Design Report

FEEG6013 Group Design Project

51
Robot Localisation

Design of a low cost, small scale localisation system suited to the Eurobot competition.

Project Summary:

Localisation is understanding one's position relative to their environment. This project proposes the design of a localisation system for autonomous robots featuring an innovative combination of gyroscopic, optical and ultrasonic sensors to determine its position. The system has been designed to improve performance in the Eurobot competition and has wider applications ranging from automotive vehicles to domestic robotics.

Throughout this project, various types of sensors were assessed to find the combination which provided the most cost effective solution for this application. The use of a Kalman filter, which is a technique used to fuse readings from an array of different sensors and a PID controller, used to correct any inaccurate motions as the robot traverses its path, were investigated.

To evaluate the accuracy of the platform, a bespoke test method has been designed and manufactured. It uses an overhead camera to track the actual position of the robot, which is compared to the robot's estimate.

At the conclusion of the project, a completed system using an optical sensor and PID controller has been designed and manufactured. The system is able to estimate its position to within an accuracy of 20 mm. The test method has been validated to measure the position trajectory of a robot with an accuracy of 3 mm.

Group Members:

ID Number	Name	ID Number	Name
27643417	Caroline Layzell	27758265	Alice Loneragan
27657647	Michael Leat	27055396	Justin Godden
26121093	Oliver Heilmann		

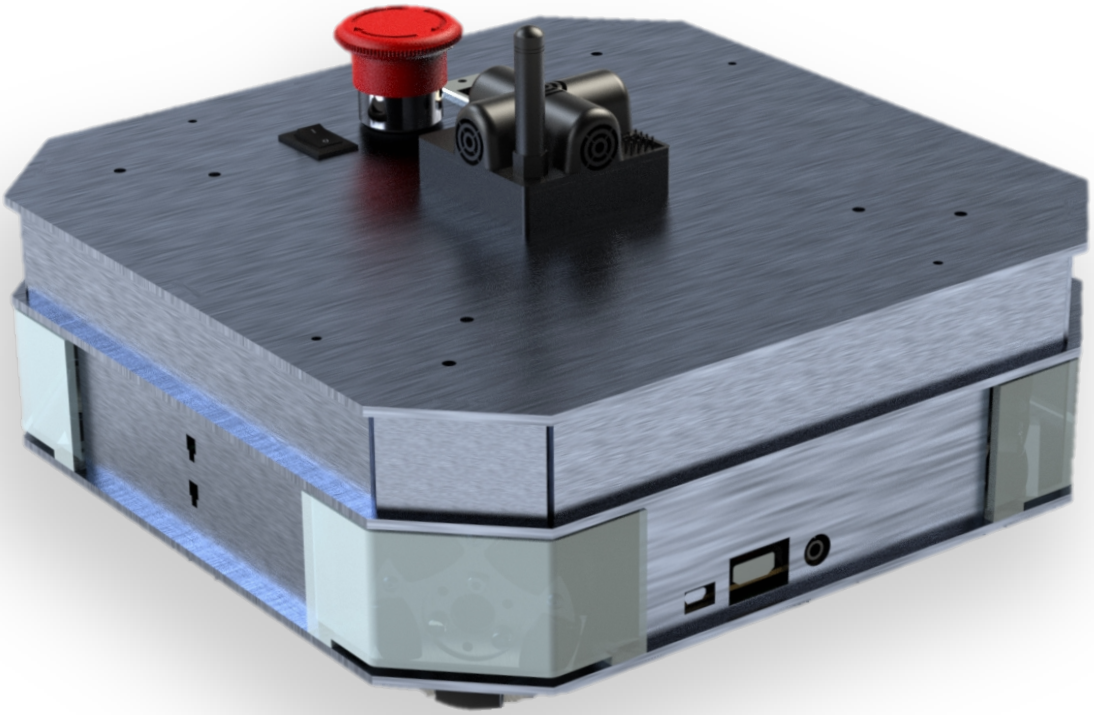
Primary Supervisor: Professor Paul White

Co-Supervisors: Professor Martyn Hill

Submitted on: 09/05/2019

OVERVIEW

Through research and testing of various sensors, a sensor fusion system has been designed and built using the most suitable sensors, and employing a Kalman filter. This system has been implemented on a robot with omni-wheels, creating a robot that knows where it is in its environment and is capable of reaching its destination with only 25 mm error. A validation method has been developed and built using a visual tracking system. This will enable the navigation system for future generations of robot to be refined.



CONTENTS

ABBREVIATIONS

CPU – Central Processing Unit
DPI – Dots Per Inch
FOV – Field of View
GUI – Graphical User Interface
IMU – Inertial Measurement Unit
ORB – Oriented FAST and Rotated BRIEF
PCB – Printed Circuit Board
PID – Proportional-Integral-Derivative
PLA – Polylactic Acid Plastic
SIFT – Scale-invariant feature transform
SPI – Serial Peripheral Interface
SSH– Secure Shell
SURF – Speeded Up Robust Features

ACKNOWLEDGEMENTS

The group would like to thank the following people and organisations, without whom our project would not have been possible.

Professor Paul White and Professor Martyn Hill for their supervision, guidance and enthusiasm throughout the project.

Dr. Stephen Prior for his advice and help securing parts and additional funding.

Alastair McDonald for his belief in the project and support enabling the testing method to be further developed.

Workshop staff for providing help and advice.

Boeing for providing sponsorship, enabling the project to be enhanced.

INTRODUCTION AND DESIGN BRIEF

THE IMPORTANCE OF LOCALISATION

The ability of an autonomous robot to determine its position and orientation with respect to its environment can be crucial to the successful application and completion of a required task. Information regarding the robot's location allows the future movements of the robot to be calculated and executed. Localisation technology is employed in many everyday items, including mobile phones, cars and domestic robots. With the recent surge in electrification and autonomy of every day technology, accurate localisation systems are becoming increasingly important.

LOCALISATION TECHNIQUES

Localisation systems can vary widely in complexity, with some systems simply measuring the rotation of the robot's wheels to calculate the distance travelled. This technique is called odometry. More robust systems use a combination of data from multiple sources in a process called sensor fusion. Sensor fusion is a method of combining readings from multiple disparate sources to give information about a system's state that has less uncertainty than reading each sensor individually. Sensor fusion is different from multi sensor integration in that it includes the combination of sensory data into one format [2].

Sensor fusion applications extend further than robot localisation, appearing in areas such as determining the state of traffic from data including acoustic and visual sources, aircraft condition monitoring and financial prediction algorithms [2].

LOCALISATION IN THE EUROBOT COMPETITION

Eurobot is a robotics competition in which teams design automated robots with the aim to score points by completing various tasks, set out on a 2 x 3 m competition board (Figure 1) within a set timeframe of 90 seconds. The tasks to be completed vary every year, but the size of the board remains the same. Teams from the University of Southampton have historical success in the UK final, with teams placing in the top 3 for the past six years. However, Southampton teams have yet to be successful on the international stage. The largest difference between the systems designed by the successful teams and the current Southampton entrants is the sophistication of the robot localisation systems.

CURRENT CHALLENGES OF EFFECTIVE LOCALISATION

The vast majority of Southampton teams enter through a module taken in Part II. The two limiting factors preventing Southampton teams from using robust localisation methods and becoming more competitive in the Eurobot international stage are budget and prior knowledge.

BUDGET

Winning Eurobot teams enter robots that cost in region of £20,000. Southampton entrants typically spend around £200 - £500 on their robots. The lack of funds available to Southampton students limits the quality and precision of sensors used. It also means that teams are limited in the number of sensors that they can trial to find the most suitable candidates.

PRIOR KNOWLEDGE

Part II students enter the Eurobot project with a limited knowledge of robotics, the only localisation system most students have experienced at this stage is odometery. The Eurobot project in second year is run for a single semester which does not give the students time to experiment with more sophisticated localisation methods. As a result, odometry is the most common localisation technique used by Southampton teams, which is known to compound error over time. Additionally, knowledge is not passed on from year to year as the teams are replaced, meaning the teams start with the same limited amount of knowledge each year.

This project aims to address the two issues by researching and testing alternative methods of localisation and developing a system that is robust, cost effective and can be used by a Southampton team to improve performance in the Eurobot international final.

EUROBOT COMPETITION REGULATIONS

To be suitable for use in a Eurobot competition, the designed system must work on a Eurobot competition board. There are certain features of a Eurobot match that are consistent every year and must be considered when designing the system.

