Type	Stepper	Brushless DC
Model	SY57STH76-3008B NEMA 23[10]	Maxon EC-4pole DC Brushless motor with planetary gearhead[9]
Unipolar Holding Torque (Nm)	1.4	N/A
Bipolar Holding Torque (Nm)	1.89	5(with gearhead)
Rated Voltage (V)	2.6	24
Rated Current in Parallel (A)	4.2	n/a
Rated Current in Series (A)	2.1	8.0
Control Required	Stepper Driver	None
Feedback Mechanism	None (Open Loop)	Encoder attached to measure motor rotation accurately
Step Angle (degrees)	1.8	N/A
Efficiency	Moderate	High
Cost (£)	£0 (university supplied)	£1000+

Table 1 – Motor comparison table

Based on Table 1 and the project criteria, the stepper motor is the optimal choice for the initial design. This is due to its precision, cost-effectiveness and availability. The brushless DC motor, while providing superior results in the performance aspect of the ball drive mechanism, is not feasible within the budget and complexity constraints.

TB6600 microstep drivers were available from the university, which are compatible with the already selected control of NEMA 23 stepper motors. A table showing the characteristics is shown below.

Specification	**Microstep driver – TB6600** [11]
Supply Voltage	9-40 VDC
Output Current	• Up to 4.0 A (Peak), • selectable in 8 steps via DIP switch
Microstep Resolution	• 6 selectable micro-step resolutions • 200 up to 6400 steps/rev with 1.8-degree motors
Control mode	Supports PUL/DIR mode
Pulse Input Frequency (kHz)	• up to 20 kHz • Pulse input frequency: min=0, max = 20 when the duty cycle is 25 high/ 75 low, • 13kHz when duty cycle 50/50
Additional features	• Overheat and over current protection from heat sink. • Automatic Idle current protection • Has Enabled the Pin to stop/enable the motors through the microcontroller

Table 2 – Microstep driver characteristics

3.2.2 Microcontroller

To control a dynamically stable system, the microprocessor needs to be able to communicate with sensors, calculate the raw data through a control system, and send the control signals to the motors quickly, using the previously selected microstep driver. The main issue with the design is that the control system can be complex, involving some high computations and filters which can be intensive. The microcontroller needs sufficient computational power to handle intensive calculations and real-time control tasks.

Preference was given to a development platform compatible with the Arduino Integrated Development (IDE) due to its widespread use in educational contexts and abundance of open-source libraries, which accelerates development and debugging processes.

Microcontroller Criteria:

- **Pulse Width Modulation (PWM) channels**: With three stepper motors to control, a minimum of three independent PWM channels on distinct timers is essential. This allows for independent control of the stepping frequency of each motor.
- Real-time processing: The control system demands intricate computations, including feedback loops and sensor algorithms, which are computationally demanding. The microcontroller must be able to execute control algorithms and output commands to the drivers without latency.
- **Communication Channels**: Sensors usually communicate with microcontrollers typically through Serial Peripheral Interface (SPI) or I2C to read the data in the control system. Therefore, a microcontroller should have SPI and I2C ports to ensure the sensor selected will be compatible with the microcontroller.
- **Digital I/O pins**: the microcontroller should have enough digital GPIO pins to be able to control the PUL, DIR and ENA pins from each microstep driver. This requires a minimum of 9 pins including the previously mentioned PWM pins.

Specification	**SparkFun Thing Plus - ESP32 WROOM** [12]	**Arduino Uno Rev3**[13]
Digital I/O Pins	21 GPIO	14
PWM Channels	16	6
Analog Input Channels	13	6
Operating Voltage	2.3 to 3.6V	7-12V
Input Voltage (recommended)	3V	5V
Flash Memory	16MB	32KB
SRAM	520KB	2KB
Clock Speed	240MHz	16 MHz
SPI/I2C	Yes	Yes
Wi-Fi and Bluetooth	Yes	No
Dimensions	2.30 x 0.90 Inches	53.4 mm x 68.6 mm
Price	£21.50	£16.26

Table 3 - Microcontroller Comparison Table

The Arduino Uno was chosen due to its design; it came with a base plate that could easily be mounted to the base plate of the robot with nuts and bolts. It also comes with soldered pins, which simplifies the physical integration into the robot's electronic system, reducing the risk of connection errors from soldering skills. This model of Arduino Uno was available from the mechatronics workshop for use within the project for the year, giving a cost-effective solution without compromising the functionality of the microcontroller. Arduino also has a vast support community and detailed documentation of the microprocessor chip used. The ATmega328p confirms its ability to be used with 3 discrete timers on Individual PWM pins. While the ESP32 boasts a higher clock speed and more memory, the Arduino Uno's 16 MHz clock speed is sufficient for the real-time processing demands of the project's control system. The selection of the Arduino Uno reduced the learning curve of how to integrate the control system through programming, making it an ideal choice for educational projects such as this, without limiting its capabilities.

3.2.3 Sensors

For a balancing robot, the sensor system is a critical component that must provide real-time, accurate, and comprehensive data to ensure stability and responsiveness. The sensor chosen must be capable of:

- **Real Time data acquisition:** The sensor must provide rapid feedback on the robot's state to enable quick adjustments from the control system, always ensuring dynamic stability.
- **Linear Acceleration:** An accelerometer would be required for the control system. An accelerometer measures changes in speed or direction typically in m/s^2 but can also measure in terms of gravity. This is useful as it allows us to tell the position of the robot when stationary from values of acceleration in all axes (X, Y, Z).
- **Angular Velocity**: a gyroscope would be required to measure the angular velocity of the BBR body. This is useful in measuring orientation changes, which is vital to the control system.
- **Precision and Accuracy:** High precision and low drift are crucial for the sensor to detect small changes in movement and orientation.
- **Ease of integration:** This is to ensure the sensors chosen are compatible with the selected microcontroller through either SPI or I2C. Also, the ability to be powered by the microcontroller's 3.3V or 5V out pins. The use of Arduino libraries included with the sensor also helps with the integration of the sensor.
- **Durability and Reliability:** The sensor must be able to handle the continuous operation of the robot and mechanical vibrations which may occur.

Given these requirements, an Inertial Measurement Unit (IMU) was selected. An IMU is an electronic device capable of measuring both angular rates and orientation through the combination of accelerometers and gyroscopes, and sometimes magnetometers for enhanced accuracy.

The DoF of an IMU refers to the number of distinct movement axes the sensor can measure data. These axes include translation (measured by an accelerometer), rotation (measured by a gyroscope), and orientation (measured by a magnetometer). A 6DoF IMU combines an accelerometer with a gyroscope, providing 6DoF, while a 9DoF IMU has an additional magnetometer, which provides orientation data in addition to movement and rotation.