A Eurobot match lasts 90 seconds. The localisation system must function to the desired accuracy for this length of time.

There are 3 low mounts at the side of the table and a higher mount in the top centre for the teams to install external localising systems (Figure 1). The designed localisation system must fit within the perimeter of the robot and these areas only.

Ramps, seesaws and stairs with an incline up to 12° can be featured as an obstacle on the table. The designed system must be able to traverse all areas of a Eurobot table.

The perimeter of the robot must be less than 1200 mm to meet regulations. The height must be less than 350 mm.

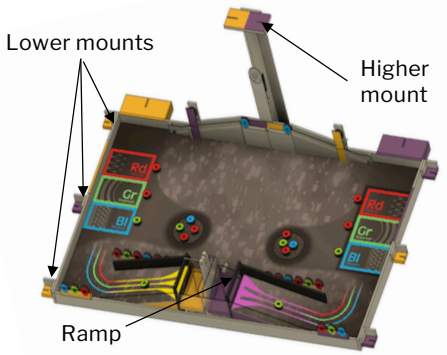


Figure 1: 2019 Example Eurobot competition board [1].

DESIGN BRIEF

The goal of the project is to create a small-scale localisation system with millimetre accuracy to improve Southampton's success at the Eurobot competition.

In order to achieve this, the specific project aims and deliverables were set out as follows:

Project Aims	Project Deliverables
Design a robotic platform that can determine its position on a Eurobot table with millimetre accuracy for 90 seconds.	A mobile robotic platform consisting of a chassis, sensors and microprocessor
Create a low cost design with respect to leading Eurobot finalists.	An evaluation of the most effective sensor combinations.
Design a system that is easily built upon and further developed by future entrants	An algorithm for combining sensors to optimally reduce error in the system.
Develop a reliable method of determining the accuracy of the system	A testing protocol to test and verify the system

STAKEHOLDER ANALYSIS

Considering the requirements of direct and indirect project stakeholders allowed the design brief to be adapted and developed where required.

Stakeholder	Responsibility	Stake in the project	Impact	What do we need from them?	Perceived attitudes / risks	Stakeholder Management Strategy
Team members (Direct)	To drive and deliver the project	Project quality and final grade	High	Cooperation, delivering tasks on time	Desire to succeed. Dependency of grade on the project outcome / success.	Heavy involvement in the development of the project. Task management systems.
Elevator Pitch Board (Direct)	Potential sponsor	Engineering department reputation	High	Funding	Southampton already perform well in Eurobot, may not be used.	Provide a short presentation to demonstrate how their funding will help the project.
RoboSoc (Direct)	End-user of the final system	Eurobot competition performance	High	Information on their project needs	Concerns that the system will be too difficult / complicated to utilise.	Listen to how they want to use the system and accommodate their requests where possible
Supervisors (Direct)	Consulting role	Reputation and project quality	Medium	Support and expertise during the project	Concerns about the achievability of the project.	Weekly meetings to provide updates and seek advice on problems concerning the project.
Part Suppliers (Direct)	Sale of parts and equipment for the project	Revenue	Low	Parts arriving on time	Risk of parts not arriving on time or being of low quality.	Ensure parts are in stock. Order parts as early as possible.
EDMC (Direct)	Manufacturer	Health and safety	Low	Design approval and manufacture	Long lead times. They are unaffected by the project's outcome.	Liaise with technicians to ensure drawings are suitable. Submit as early as possible.
Eurobot Association (Indirect)	Regulators of the Eurobot competition rules	Fair and equal competition	Medium	Provide the rules and clarification on any queries	Unaffected by project outcome but are supportive of entries to the competition.	Email, if required, with queries about the design and ensure it adheres to the regulations.
University of Southampton (Indirect)	To provide facilities, rules and support for students	Reputation and quality service as an institution	Low	Clear rules and possible support if needed	Concerns with plagiarism, missed deadlines or under performance.	Adhere to University rules and meet deadlines on-time to a high quality and standard.
Environment-al Agency (Indirect)	Regulate laws on recycling, waste and pollution	Law adherence, Health and safety	Low	Clear laws made available	Hazardous materials, poor recycling and excess waste.	Use appropriate batteries and materials, and follow strict guidelines on recycling and disposal.
Part II Students (Indirect)	None	Possible future users	Low	None	Only use if affordable/ improves results	Inform of project and benefit to their design.

Table 1: Stakeholder Analysis

DEVELOPMENT

Through meeting with key stakeholders, the design brief has been developed to reflect on their needs.

A meeting with Robosoc, a student robotics society, confirmed their interest in the project. A discussion regarding the design brief revealed the need for the robot to be as small as possible with standardised ways to attach parts, allowing for them to have more design freedom.

Through presenting at the elevator pitch another stakeholder was identified; Alastair McDonald. His interest is in developing a robust testing method viable for use in a range of future projects, and he has provided funding for its development. The design brief focus became more equal regarding the development of the system and the development of the test method.

This test method will be also available as an interactive demonstrative tool for use on open days or for events such as the *Engineering Design Show 2019*.

MARKET STRATEGY

The marketing of the robot would be aimed at student teams entering Eurobot from Southampton, similar to Robosoc. Their options are limited, and despite the aim of being low cost, the system will cost more than their existing methods. This places the product in the differentiation focus sector of Figure 2.

The system itself can be adapted for use in other non Eurobot-specific applications, such as in robotic vacuums that are aware of their environments. Smart robotic vacuums in this category cost in the region of £600 [3]. Additional features are required for a physical product to be realised, which will increase the cost to a similar range. The method of localising within its environment is unlike those used in existing vacuums in this market, leading to a product differentiation.

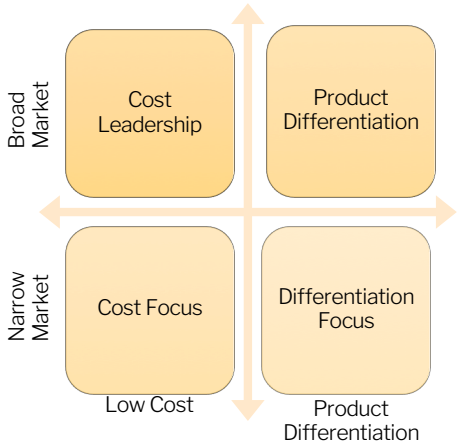
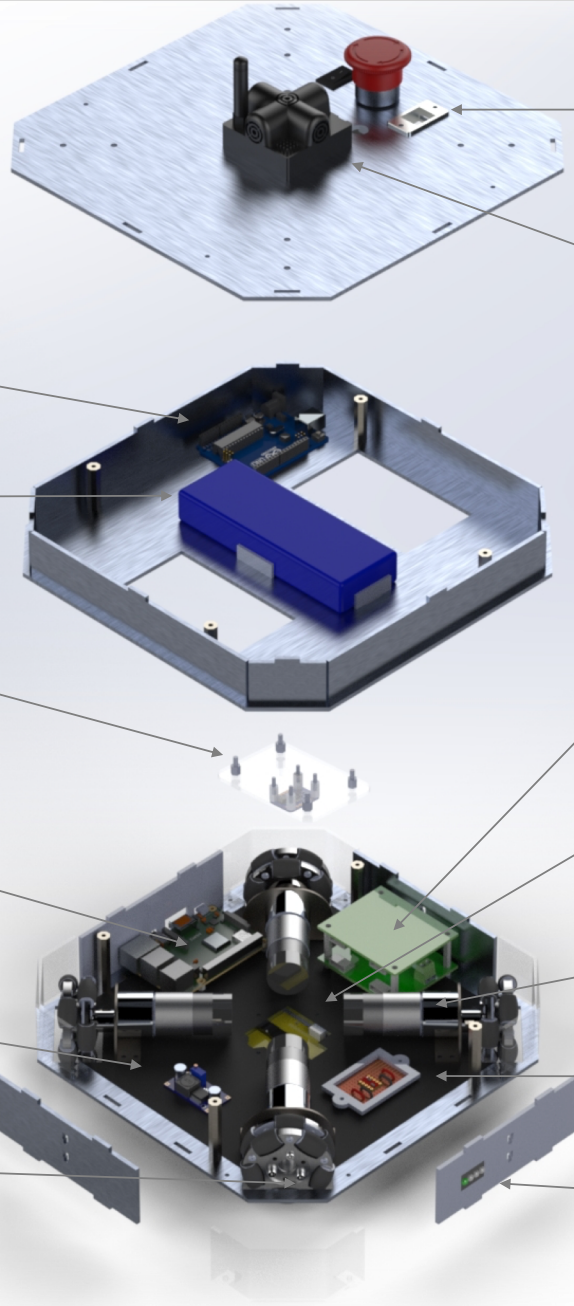


Figure 2: Market Strategies. Adapted from [4]

ROBOT OVERVIEW

DESIGN PROCESS

The following sections individually review the design process undergone for the selection and implementation of each aspect of the project, enabling the design of a system consisting of a unique combination of features.