Feature	BNO055 (Adafruit 9DoF)[14]	ISM330DHCX (SparkFun 6DoF)[15]	ICM-20948 (SparkFun 9DoF)[16]
Supply Voltage	2.4 to 3.6V	1.71V to 3.6V	1.95V to 3.6V
Acceleration Range	±2g/±4g/±8g/±16g	±2/±4/±8/±16g	±2g, ±4g, ±8g, ±16g
Gyroscope Range	±125°/s to ±2000°/s	±125/±250/±500/±1000/±2000/±4000°/s	±250 °/s , ±500 °/s , ±1000 °/s , ±2000 °/s
Magnetic Field Range	±1300µT (x, y-axis); ±2500µT (z-axis)	N/A (Not applicable)	Up to ±4900 µT
Temperature Range	-40 to +85°C	-40 to +105°C	-40 to +85°C
Communication Interface	I2C (0x28 default, 0x29 alternate)	I2C (0x6B default, 0x6A alternate), SPI	I2C (0x69 default, 0x68 with jumper), SPI
Additional Features	Magnetic field resolution ~0.3µT, No SPI	Six-channel synchronized output, Embedded FIFO up to 9kB, 2x Qwiic Connection Ports	Embedded smart FIFO, On-board DMP, 2x Qwiic Connection Ports
Degrees of Freedom (DoF)	9DOF	6DOF	9DOF
Price	£34.38	£32.40	£17.30

Table 4 – IMU Comparison Table

It was decided to go with the ICM-20948 9DOF IMU because, this is the upgrade on the MPU9250 IMU, which was the common choice of sensors in past balancing robot projects. The MPU9250 has since been discontinued by InvenSense, and the ICM-20948 is the upgrade. All 3 IMUs in Table 2 provide high resolutions and ranges which would be sufficient for the control system, and all satisfy a high-temperature range, which is useful for near use of the microstrip drivers and motors. The sensor selected is also equipped with Digital Motion Processing (DMP), which offloads the computation of motion processing algorithms from the host processor, improving system power and performance [[17]. However, it does require 14kB of Flash memory on the microcontroller, which may be substantial if used with the full control system as the Arduino Uno's flash memory only is 32 kB. The ICM-20942 was also the cheapest of the sensors in Table 2 and boasted the highest range of magnetic field. Two sensors were purchased through PORF, because small sensors like IMUs can easily be damaged and combining sensors can prove useful in providing more accurate sensor data when processed. All the sensors compared in Table 3 were chosen as breakout boards to save time in making a PCB and configuring the sensors themselves, however, they did need header pins to be soldered on.

3.2.4 Power Supply

The power supply for the ballot was a critical component that required careful consideration. Given the operational context of the robot space in the mechatronics lab, it was decided to utilise

a tethered power solution rather than develop a battery system to power the robot. This approach was decided by several key factors:

- **Simplicity:** Using a direct power supply, reduced the complexity of implementing a battery system incorporating charging circuits, battery management and power level monitoring. This allowed the focus to remain on the primary objectives of the project.
- **Cost-Efficiency:** High-capacity batteries capable of delivering 24V and 6A to accommodate the stepper drivers, along with the reduced power requirements of the Arduino and sensors, would be a substantial cost.
- **Reliability:** a constant power source ensured there would be no interruption from a depleted battery, which is good for controller testing.
- **Safety:** Using a power supply eliminated the risks of batteries, such as overheating and potential chemical leaks

A 4m coil cable with heat shrink was selected for its ability to provide a secure connection to the power supply. This will come out of the top plate centre hole of the robot and is referred to as the umbilical cord; there is also enough length for the robot to manoeuvre without restriction from the length of the cable and rather the laboratory environment.

3.2.5 Emergency Stop

Following the implementation of the power supply umbilical cord, an emergency stop needed to be implemented to satisfy the project's risk assessment and safety form. For safe use of the robot, it needed to have an emergency stop fly lead, which, when pressed, would power off the robot completely in case of malfunction or safety hazard. The emergency stops also needed to satisfy the power requirements of the robot (24 V, 6 A), and have a clear and readily accessible stop button which would sit flat on a table to ensure the button can be pressed easily.

An emergency stop from Schneider was obtained from the electronics lab, which satisfied the power requirements, it was single-pole single-throw (SPST) and would be connected to the live wire of the power coil. The emergency stop needed to be fitted inside a housing with grommets and glands to ensure the power coil didn’t come loose under tension.

3.3 Mechanical Design

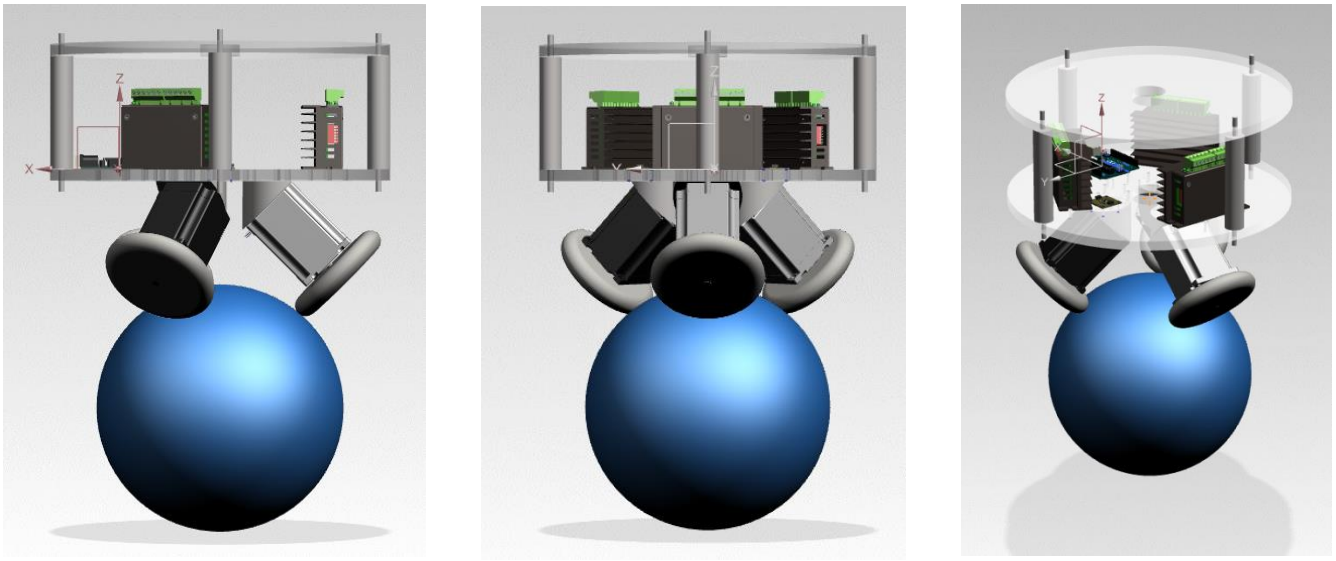
The mechanical design of a robotic system is vital for the integration of structural components, with electromechanical components, to achieve the desired functionality. For a BBR, the mechanical design was pivotal for dynamic stability, durability and integration with the control and electrical systems.

There was a base plate and top plate both made of acrylic; 10mm thick and 300mm in diameter with a 35mm diameter hole in the centre to allow wires through. The previous project the stepper motors were obtained from also included the base and top plates and 4 supports to connect them, placed equally around the radius of the plates. The acrylic material allowed for easy and precise machining, which was useful for the placement of components. The design of the baseplate was aimed to maintain the centre of mass of the robot when assembled.

In designing the baseplate, the stepper motors and mounts were already mounted to it at the desired position and angle of 120 degrees from the centre and at 45 degrees to the ball. Similarly, the TB6600 microstep drivers were arrayed with the same consideration for mass distribution by

placing them above their respective motors. The microcontroller was low in weight in comparison to the motors and was machined to mount at the back of the robot. The two IMU sensors were also placed at an equal radius apart from the centre of the baseplate. This is useful as it allows the data from the sensors to be combined and used together. All components were functionally placed but also easily accessible for maintenance and adjustment.

An integral aspect of the design was the consideration of the wiring network that would interconnect the various components. There was space for orderly cable routing, thus reducing potential damage during the robot's movement.



(a) Side View (ZX) (b) Front View (ZY) (c) Trimetric View

Figure 5 - CAD Design of Ballbot

A CAD model of the full assembly of the robot on the ball is provided in Figure 5.

3.4 Electrical Design

The electrical design is a crucial aspect of the project as this is what the power, control and data-processing systems go through to be able to keep the robot dynamically stable.

3.4.1 Power Supply System

The power system is designed to efficiently distribute power to all components of the ballot. A High capacity 24V and 6A power supply was selected to meet the high-power requirements of the stepper motors, which are the major power consumers in the system. This supply ensures that the motors operate at optimal torques when necessary. The sensors are powered through the Arduino, so they only need to be wired to the Arduino's GND and 3.3V pins.

- **Power Distribution**: to regulate the power supply to the Arduino, which operates between 7-12V, an LM7812 Voltage regulator chip was incorporated. This regulator steps down the voltage from 24V to a safe operating voltage of 12V for the Arduino. The integration of the chip on a Veroboard allowed for streamlined power distribution of the robot.
- **Wiring and Connections**: Power connections for the stepper drivers and the Arduino were combined using a large terminal block in the robot's chassis. This approach minimises wiring clutter and improves the reliability of the electrical connections.

- **Safety Measures**: the inclusion of an emergency stop within the umbilical cord addresses the safety concerns by allowing for the shut off power to the robot's components. This also enhances the user's control over the robot's operation.

3.4.2 Motor and Driver Integration

The stepper motors were decided to be configured in a bipolar series arrangement to maximise the torque and the lowest current for the motor to improve efficiency. This wiring configuration is beneficial for reducing the power consumption of the motors without reducing the mechanical output.

The TB6600 has a DIP switch for microstep resolution and current output, its current was set to 2A and the microstep resolution can be adjusted in the control system.

3.4.3 Control Electronics

The Arduino UNO R3 is the core of the control electronics as it will process all the sensor data and control the motors via the TB6600 stepper drivers.

Motor Control Integration

Each TB6600 stepper driver requires two digital pins on the Arduino for operation: one for Pulse (PUL) and one for direction (DIR). The Arduino uses its ATmega 328P microprocessors' built-in PWM capabilities to control these inputs:

PWM configuration: The ATmega 328P has 3 distinct timers that manage pairs of PWM outputs across six pins, these are as follows[18].

- Timer 0: Controls pins 6 and 5.
- Timer 1 Controls pins 9 and 10.
- Timer 2: Controls pins 11 and 3.

For the BBR, each motor needed to be connected to a pair of PWM pins that share a timer, allowing for precise control over the stepping frequency. This setup is crucial to the control system as it allows each motor to move at different speeds to each other, this is especially useful in controlling the robot's movements and balance.

Sensor Integration

The integration of the sensors with the Arduino is achieved using the I2C communication protocol, chosen for its simplicity and efficiency:

- I2C communication: unlike SPI, I2C requires only two wires (SDA for data and SCL for the clock), simplifying the wiring and reducing the potential of connection errors. Using I2C also changes the address configuration of the sensors to be able to distinguish between the two sensors on the same bus.
- Address configuration: The sensors have an ADO pin which when sent a logic low or high signal can change the I2C address.

3.5 Controller Design

For this project the challenge of balancing the robot on the ball was broken down into two inverted pendulum control problems in the ZX plane and ZY plane of the robot, shown in Figure 5(a) and (b). This is to simplify the system from a 3-D model into two 2D control problems,

allowing for the use of a PID controller which is easier to implement in 2D dynamics. This also follows models of BBRs in the Literature Review section. The IMU sensors on the robot measure the linear acceleration, angular velocities, and magnetic field strength of the robot. These values needed to be converted into Roll, Pitch and Yaw angles through an IMU filter to be used as the control input to the system. Using the angles about the X and Y axis, velocities for the mechanism in the X and Y direction can be calculated to return the system to its desired dynamic equilibrium atop the ball.

A PID controller was selected due to familiarity. Control loops were used to calculate the control output and the speed of the motors connected to the omni-wheels in contact with the ball. Before deriving the equations in 2D the following assumptions were made [19]:

- **Independent vertical planes:** When modelling in two-dimensional planes, the vertical planes are assumed to be independent.
  Friction: It is assumed that besides static friction, all other types of friction are negligible.
- **No slip:** It is assumed there is no slip between the ball and the omni-wheels and the ball and the floor.
- **Horizontal Floor:** It assumes the floor on which the ball moves is perfectly horizontal.

3.5.1 Sensor filter

The IMUs selected provide 9 axes of raw data from the accelerometer, gyroscope, and magnetometer. To convert them into Euler angles (Roll, Pitch, Yaw), the sensor first had to be put through sensor fusion, this is a method of combining the readings to reduce noise and minimise calculation error. The chosen method of sensor fusion was Sebastien Madgwick's Orientation Filter, seen in Figure 6 below.