ESSENTIAL WIRING (p. 16)
Including on/off rocker switch, emergency stop, USB port

BEACON (p. 7)
Absolute positioning sensor
Marvelmind mobile beacon (hedgehog)
Gives position of the robot, with respect to the environment

MICROCONTROLLER (p. 8)
Arduino Uno
Implemented for use with the absolute positioning sensor

BATTERY (p. 16)
3-Cell lithium polymer 5 Ah
A single battery powers the entire robot

IMU (p. 9)
Orientation sensor
Outputs the heading of the robot

MD25 MOTOR DRIVER (p. 11)
Incorporated motor speed control

OPTICAL FLOW SENSOR (p. 6)
Including lens, suspended mount
Relative positioning sensor
Gives position of robot with respect to its previous position

MICROCONTROLLER (p. 16)
Raspberry Pi
Multiple input/output ports
High processing power

DRIVE TRAIN (p. 11)
Including EMG30, encoder

DC DC BUCK CONVERTER (p. 16)
Steps down voltage to 5 V to supply the Raspberry Pi

I2C LEVEL SHIFTER (p. 16)
Allows communication between MD25 and Raspberry Pi

OMNI– WHEELS (p. 10)
Provide motion in any direction whilst keeping robot orientation constant

CALIBRATION LEDS (p. 9)
Indicate when IMU is calibrated

RELATIVE POSITIONING SENSOR

Relative (or local) positioning sensors measure the distance travelled with respect to a datum. They typically have fast update rates and high precision, ideal for tracking small movements. The aim for the relative positioning sensor in this project is to measure the distance travelled in each movement, with respect to the robot's previous location.

INTRODUCTION TO OPTICAL FLOW

Optical flow sensors are low resolution, high framerate cameras that can detect movement of surface features. Each pixel generates a scalar value of perceived motion, and the microchip processes average movement in the X and Y direction using: $\left(\frac{\sum x_i}{n}, \frac{\sum y_i}{n}\right)$. A scaling factor is applied to the sensor output so the pixel movement translates to a physical distance.

This technology is used in computer optical mice. An area of the work surface is illuminated with an LED, revealing a pattern of highlights and shadows. This pattern is reflected into the sensor, which takes thousands of pictures a second to measure motion.

Optical flow sensors are used in this project as a relative position sensor rather than motor encoders because they measure the relative distance the robot chassis has moved. This means the sensor readings are unaffected by wheel slip, unlike encoders, making the optical flow a superior relative sensor.

ADNS-3080 OPTICAL FLOW SENSOR

A range of optical sensor chips were compared based on their rated maximum speed, which accommodate for faster robot travel velocities, and their rated resolution, which provide readings to a higher precision. The ADNS-3080 by Broadcom (Figure 3) was further investigated in this project as it showed the best performance in both of these areas.

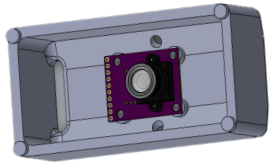


Figure 3: 3D printed mount to test the ADNS-3080 sensor

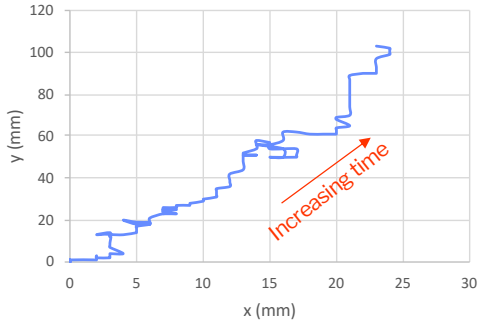


Figure 4: Results obtained from a stationary optical flow sensor over a 40 s period

However, this sensor is not used in the final design as it was highly susceptible to drift. A mount was 3D printed to hold the sensor a suitable distance away from the surface for testing. Two angled holes allowed for LEDs to illuminate the surface to be detected (Figure 3). Figure 4 shows the drift in readings of the ADNS-3080 optical flow sensor within a 40 second period where the sensor is stationary on a table. 100 mm of drift occurred in the y direction and 20 mm of drift occurred in the x direction during this time frame. This amount of drift was deemed unacceptable for application in this project, leading to the consideration of other sensors.

REPURPOSING OF A MOUSE SENSOR

The optical sensors used in computer mice are provided with a sensor and lens optimised to correctly focus light from the surface below. The final design uses the sensor from a Zelotes gaming mouse, which has resolution of 3200 DPI (dots per inch).

A similar test was carried out to assess the drift in the system, with much more satisfactory results. The drift in this sensor over a 40 second period was only 3 mm in the x direction and 2 mm in the y direction.

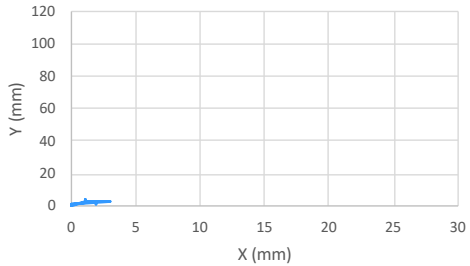


Figure 5: Drift in the mouse sensor over 40 s

While the mouse sensor has superior performance compared to the ADNS-3080, there is a drawback that it must be positioned only a few mm from the reference surface to function correctly. This can cause problems in the event of the robot needing to travel over small ridges on the Eurobot table. To overcome this problem, the PCB, LED, lens and sensor chip have been removed from the mouse casing and supported in a 3D printed mount for use in the final robot design (Figure 6, 7).

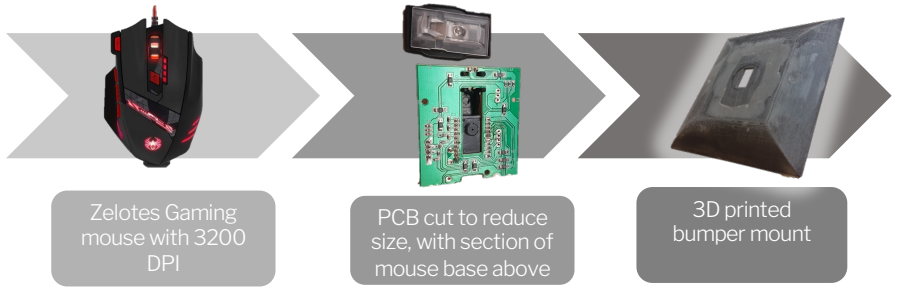


Figure 6: The steps involved in repurposing the mouse sensor

The mount is suspended from the base of the robot and has three features to keep the sensor stable and in contact with the floor (Figure 7).

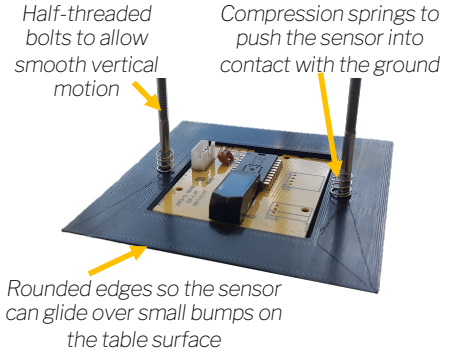


Figure 7: Mouse sensor suspended mount

Finally, the sensor was calibrated by performing manoeuvres over known distances, to find the relationship between the sensor's output values and actual distance in millimetres. The experiment yielded a straight-line fit with a precision of 1.15 mm, and R squared value of 0.9856, or a 99% 'goodness of fit'.

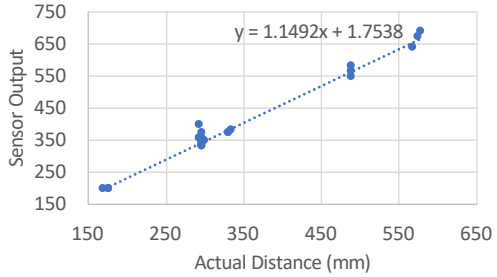


Figure 8: Example calibration data

ABSOLUTE POSITIONING SYSTEM

SENSOR SELECTION

Unlike odometric systems, absolute positioning systems do not rely on previous data and, as a result, do not compound positioning error. Implementing this type of system in the robot design allows for a reduction in positional uncertainty in later stages of a given Eurobot match when relative position sensors have accumulated drift. A beacon system using the lower side mounts on the Eurobot table is proposed in this section.

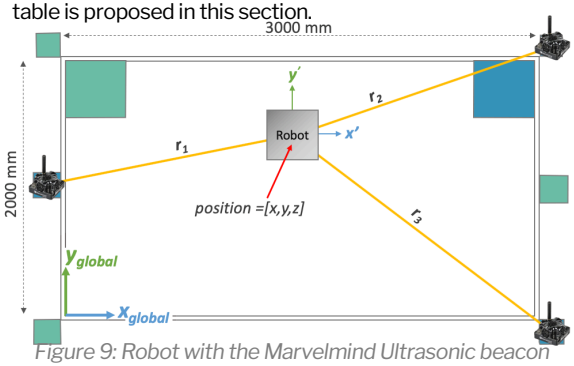


Figure 9: Robot with the Marvelmind Ultrasonic beacon system positioned on a basic Eurobot playing table. Basics of triangulation illustrated.

Eurobot regulations allow for three stationary "beacons" to be placed around the playing table to facilitate trilateration, using trigonometry and known distances to calculate position. Figure 9 illustrates how a trilateration system is setup using beacons in known locations.

Top performing Eurobot teams boast accuracies of ± 4 cm for their trilateration systems [5]. This was confirmed using equations 1-3 as being optimistic for commercially available 360⁰ spinning LIDAR sensors which achieve sampling gaps of 1.0 - 4.7 cm (Table 2).

$$Sampling\ Frequency = SF$$
$$Scanning\ Speed = SS$$
$$Max\ Distance\ from\ Robot\ to\ Stationary\ Beacon = r_{max}$$
$$Circumference = 2\pi r_{max}$$

$$T = \frac{1}{f} = \frac{1}{SS} = Time\ for\ 1\ Rotation\ (TR) \quad \text{Eq. (1)}$$
$$TR \times SF = Samples\ per\ Rotation\ (SR) \quad \text{Eq. (2)}$$
$$\frac{2\pi r_{max}}{SR} = Gap\ Between\ Samples\ (at\ 3\ m) \quad \text{Eq. (3)}$$

Table 2 is populated using the aforementioned calculations. It can be seen that ± 4 cm is unachievable for affordable LIDAR sensors such as RPLIDAR A1. It is worth noting that N301 LIDAR peak performance may operate at ± 4 cm uncertainty but is likely drop below this during normal operation with high CPU loads.

Name	Cost (£)	Sample Gaps at 3 m (cm)	Distance Uncertainty (cm)
RPLIDAR A1	£93.54	4.7	± 2.0
N301 LIDAR	£1840.00	1.0	± 3.0

Table 2: An example of two, commercially available, continuous rotating LIDAR sensors [6]. Distance uncertainties taken from respective technical specifications.

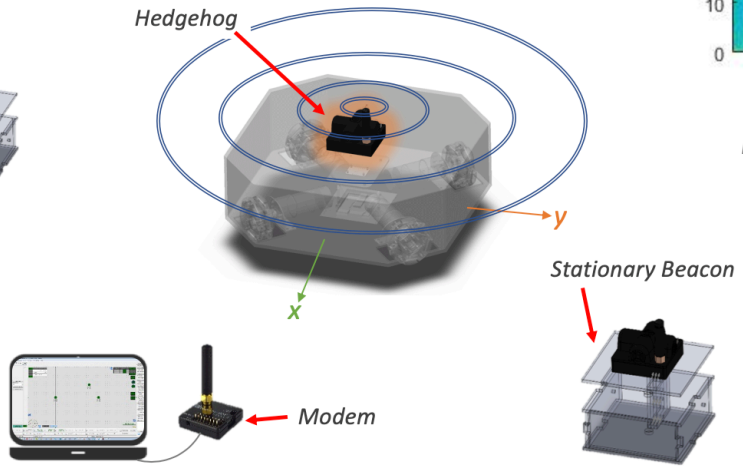


Figure 10: Illustration of the Marvelmind beacon system employed on our robotic system.

MARVELMIND ULTRASONIC BEACON

Due to the cost and uncertainty of LIDAR, this was not a viable option. Triangulation using infra-red and trilateration using ultrasound are susceptible to interference and require substantial filtering; the manufacture of such a system was deemed outside of the project scope. Commercially available Marvelmind ultrasonic beacon system, costing £390.00, promised ± 2 cm positional uncertainty and was therefore purchased for implementation in our system. The hedgehog emits 31 kHz ultrasonic pulses while three stationary beacons receive them and transfer the data via radio frequencies to the modem (Figure 10), calculating the position.

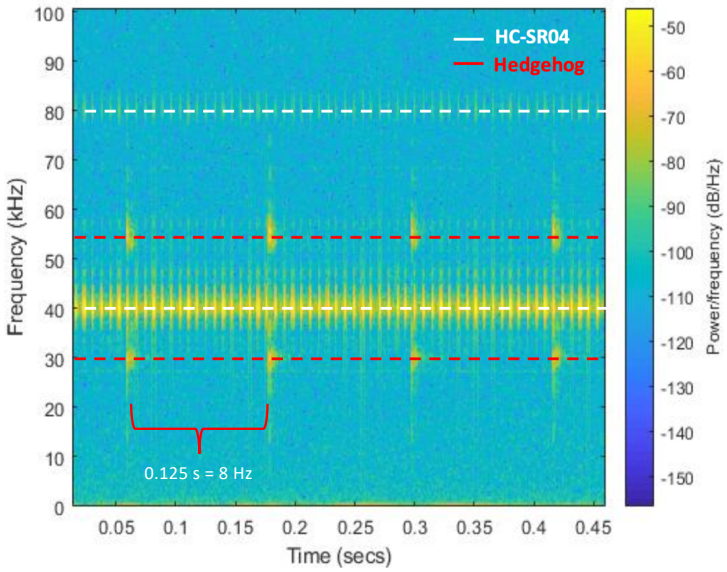


Figure 11: Hedgehog and HC-SR04 sensors pulsing at the same time. Both display harmonics frequencies. The beacon pulse frequency rate is maximum, at 8 Hz as indicated by the 0.125 s gaps.

BACKGROUND INTERFERENCE

An ultrasonic trilateration system was selected with accuracy being the most important criteria however, as previously mentioned, this can be affected by background noise.

Figure 12: HC-SR04.

Eurobot competitors commonly use HC-SR04 ultrasonic sensors for obstacle avoidance (Figure 12). This sensor uses a 42 kHz frequency.

The Marvelmind beacon system was set at 31 kHz to avoid interference from the HC-SR04 and other, similar sensors. Experimental analysis in the ISVR Anechoic Chamber quantifiably demonstrated that the Marvelmind beacons are not affected by the HC-SR04 sensors at these frequencies (Figure 11).

STATIONARY PERFORMANCE

To define parameters for sensor fusion algorithms and validate the ±2 cm uncertainty claim made by Marvelmind, stationary and moving tests have been conducted.

The hedgehog (see Figure 14) was placed in the middle of the test table with the stationary beacons positioned as illustrated in Figure 10. Over a period of 30 seconds, all the positional data points from the hedgehog were recorded. From this, standard deviations and ranges were calculated for *[x,y,z]*; (Table 3). These results showed that the beacon outputs had similar performance in both *x* and *y* directions.

Comparing the range of positional results collected, against the real beacon position, the maximum positional error of the Marvelmind system was determined to be 2.83 mm.

31 kHz Experiment	HC-SR04 Disabled			HC-SR04 Enabled		
	X	Y	Z	X	Y	Z
Standard Deviation (mm)	0.71	0.88	3.59	0.79	0.81	3.26
Range (mm)	3.0	4.0	13.0	3.0	4.0	14.0

Table 3: Standard deviation and range with and without background noise.

SAMPLE UPDATE RATE

The highest attainable sampling update rate for the Marvelmind beacons is 8 Hz. The update rate of sensor fusion algorithms are limited by the sampling frequency of the slowest sensor. For this reason, it is critical that the Raspberry Pi can resolve a result from the incoming data at 8 Hz.