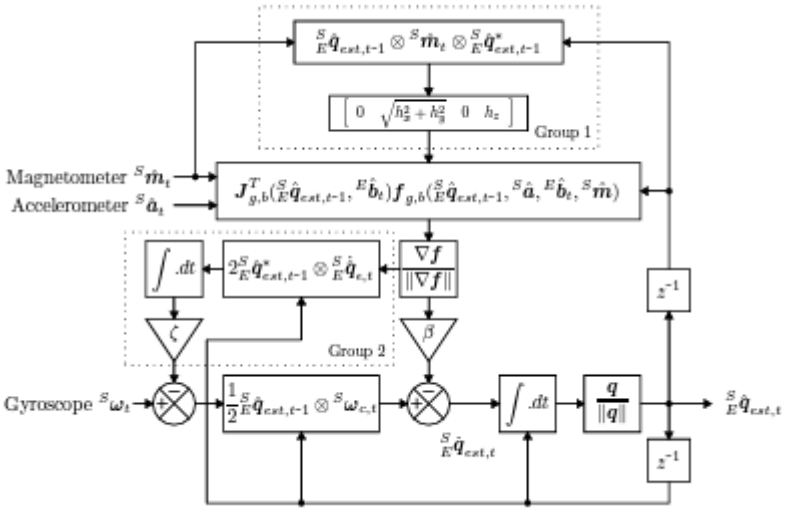


Figure 6 – Madgwick Orientation Filter Algorithm[20]

This sensor algorithm inputs the data from the three measurement tools and outputs a quaternion, which is a four-dimensional unit vector. This quaternion can be used to find Roll, Pitch, and Yaw using the following equations:

$$Roll = \theta_x = arctan\left(\frac{2(q_0 q_1 + q_2 q_3}{1 - 2({q_1}^2 + {q_2}^2)}\right)$$

$$Pitch = \theta_y = arcsin\big(2(q_0q_2 + q_3q_1)\big) \quad (1)$$

$$Yaw = \theta_z = arctan\left(\frac{2(q_0q_1 + q_2q_3}{1 - 2({q_1}^2 + {q_2}^2)}\right)$$

This method was chosen because it is said to be more accurate than a Kalman filter and is one of the only methods found to combine data from a 9DOF IMU to calculate RPY angles as accurately as possible less than a degree of error) [[20]. A Madgwick filter will be implemented in the control system through an Arduino Library.

3.5.2 Mathematical Model of Controller

Following Figure 4's model, ωx, ωy, ωz represent the angular velocities of the ball with respect to its axes. These angular velocities can be related to the horizontal velocities of the ball via:

$$v_x = \omega x \times R$$
$$v_y = \omega y \times R \quad (2)$$

Let v_{s1}, v_{s2}, v_{s3} represent the velocity input of each motor. Following the literature review the input of each motor with respect to each axis is shown below using these equations from TGU[8]:

$$v_{s1} = -v_y cos\phi + Kz\omega z$$
$$v_{s2} = \left(\frac{\sqrt{3}}{2}v_x + \frac{v_y}{2}\right)cos\phi + Kz\omega z \quad (3)$$
$$v_{s3} = \left(-\frac{\sqrt{3}}{2}v_x + \frac{v_y}{2}\right)cos\phi + Kz\omega z$$

Where $Kz = -r\sin\theta$ is the coefficient for rotating the ball in a fixed position, about the Z axis and is the zenith angle previously described. The main goal of the project is to balance the robot on the ball so $\omega z = 0$

It is assumed that the inverted pendulum in the ZX and ZY planes are equal, which allows the PID parameters to be equal for both controllers. Where error, $u_{x,y}$, is measured in terms of pitch θ_y and roll θ_x, the following PID control equations are shown below. $\omega_{x,y}$ is measured by the IMU gyroscopes directly.

$$u_{x,y}(t) = K_p \cdot \theta_{x,y}(t) + K_i \int \theta_{x,y}(t)dt + K_d \cdot \frac{d\theta_{x,y}}{dt}$$
$$u_{x,y}(t) = K_p \cdot \theta_{x,y}(t) + K_i \int \theta_{x,y}(t)dt + K_{dx} \cdot \omega_{x,y}(t) \quad (4)$$

Following these motor speeds worked out need to be converted into PWM frequencies as a function to move the motors at different speeds independently. So, the following equations were derived:

$$MotorSpeed(RPM) = \left(\frac{v_{s1,2,3}}{2\pi}\right) \times 60$$
$$f_{step} = \frac{Microsteps\ per\ Revolution\ \times\ Motor\ Speed\ (RPM)}{60}$$
$$Microsteps\ per\ Revolution\ = \frac{200}{m}$$

Where $m = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ depending on the Microstep Resolution setting of the stepper drivers.

These equations will be used as a function in implementing the PID controller motor outputs by changing the frequency of each Arduino Timer. However, this may not be sufficient for the control system as changing the ATmega PWM registers directly only allows the change of Timer Clock Frequencies by dividing the Clock Frequency of 16MHz by the set prescale values of 1, 8, 64, 256, or 1024.

4 Methodology

4.1 Introduction

A systematic approach to the implementation, testing, and calibration of the BBR control system is outlined in this methodology. The full scope of the planned methodology could not be carried out because of constraints on time and resources. Therefore, this section will go into detail about the actual procedures conducted and the methodology that was intended for the complete development of the BBR.

The initial testing included the activation and control of robot motors on the floor, as well as the calibration of sensors and the implementation of a Madgwick filter to measure their orientation accurately. To verify the functionality of the individual components and the integration of the sensor system with the microcontroller, these preliminary tests were essential.

Subsequently, this methodology was planned to continue with a series of systematic steps aimed at refining the robot's control systems to achieve dynamic balance on a ball. The described process will provide a framework for future work to achieve the project objectives, despite the implementation phase of the PID controller not being completed.