During testing, it was discovered that the Raspberry Pi cannot adequately process all of the incoming data from the hedgehog. The Raspberry Pi CPU usage for the beacon alone was near 40% capacity and could only resolve an updated position at a rate of 0.8 Hz.

During normal robot operation, the Raspberry Pi CPU usage increases to 70% of its total allowable limit. As a result, new beacon positions resolve even more infrequently than the values listed in Table 3. To improve this rate, streamlining the python script, as well as compiling into C (via the Cython extension) was tested; these improvements did not increase the CPU performance enough. Introducing a separate Arduino processor dramatically improved performance and will be discussed in the following section.

	Rate at which the Pi received new position from Beacon (Hz)	Raspberry Pi CPU usage (%)	Normal Robot Operation Affected
Python Script	0.8	38- 45	Yes
Not scanning for IMU data	1.7	35-40	Yes
Compiled to C	4	35- 40	Yes
With Arduino Processor	>8 (beacon sampling frequency = 8 Hz)	4-6	No

Table 4: Allowable rate the Raspberry Pi could resolve a position from the beacon before returning repeated positional data. Identifies whether these rates were affected by normal robot operation.

ARDUINO INTERMEDIARY

An Arduino Uno was introduced as an intermediary (as a slave to the Raspberry Pi) between the beacon and the Raspberry Pi (see wiring diagram on page 16). This dramatically reduced the computational load on the Raspberry Pi from 40% to 6% capacity (Table 4) and allowed for the Raspberry Pi to receive an updated beacon reading at a 8 Hz, or higher if the beacon allowed for this.

Note: *The Arduino script was designed to pass was a string in the format '[x:PositionX y:PositionY z:PositionZ].' This would then be decoded by the Raspberry Pi.*

PREDICTIVE ALGORITHM

To improve sampling rates of the beacons a script was developed to extrapolate positions in the intervals between readings. By estimating the location of the beacon based on previous points, estimations can be used alongside real-time readings form the optical sensor and IMU. This facilitates more frequent updates from the sensor fusion algorithm.

For simplicity, the robot only moves in straight lines which simplified the path prediction dramatically. By using a least squares regression on samples collected during the movement, rates as high as 16 Hz can be achieved with minimal implications on the accuracy of the results, keeping within the ±2 cm uncertainty range indicated by Marvelmind specifications (Figure 13).

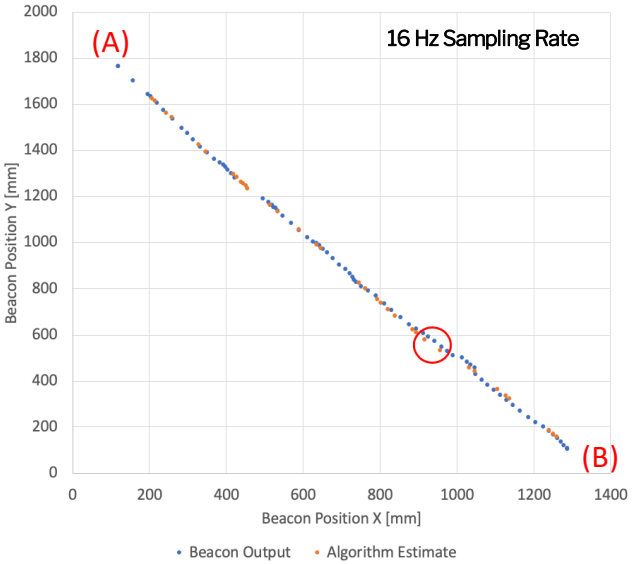


Figure 13: Beacons' position as it moves from (A) to (B); 16 Hz sampling rate (double beacon limit). Blue dots indicate a real updated position by the beacon, orange dots indicate the algorithm estimate. 45% of results were estimates which is expected as sampling rate is double the beacon limit. Red circle indicates outlier estimates however are still within ±2 cm uncertainty.

BEACON PERFORMANCE DROP

Despite significant reductions in CPU usage and increased sampling rates from a predictive algorithm, once integrated into the robot, the performance dropped dramatically. Not only did the sampling rate decrease, but the latency of the results increased beyond a usable limit. For this reason, the beacons were not integrated into the final system. Factors including CPU load on the Pi, radio frequency and electronic interference are discussed in further detail in the Results section.

Figure 14: Marvelmind beacon used for triangulation (left) and the modem (right)

ORIENTATION

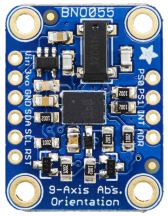


Figure 15: Adafruit BNO055

ORIENTATION METHOD

Orientation may be determined via odometrical means, with static reference points, or by gyroscopic measurements. Currently, the robots made by Part II students determine heading by measuring the difference in wheel rotations; this method is prone to errors such as wheel slip, motor encoder inaccuracies and motor step size. Incorporating a reliable orientation system with minimal vulnerability to aforementioned factors was desirable.

DESIGN CONSIDERATIONS

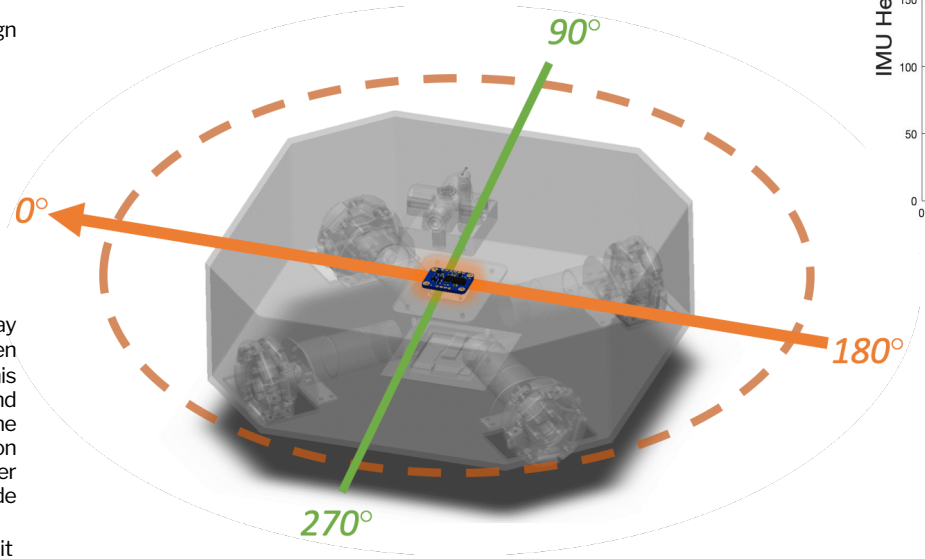
The following were deemed as being important design parameters for our selected orientation method:

- Sensor Size: As future Eurobot users would require a low profile platform to build upon, it was important for the orientation method to be small and non invasive.
- Sensor Placement: Ensuring the robot's centre of rotation aligns with the centre of the sensors used. By using a singular IMU allowed for a reduction of post processing calculations required.
- Sensor Dependency: As mentioned, orientation may be determined by measuring the difference between two odometric sensors. Creating a robot with this setup would result in both orientation and displacement measurements being derived from the same sensors. Inertial measurement units (IMUs) on the other hand employ gyroscope, accelerometer and magnetometer readings and exclusively provide orientation data.

With these considerations in mind, the Adafruit BNO055 inertial measurement unit (IMU) was selected for design implementation. This sensor is small enough to fit in the centre of the robot

ADAFRUIT BNO055

The BNO055 (Figure 15) employs a gyroscope, accelerometer, magnetometer and temperature sensor however, what makes it superior to its competitors, is that the sensor fusion is implemented on board at a 100 Hz update rate (Euler and Quaternion Orientations as outputs). This not only simplifies the programming for our applications, but also dramatically reduces the computational load on the Raspberry Pi's CPU. Furthermore, the original sensor outputs can be called by the Pi if required in future work.



CALIBRATION WITH LEDS

Upon robot start-up, the BNO055 sensor should be calibrated to ensure optimal sensor performance. As an aid to the user, LEDs have been installed in robot to indicate the state of calibration. One white LED indicates *Gyroscope Calibration*, two indicates *Accelerometer Calibration* and so on. Upon completion, all four LEDs will flash four times to indicate a successful start-up phase. Calibration settings will be saved on the Raspberry Pi until reset. However, due to time constraints in the project, the orientation was not included in the sensor fusion techniques in the final design.

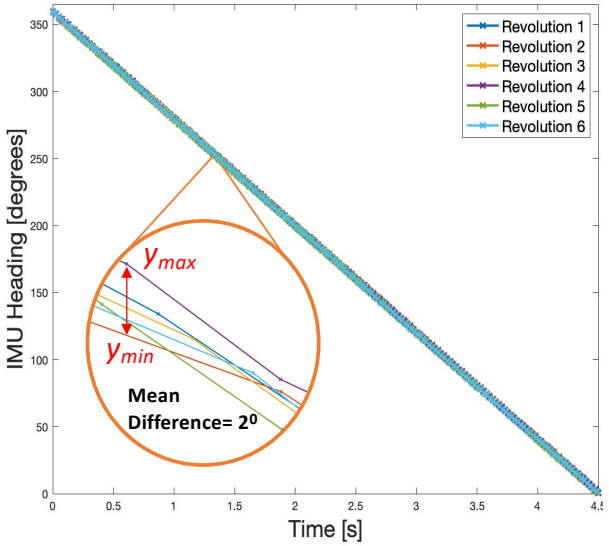


Figure 16: Adafruit BNO055 heading for six consecutive, full revolutions of the robot. The mean difference across all rotations is 2⁰. No observable drift.

BNO055 PERFORMANCE

Testing identified that sensor drift was insignificant (**Figure XX),** after six consecutive revolutions, the sensor readings were consistent. A mean difference of 2⁰ was deduced from the plot above using the following equation:

$$Mean\ Difference = \frac{\sum_{i=0}^{n} y_{max}(t_i) - y_{min}(t_i)}{n+1}$$

Where $n = number\ of\ samples$. 2⁰ equates to 2.28 mm variance between readings (for a robot footprint of 265 x 265 mm). Furthermore, average angle measurement error was calculated as being as 0.05⁰. This was determined by aligning the IMUs four flat faces (one at a time) with the flat face of a wall and comparing the IMU output against 0⁰, 90⁰ and 270⁰.

DRIVE TRAIN

WHEELS

There are three main classes of wheel: standard wheels and holonomic wheels, which are split into omnidirectional and spherical wheels. Each vary in their kinematics so have a large effect on the overall motion of the robot and therefore the path planning.

The standard wheel is used by current Southampton Eurobot teams, two wheels are used in a differential drive configuration with a ball caster to provide support. Robots with wheels such as this can move in two main ways: forward and reversing linearly and turning about the centre of its wheel base [7]. Arcs can be achieved by writing forward velocities of different magnitudes to each wheel.

Control of the standard wheel is the simplest of all robot mobility mechanisms but the low number of degrees of freedom of adds complexity to the path planning process.

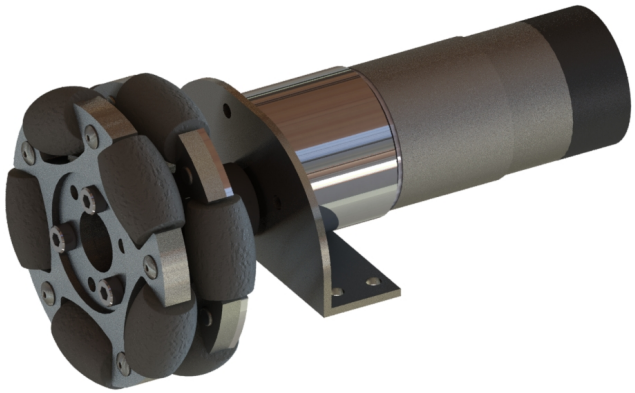


Figure 17: Motor, mount and omni-wheel CAD model used in the final design.

There are two classes of holonomic wheel:

i. Omnidirectional wheel – Omnidirectional wheels have three degrees of freedom. Their shape is similar to that of a standard wheel and is also controlled by a single actively powered shaft through its centre. There are an array of passive rollers on the periphery (Figure 17) that allow movement in multiple directions with little resistance, which is the main advantage of this type of wheel [8]. The main disadvantage is that their configuration typically consists of three or four wheels and the slip dynamics of the wheels is difficult to model. The required slip between the passive rollers and the surface to provide omnidirectional motion reduces the maximum amount of traction that can be provided by these wheels.

ii. Spherical wheel – The spherical wheel has the most degrees of freedom and can actively spin in any direction. However, it is difficult to confine the movements of a spherical drive due to the many directions of motion they allow [7].

OMNIDIRECTIONAL WHEELS

Of the three classes of wheel, the omnidirectional drive proves to be the most suitable for the robotic platform in this project. The primary reasons for selecting this class of wheel is due to the versatility it provides for testing a variety of sensors and localisation methods. For example, localisation systems that depend on distances to the surrounding walls are more easily implemented on a system that can move in multiple directions without changing overall rotation. An omnidirectional robot could facilitate this much more simply than one that has standard wheels. Two types of omnidirectional wheels were considered: Mecanum wheels, and standard 'omni-wheels'. The key differences between both are compared in Table 5 below.

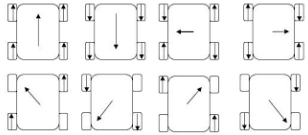
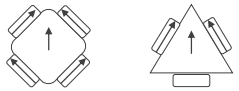
	ROLLER LAYOUT	FEASIBLE WHEEL CONFIGURATIONS	ADVANTAGES	DISADVANTAGES
Mecanum wheel	 [9]	 Car-like wheel positions.	More suited to higher payloads – wheels have more traction.	Higher cost. For a given wheel angular velocity, the forward velocity is 30 % slower than an equivalent omni drive.
Omni-wheel	 [10]	Circular wheel layout  4-wheel 3-wheel	Lower cost. True omnidirectional motion.	Circular position of wheels makes robot layout more awkward. Lower transmitted torque compared to Mecanum wheels.

Table 5: Comparison of wheel types.

Omni-wheels were selected to be used in the final design. This is due to the large cost difference between the two types of wheel. Mecanum wheels are larger than omni-wheels. This limited the range and quality of Mecanum wheels available to purchase that are suitable for a robot of size that fits within the *Eurobot* regulations.

LAYOUT CONFIGURATION

Platforms with omni-wheels can be configured for both three wheels or four wheels. The three wheeled system spaced the wheels 120° apart, and the four wheeled configuration requires 90° spacing.

Three wheeled omni robots are superior when tasked with traversing uneven terrain. This is because three points of contact are required to define a plane. Therefore, all wheels in a three wheeled system will touch the ground. A platform with four wheels is not guaranteed to have all four wheels touching the ground at once. However, four wheeled platforms were chosen for the final design as they have more motors driving the system. Therefore, the overall pushing force is greater. The free body diagrams below compare the forces in the possible driving directions for both three wheeled and four wheeled robots.

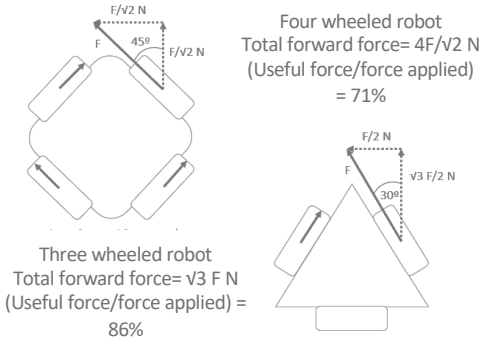


Figure 18: Wheel Layout Options.

Additionally, the four-wheel omnidirectional set up allows for more directions of travel (eight versus six). Four of eight directions of travel align with the cartesian coordinate system. This makes the transformation calculations to the global coordinate system simpler.

MOTOR SELECTION

To create a system that can successfully compete on multiple future competition boards, the robot's drivetrain must provide sufficient torque to allow the robot to climb gradients in the form of slopes, ramps and see-saws. Limits were set for the overall mass of the robot and maximum incline to be traversed. The maximum incline is set from reviewing previous Eurobot competition boards and selecting the largest traversable angle. Further assumptions are listed in the table below. The calculated value of torque acts as a guidance figure to use as a reference when searching for appropriate motors to use in the system.

Maximum mass, $\mathbf{m}$	6 kg	Wheel radius, $\mathbf{R}$	0.030 m
Maximum acceleration, $\mathbf{a}$	0.10 ms^{-2}	Minimum number of driving motors, $\mathbf{N}$	4
Maximum slope angle, $\boldsymbol{\theta}$	12°	Minimum assumed efficiency, $\boldsymbol{\eta}$	70 %

Table 6: Comparison of motor options.