4.2 Construction of Robot

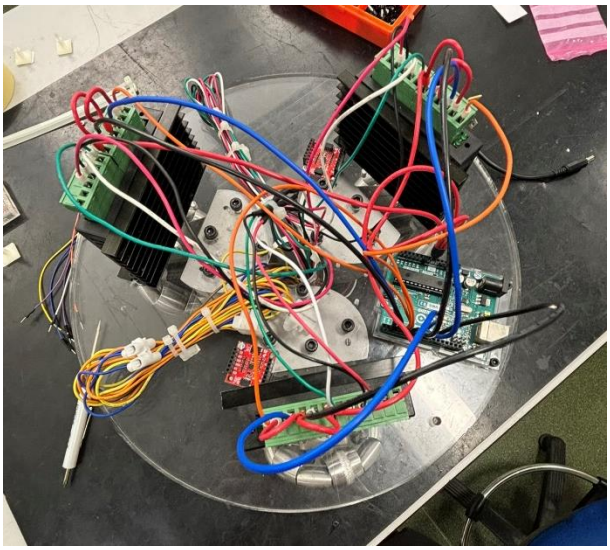
This section briefly outlines the key steps taken in assembling the components of the Ballbot, following the guidelines set in the mechanical and electrical design specifications. The assembly focused on integrating the Arduino UNO, three TB6600 Microstep drivers, and two ICM-20942 IMUs onto a machined baseplate. Full assembly of the robot is shown in Figure 7(b).

4.2.1 Baseplate Modifications

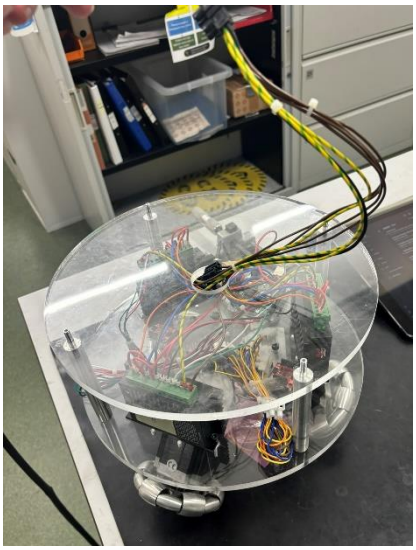
The baseplate needed holes to be drilled to mount the components. The placement of the drivers was described in the mechanical design section and was implemented. The IMUs needed to be precisely machined and there was difficulty in milling the holes In the right orientation(horizontal) to align with the desired robot axis but it was accomplished using a milling machine ensuring both were placed in equal radius about the centre of the plate. Finally, the Arduino was mounted in the desired position by clamping the Arduino's plastic base to the baseplate and drilling holes through with a power drill which simplified the process. All components were mounted using standard nuts and bolts, apart from the drivers which were done with wingnuts.

4.2.2 Wiring Implementation

Electrical connections were carefully planned following design specifications to maintain a clean layout, with wires anchored (shown in Figure 7(a)) in strategic locations to preserve the robot's centre of mass. A full wiring diagram of all the components is included in the appendix.



(a) Baseplate Modification



(b) Ballbot Assembled

Figure 7 – Assembly of the BBR.

4.3 Initial Testing and Validation

4.3.1 Motor Control Test

Objective: This test is to ensure that each motor responds correctly to PWM signals.

Method: With the robot placed upside down on its top plate (shown in in Figure 8(a)), PWM signals were sent through the Arduino IDE to each motor individually, this setup was chosen to prevent any damage or uncontrolled movement of the robot during testing.

Procedure:

- Code was uploaded to the Arduino which generated a PWM signal.
- The signal frequency was increased to observe the motor's response at different step frequencies.
- The motors were observed for consistent motion and any signs of stuttering or irregular movement which could indicate issues with the motor or driver.

Results: Each motor's response was documented, noting any deviations from expected behaviour.

Conclusion: The tests were used to confirm motor functionality before proceeding to more advanced tests.

4.3.2 Straight Line Movement Test

Objective: This test was to assess the coordinated response of the motors in enabling the robot to move in straight lines. This test will help the omnidirectional movement of the robot on the ball without the balancing aspect.

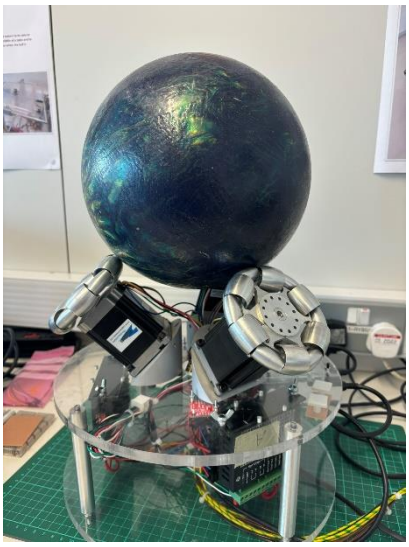
Method: The Robot, placed on flat ground in a safe environment (shown in Figure 8(b)), receives single coordinated steps for each of the motors in pairs, intending the robot to move in straight lines at angles: 0^o,120^o,240^o from the robot's centre. The code was adjusted to make each pair of wheels move the robot forward at each angle through the pressing of W, A or D on the keyboard in the serial monitor of Arduino.

Procedure:

- Each pair of motors was activated through the pressing of W, A or D to move at 0^o, 120^o, 240^o in a straight line.
- The motors were measured to verify movement in straight lines along the desired angles.
- The speed and acceleration for each movement direction were also observed.

Results: The results from each movement were documented, with emphasis on the robot's ability to maintain set trajectories. Any tendencies to veer off the path were noted, and each angle was visually inspected for linearity.

Conclusion: The results from this test will be used in more complex motor control algorithms. Any deviations from behaviour would require troubleshooting of mechanical alignment or software control.



(a) Motor Control Test

(b) Straight Line Test

Figure 8 – Rigs for initial motor testing.

4.3.3 Sensor Calibration and Data Validation Test.

Objective: To measure the raw data readings from the sensors when the robot is stationary to calibrate the IMUs.

Method: Use the ICM-20942 Arduino library examples to measure the raw data output from each IMU when stationary.

Procedure: The Raw data was measured over a set time of 30 seconds to measure any change to establish baseline sensor readings.

4.3.4 Orientation Data Accuracy Test

Objective: To implement the Madgwick filter through an Arduino library to obtain RPY angles using a 9DoF IMU. Measure the RPY angles when the robot is stationary and rotated about each axis.

Method: Compare the Madgwick filter's RPY angles to known orientations.

Procedure: The robot is manually rotated about each axis, and the RPY outputs are recorded and compared against the actual angle of rotation to determine accuracy.

4.3.5 Magnetometer Orientation Response Test

Objective: To evaluate the magnetometer's accuracy in detecting changes in orientation around the robot's Z-axis.

Method: Analyse the magnetometer's raw data to see changes under rotational movement around its Z-axis. Calculate the heading to measure Yaw from magnetic north.

Procedure:

- Set the Robot up in an environment with minimal magnetic interference.
- The robot is then rotated around its Z-axis by 90^{o}.
- Magnetometer readings are recorded and analysed under rotation to measure the sensor's ability to accurately track orientation.