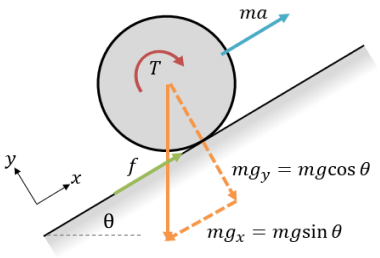


Figure 19: Free body diagram of a wheel accelerating up a slope with no slip condition.

The dynamics of an omni-wheel is impossible to perfectly capture by simple calculation. To attain a guidance torque, the omni-wheels are assumed to be standard wheels of mass, m, and radius, R, accelerating up an incline of angle, θ. An efficiency scaling factor is used to account for the vector cancellation (calculated previously in the wheel section). This gives the expression of the required torque to be:

$$T = \frac{R}{\eta N}(ma + mgsin\theta)$$

On substitution of the values this yields a target torque of:

$$T = 0.138\ Nm$$

Four possible motors that provide a torque exceeding the stated value are described in the table below.

Motor	Devantech EMG49	Devantech EMG30	NEMA 17	Maxon EC45
Type	Brushed DC	Brushed DC	Stepper	Brushed DC
Rated Torque	0.369 Nm	0.147 Nm	1.3 Nm	1.46 Nm
Rated Voltage	24 V	12 V	24 V	24 V
Cost per motor	£71.49	£28.45	£10.00	£103.34
Integrated Encoders	Yes	Yes	No	No
Designed Motor Driver	Yes	Yes	No	No

Table 7: Comparison of motor options.

After considering how the motors compared, the Devantech EMG30 12 V gear motor (EMG30) and MD25 driver set was found to be most appropriate, primarily due to their wide availability and low cost. The EMG30 motors are used by Part II students in both the odometry exercise and Eurobot projects, therefore they are readily available at no cost to this project. They have low power requirements, so a smaller battery can be used in the system. An integrated encoder means that odometry readings can be taken directly. The motor controller (MD25) has incorporated speed control and is also readily available. However, the EMG30 do provide the lowest torque out of all the motors compared, and their maximum rated torque is close to the calculated value. While the motors are suitable for Part II Eurobot projects, the performance of the motors may be different for an omnidirectional drive system that has more on board components and is likely to have higher mass than a typical second year project. To ensure the motors are appropriate for the platform, an experiment has been devised to determine the operational limits of the motors. A prototype chassis was driven up a slope of 12°, each time with increasing mass. The current is measured to ascertain whether the limit of the motors had been achieved. The experiment was completed for both a four wheel drive configuration and a two wheel drive configuration.

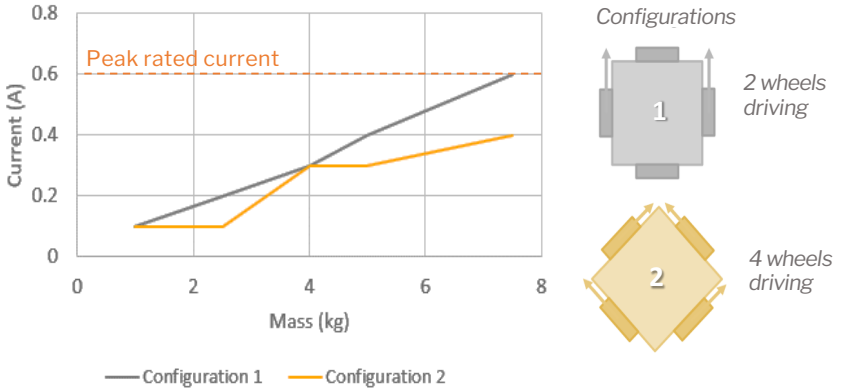


Figure 20: Graph showing the relationship between current and mass as a robot traverses the incline for both the 2 wheels driving and four wheels driving configurations

The motors in the two wheels driving configuration reached peak current when the robot had a mass of 7.5 kg (Figure 20). The four wheel configuration did not reach the peak current in the range of masses tested. As 7.5 kg is more than the specified maximum robot mass, the results show that the EMG30 motors were suitable for the robot platform in this project.

MOUNTING BRACKET

The mounts provided with the EMG 30 motors (Figure 21, left) proved to be too large for the selected wheels.

New mounts were designed to position the motor shaft closer to the base, allowing for a larger range of wheel sizes. The overall height of the mount was also reduced to allow for a shorter chassis. The mounts were manufactured from 1 mm sheet steel.

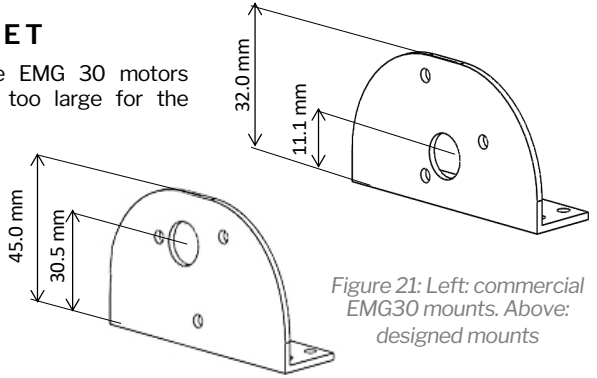


Figure 21: Left: commercial EMG30 mounts. Above: designed mounts

POSITION CONTROL

OMNIWHEEL KINEMATICS

The omniwheel configuration has different kinematics to that of a standard wheel. To obtain equations describing the kinematics, some assumptions must be made:

1. The final destination point of the robot and the velocity of the robot is known.
2. The motion of the robot is described by the motion of the robot's centre point, O, which is the point of intersection of all motor angles.
3. The orientation of the robot remains the same throughout the motion. Therefore wheel velocities V1 = V3 and V2 = V4. To complete a translational motion, the robot first must rotate about its centre to align its local coordinate system (Xr, Yr) with the global coordinate system (Xg, Yg).

Figure 22 shows how the velocities of the robot translate to a vector diagram, where ϕ is the angle between the Xr axis and the wheel axle, in this case 45°.

ϑ is the angle between the robot's coordinate system and the global coordinate system. V is the resultant velocity vector of the robot.

It can be deduced from Figure 22 that the velocity vector, V, can be split into components in the global coordinate system, Vx and Vy.

$$Vx = Vcos(\vartheta), \qquad and \qquad Vy = Vsin(\vartheta)$$

Vx and Vy can be given in terms of V1 and V2 with the following:

$$Vx = V_1cos(\phi) - V_2cos(\phi)$$

$$Vy = V_1sin(\phi) + V_2sin(\phi)$$

Which can be rearranged to give:

$$V_1 = \frac{1}{2}\left(\frac{Vx}{cos(\phi)} + \frac{Vy}{sin(\phi)}\right), \qquad V_2 = \frac{1}{2}\left(\frac{-Vx}{cos(\phi)} + \frac{Vy}{sin(\phi)}\right)$$

And as V1 = V3 and V2 = V4, the final two remaining velocities can be deduced.

$$V_1 = V_3 = \frac{1}{2}\left(\frac{Vx}{cos(\phi)} + \frac{Vy}{sin(\phi)}\right), \qquad V_2 = V_4 = \frac{1}{2}\left(\frac{-Vx}{cos(\phi)} + \frac{Vy}{sin(\phi)}\right)$$

Therefore from *a priori* knowledge of V and ϑ, the required velocity from each wheel can be calculated.