Conclusion: This test will measure the response of the magnetometer to orientation and will determine if it can be used in measuring Yaw from magnetic north.

4.4 Controller Implementation

To implement the designed controller, it needed its parameters to be tuned. This was planned to be done manually on a fixed ball first. So, the bowling ball would need a 3D-printed rig to fix It in place and prevent rotation. This is useful as it allows the robot to be tuned to the ball when stationary, so the only error it must oppose is from slip and no rotation of the ball at any angle.

4.4.1 Omnidirectional Drive System

Objective: To be able to input an angle through the serial monitor and the robot move in linearly in that angle a set distance.

Method: Create code that implements the mathematic equations in (3) so that the robot can move omnidirectionally on the floor from any angle 0-360^{o} measured from the ZX positive plane. Also, include the control of the stepping frequency as an output from the motor equations to relate it to a function of the velocity in y and x.

Procedure:

- Implement the code so that an input angle can be entered through the Arduino's Serial Monitor.
- On the floor draw on a cardboard a large protractor to measure angles.
- When the input angle is entered, measure the line moved on the floor.

Results: Following this procedure should help measure the error from the input angle to the robot and the actual angle moved

4.4.2 PID Tuning on Fixed Ball

Objective: To be able to balance the robot on the ball when fixed through the tuning of the PID parameters of the controller.

Method: Implement the controller design code derived in section 3.5, implementing all Arduino libraries such as the ICM-20942 library, and Madgwick Sensor Fusion to combine the sensor data with the controller design.

Procedure:

- Place the ball in the Static Testing Rig to prevent rotation of the ball.
- Place the robot on top of the fixed ball.
- Assume parameters of Kp, Ki, and Kd and do trial and error until the robot remains balanced.

Results: Following this procedure of the modification of PID parameters, the robot should remain balanced on top of the ball

4.4.3 PID Tuning in Dynamically Unstable Environment

Objective: To determine the PID parameters to balance the robot on the ball when free.
Method: Start with the PID parameters of the fixed ball and modify them through trial and error to obtain the PID parameters for the robot when free.

Procedure:

- Place the ball in an environment that follows the Risk Assessment where there is a defined 2m by 2m area layer out.
- Place the robot on top of the ball and stand nearby to catch the robot if topples.
- Modify PID parameters through trial and error to achieve dynamic stability of the ball.

Results: Following the procedures outlined, it should be able to obtain the PID parameters to balance the robot on the ball in a dynamically unstable environment.

5 Results

5.1 Motor Control Test Results

5.1.1 Motor Response to PWM Signals

Objective: Verify each motor responds correctly to PWM signals.

Findings: Each motor responded as intended, displaying consistent motion without any signs of stuttering, or irregular movements. Indicating the drivers and motors are integrated well and functioning as expected.

5.1.2 Straight Line Movement Test

Objective: Assess the coordinate response of the motors in enabling the robot to move in straight lines.

Findings: The robot was able to follow the path accurately at the set angles, paths traversed were linear and little deviations from expected trajectories. Deviations included different accelerations

of motors resulting in slightly off angles (less than 3 degrees). Demonstrating effective coordination between the motors and control system.

5.2 Sensor Data Analysis

5.2.1 Sensor Calibration and Data Validation

Objective: calibrate sensors and validate the data when the robot is stationary

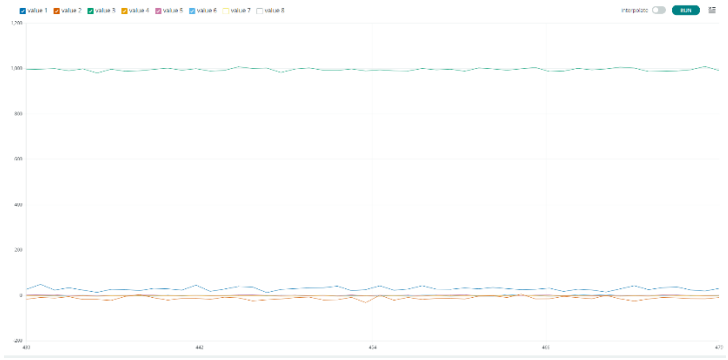


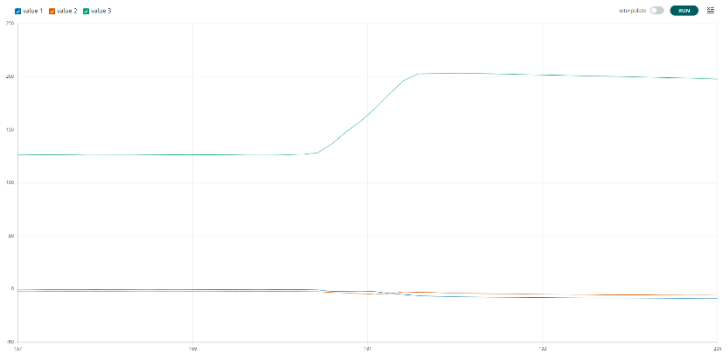
Figure 9 - Accelerometer and Gyroscope combined raw data when robot is stationary.

Findings: Figure 9 shows that the acceleration values in X and Y tend around 0, showing there is little to no acceleration in these axes. The green line on the graph is the accelerometer data in Z, showing the sensor measures acceleration in this axis. All the angular velocities are around 0, these findings affirm sensors stability of the accelerometer and gyroscope.

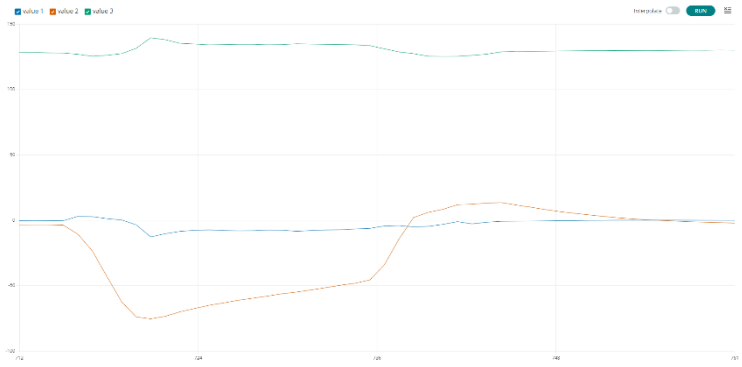
5.2.2 Orientation Data Accuracy

Objectives: Validate the accuracy of the orientation data obtained through the Madgwick filter.

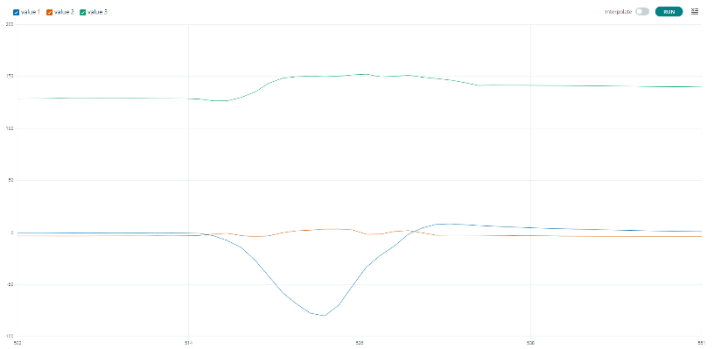
Findings: Figure 10(a) shows that the sensor measured the change in rotation around the Z-axis accurately. Figure 10(b) shows that the sensor measured the change in rotation about the Y-axis accurately. Figure 10(c) shows that the sensor measured change in rotation about the X-axis accurately. Figure 10(d) shows that the yaw angle when stationary decreases over time and tends to around 122 degrees. Suggesting potential drift that may need further investigation or calibration.



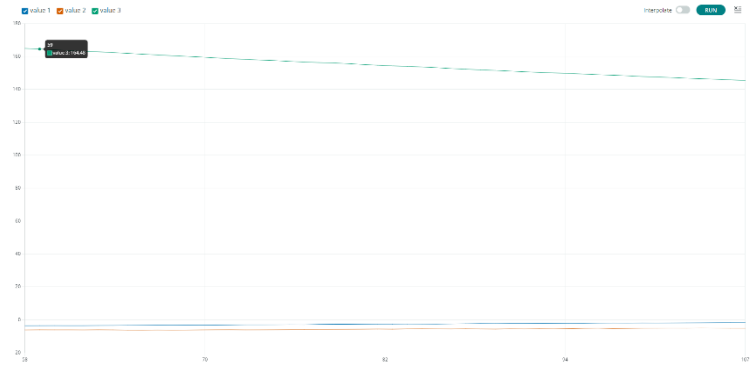
(a) Rotation 90 degrees about Z-axis.



(b) Rotation -70 degrees about Y-axis



(c) Rotation 80 degrees about the X-axis.



(d) Stationary

Figure 10 - RPY angles when stationary and under rotation.

5.2.3 Magnetometer Orientation Response Test

Objectives: Evaluate the response of the magnetometer data under rotation about the Z-axis.

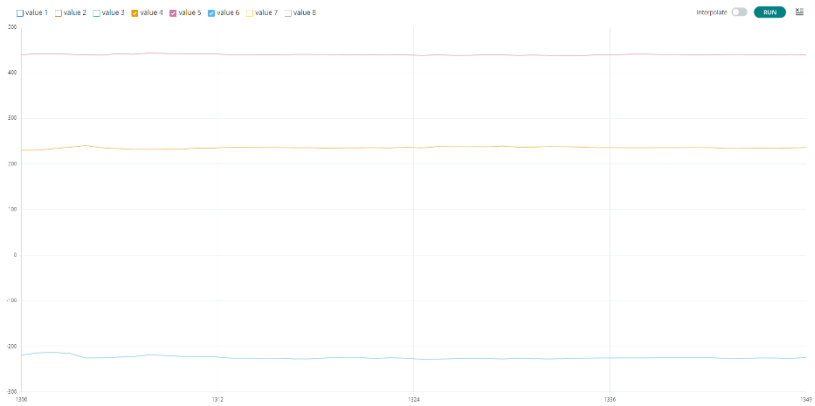


Figure 11 - Magnetometer data under 90 rotation about Z axis.

Findings: As shown in Figure 11, when the robot is rotated by 90^{o} the magnetometer data didn't change.

6 Discussion

6.1 Overview of Testing Outcomes

A series of tests to verify the performance and integration of each critical component have been carried out on the BBR. The focus ranged from the ability of the robots to respond mechanically, as well to intricate sensor feedback systems, essential for the robot's dynamic stability. The basic tests included the assessment of motor responsiveness to PWM signals, as well as the evaluation of their precise linear motion capabilities, and the analysis of sensor accuracy using Madgwick's filter for orientation data.

6.2 Motor Performance Analysis

The motor tests have shown a significant degree of responsiveness to PWM signals, which is a key factor in the precise control of balancing robots. Each motor demonstrated consistent motion without stuttering, confirming its reliability as part of the project. Nevertheless, during the straight-line tests which highlighted the challenge of synchronizing several motors to coordinate their movements, there were minor deviations in acceleration. These discrepancies even slight, outlined the need for a robust control algorithm capable of maintaining such variations to maintain balance.

6.3 Sensor Performance Evaluation

The Sensor data proved to be the most accurate, with the Madgwick filter effectively translating the IMU raw data into usable Roll, Pitch and Yaw angles. However, the magnetometer test and the RPY when stationary showed that the Yaw value drifted over time, this drift could significantly impact the robot's ability to maintain orientation over time. The roll and Pitch angles seem highly accurate, which is extremely useful in this project's controller which only depended on Pitch and Roll. This drift in Yaw suggests that a more sophisticated sensor fusion algorithm may need to be implemented or better calibration of the magnetometer.

6.4 Integration of Control System

The integration of the control system was only partially completed, with the initial tests on motor control and sensor validation successfully conducted. However, the full implementation of the PID control to achieve dynamic balance was not realised within the project timeline. This incomplete aspect highlights the complexities involved in tuning control systems for dynamically unstable robots, where theoretical calculations must meet practical application challenges. Future efforts should focus on fixing the ball to be stable for the initial PID tuning to isolate the variables affecting balance such as slip.

6.5 Implications for Future Work

The inability to test the robot under dynamic balancing conditions leaves a significant gap in the project's overall validation. Future projects could benefit from the construction of a stable testing rig to allow for tuning of the robot in a controlled environment and then after the completion of this, the project could undergo some dynamic environment testing, to let the BBR balance freely. Alternatively exploring different electronic components such as microcontrollers due to the fact its

PWM timers were restricted by the set prescale values of the ATmega microprocessor. Another thing to consider is swapping the power supply system for a battery management system, this is to improve the functionality of the robot as it would not be restricted by the 4m umbilical cord.

7 Conclusion

7.1 Project Overview

This project aimed to design and develop a ball-balancing robot capable of remaining dynamically stable on a moving sphere by the integration of complex electromechanical design and control systems. Despite the challenges, significant progress was made in designing the robot's core systems and conducting initial tests that demonstrated the effectiveness of selected components and strategies. This project also highlighted critical areas for improvement, particularly in sensor accuracy and controller implementation.

7.2 Achievements

The project's successes include the successful design and assembly of a functional electromechanical system. The motors and sensors operated as expected through various tests, such as the motor control test which verified the motor's speed could be controlled by changing the frequency of the PWM signals. The implementation of Madgwick's filter was validated through testing of orientation measurements to confirm accuracy in Roll and Pitch, and the straight-line movement tests proved the robot could manoeuvre with a reasonable degree of accuracy.

7.3 Challenges

Also, the project faced significant challenges, mainly in the implementation of the PID controller for dynamic balancing. The intended tests for dynamic balancing were not conducted, leaving a substantial part of the robot's functionality invalidated under the intended operating conditions. Also, the Yaw angle calculated through Madgwick's filter showed that it was not accurate as it was subject to drift.

7.4 Recommendations for Future Work

- Future projects should focus on the completion and fine-tuning of the PID control system. Implementing a fixed rig for initial tests could isolate variables such as slip.
- Sensor Improvements: Look to address the drift in yaw by enhancing the sensor fusion algorithms or refining the magnetometer calibration techniques.
- Power system upgrade: explore the integration of a battery management system to replace the tethered power supply for enhanced mobility of the robot.
- Component Upgrades: Investigate alternate microcontrollers and components that can provide greater control over PWM configurations to improve the robot's balancing capabilities.
- Testing Rig Construction: Build a stable testing rig for the controlled tuning of the controller parameters while static before attempting dynamic balance tests.

8 References

[1] A. Bonci, 'New dynamic model for a Ballbot system', *MESA 2016 - 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications - Conference Proceedings*, Oct. 2016, doi: 10.1109/MESA.2016.7587176.

[2] M. Kumagai and T. Ochiai, 'Development of a robot balancing on a ball', *2008 International Conference on Control, Automation and Systems, ICCAS 2008*, pp. 433–438, 2008, doi: 10.1109/ICCAS.2008.4694680.

[3] X. Su, C. Wang, W. Su, and Y. Ding, 'Control of balancing mobile robot on a ball with fuzzy self-adjusting PID', *Proceedings of the 28th Chinese Control and Decision Conference, CCDC 2016*, pp. 5258–5262, Aug. 2016, doi: 10.1109/CCDC.2016.7531938.

[4] U. Nagarajan, 'Dynamic Constraint-based Optimal Shape Trajectory Planner for Shape-Accelerated Underactuated Balancing Systems.', *Robotics: Science and Systems*, vol. 6, pp. 243–250, 2010, doi: 10.15607/RSS.2010.VI.031.

[5] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, 'A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive', *Proc IEEE Int Conf Robot Autom*, vol. 2006, pp. 2884–2889, 2006, doi: 10.1109/ROBOT.2006.1642139.

[6] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, 'A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive', *Proc IEEE Int Conf Robot Autom*, vol. 2006, pp. 2884–2889, 2006, doi: 10.1109/ROBOT.2006.1642139.

[7] P. Fankhauser and C. Gwerder, 'Modeling and Control of a Ballbot', 2010. doi: https://doi.org/10.3929/ethz-a-010056685.

[8] M. Kumagai and T. Ochiai, 'Development of a Robot Balanced on a Ball - First Report, Implementation of the Robot and Basic Control -', *Journal of Robotics and Mechatronics*, vol. 22, no. 3, pp. 348–355, Jun. 2010, doi: 10.20965/JRM.2010.P0348.

[9] 'maxon dc brushless motors make "ballbot" a reality | maxon group'. Accessed: Apr. 24, 2024. [Online]. Available: https://www.maxongroup.co.uk/maxon/view/news/PRESSRELEASE-ballbot

[10] Zapp Automation Ltd, 'SY57STH76-3008B Nema 23 stepper motor'. Accessed: Apr. 24, 2024. [Online]. Available: https://web.archive.org/web/20130725115431/https://www.zappautomation.co.uk/en/nema-23-stepper-motors/381-sy57sth76-3008b-nema-23-stepper-motor.html

[11] 'TB6600 Stepper Motor Driver - RobotShop'. Accessed: Apr. 25, 2024. [Online]. Available: https://uk.robotshop.com/products/tb6600-stepper-motor-driver

[12] 'SparkFun Thing Plus - ESP32 WROOM | The Pi Hut'. Accessed: Apr. 25, 2024. [Online]. Available: https://thepihut.com/products/sparkfun-thing-plus-esp32-wroom#product-reviews

[13] 'A000066 Arduino, SBC, Arduino UNO Rev3, ATmega328P | Farnell UK'. Accessed: Apr. 25, 2024. [Online]. Available: https://uk.farnell.com/arduino/a000066/arduino-uno-evaluation-board/dp/2075382

[14] ‘BNO055 9 DOF Absolute Orientation IMU Fusion Breakout Board - RobotShop’. Accessed: Apr. 26, 2024. [Online]. Available: https://uk.robotshop.com/products/bno055-9-dof-absolute-orientation-imu-fusion-breakout-board

[15] ‘SparkFun Micro 6DoF IMU - ISM330DHCX (Qwiic) | The Pi Hut’. Accessed: Apr. 26, 2024. [Online]. Available: https://thepihut.com/products/sparkfun-micro-6dof-imu-ism330dhcx-qwiic

[16] ‘SparkFun 9DoF IMU Breakout - ICM-20948 (Qwiic) | The Pi Hut’. Accessed: Apr. 26, 2024. [Online]. Available: https://thepihut.com/products/sparkfun-9dof-imu-breakout-icm-20948-qwiic

[17] ‘SparkFun_ICM-20948_ArduinoLibrary/DMP.md at main · sparkfun/SparkFun_ICM-20948_ArduinoLibrary’. Accessed: Apr. 26, 2024. [Online]. Available: https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary/blob/main/DMP.md

[18] ‘Secrets of Arduino PWM | Arduino Documentation’. Accessed: Apr. 28, 2024. [Online]. Available: https://docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm/

[19] J. W. van der Blonk, ‘Modeling and Control of a Ball-Balancing Robot’, 2014.

[20] S. M.-R. x-io and U. of B. (UK) and undefined 2010, ‘An efficient orientation filter for inertial and inertial/magnetic sensor arrays’, *forums.parallax.comS MadgwickReport x-io and University of Bristol (UK), 2010•forums.parallax.com*, 2010, Accessed: Apr. 28, 2024. [Online]. Available: https://forums.parallax.com/uploads/attachments/41167/106661.pdf

9 Appendices

9.1 Wiring diagram of components