A straight line path is assumed giving a constant value of ϑ throughout the movement. Assuming the current position of the robot takes value (x1, y1) and the destination point takes value (x2, y1), ϑ can be calculated using the straight line connecting P1 and P2 (Figure 22).

$$\vartheta = tan^{-1}\left(\frac{y2 - y1}{x2 - x1}\right)$$

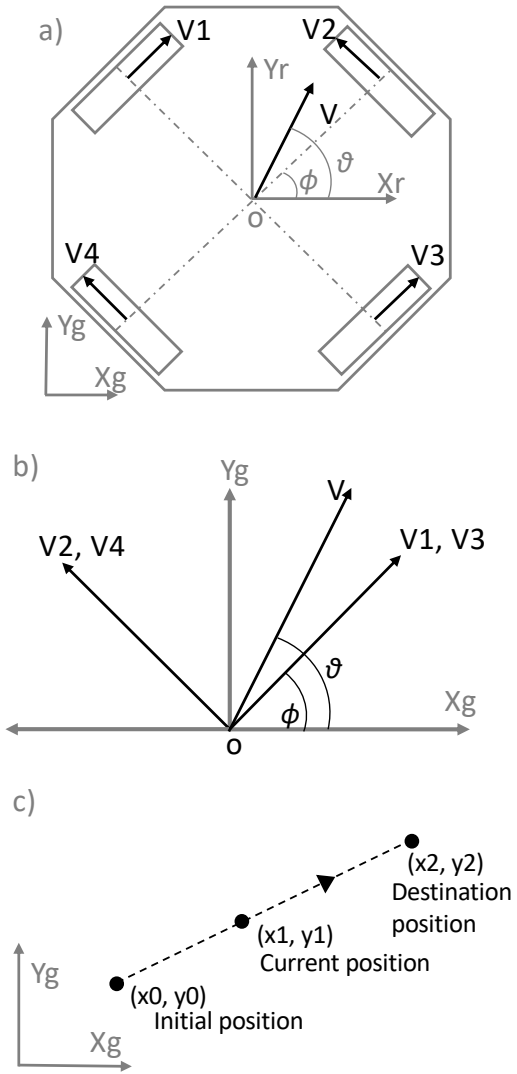


Figure 22: a) Layout of four wheeled omnidirectional drivetrain. b) Velocity vectors of each robot with respect to the global frame of reference. c) Linear path of the robot.

PID CONTROLLER

A proportional-integral-derivative (PID) controller is a widely used mechanism for applying correction to a control function by measuring the difference between a desired set point and a measured process variable [11]. It consists of three parts:

1. A proportional term which gives an output proportional to the current error.
2. An integral component which eliminates the steady state error by integrating the error over a period of time until the error reaches zero. It affects the stability and speed of the response.
3. A derivative component uses the change of error with respect to time to anticipate future error. It compensates phase lag caused by the integral part.

PID FOR ROBOT POSITIONING

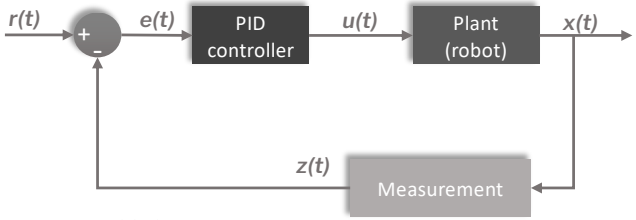


Figure 23: Control loop feedback block diagram of the system with applied PID controller

As the PID controller is to be used to control the position of the robot (Figure 23). In this case, the set point, r(t), is the destination point (x2, y2). The error term, e(t), is calculated by taking the difference between the current position of the robot, y(t) (x1, y1). This value is taken from the optical sensor, which measures the motion of the robot in the Xg and Yg directions. u(t) is the control input signal, which takes the form of wheel velocities (V1, V2, V3 and V4). The addition of a Kalman filter (Figure 24) to increase the reliability of the readings of the current position is investigated in the latter parts of this report.

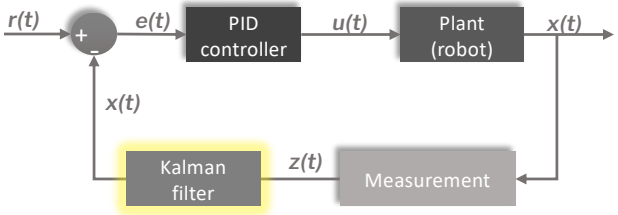


Figure 24: Control loop feedback block diagram of the system with applied PID controller and Kalman estimator

CHASSIS DESIGN

PROTOTYPE CHASSIS

Having a working chassis was crucial to properly test the design of the internal systems. Initial prototypes were laser cut from plywood and used to develop the movement code using the four omni-wheel.

Figure 25: Prototype robot.

PARAMETERS

To fulfil the design brief, the chassis must comply with the Eurobot regulations. The chassis has been designed within these parameters (Table 8).

	Maximum permissible (mm)	Robot (mm)
Perimeter (mm)	1200	977.2
Height (mm)	340	99

Table 8: Applicable Eurobot regulations vs. design dimensions [1].

Alongside these regulations, stakeholder Robosoc required these parameters to be minimised as not to limit future additions, leading to a compact design.

MATERIAL

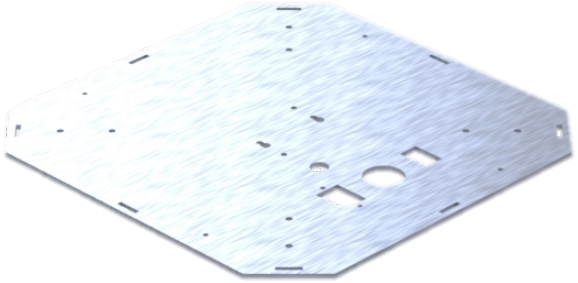
3 mm aluminium was selected due to the high strength to weight ratio and is able to withstand years of use. Plywood would need to be thicker to withstand the same forces, and is prone to warp. Acrylic was considered, but is prone to brittle fracture, so was not used for load bearing components. Although structurally sound, testing found the aluminium interfered with the radio wave communication in the absolute positioning system. The aluminium absorbs radio waves, reducing the performance. Tests were done using an acrylic top plate to reduce this phenomena.

Standoffs were used with all PCBs, an insulating layer separated the base plate from the wiring and ensuring all removeable wires were female-female mitigated the risk of shorts occurring both during building and in future use.

BUILD

The aluminium was cut with a waterjet to achieve tight tolerances. The chassis is held together with metal standoffs between the layers, allowing for a sturdy chassis that can be quickly disassembled.

TOP LAYER

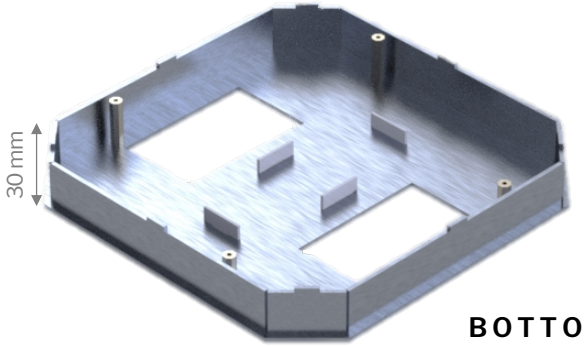


Contains: Ultrasonic beacon, rocker switch, emergency stop, USB port

Holes for QR code mounting.

Easy access to battery: removed by undoing four M3 bolts. The bolts can be replaced with standoffs providing easy addition of further layers, as per Robosoc's requirements along with the USB port, removing the need for disassembly when using with an additional Arduino .

MIDDLE LAYER

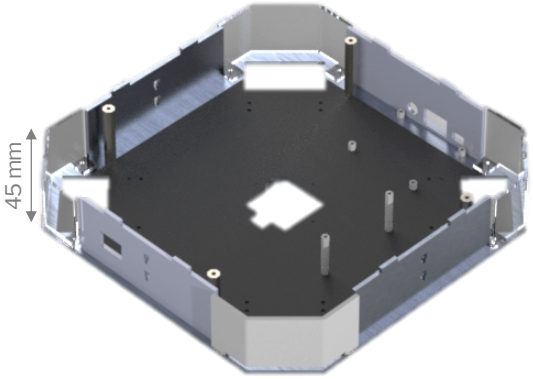


Contains: Battery

Separates battery from the other electronics, allowing for more space to add and remove sensors from the bottom layer.

Although overheating is unlikely, large holes increase ventilation. They also provide easy access to the MD25s and buck converter.

BOTTOM LAYER



Contains: Raspberry Pi, MD25s, I2C level shifter, buck converter, drive train, optical sensor, IMU

Optical sensor suspended below plate, the mounted IMU directly above; both centrally positioned.

The MD25s are mounted on top of each other, optimising space.

The Raspberry Pi micro USB and HDMI ports are accessible through the side panel.

Omni-wheels are enclosed for protection, using non-structural acrylic 'windows' designed to showcase the impressive movement of the wheels, aimed to attract attention to the workings of the robot, ideal for showing on open days.

Figure 26: CAD model chassis.

SENSOR FUSION

A Kalman filter is implemented in this project to robustly combine different types of sensor to gain a better estimate of the robot's position. There are two types of position sensor employed on the robot: relative positioning sensors and absolute positioning sensors, each have their own associated drawbacks. The sensor fusion algorithm is being employed to overcome the shortcomings of both groups of sensors by combining them. Different configurations of the filter can be used to assess the effectiveness and impact of each sensor on the system. The algorithm can be used to combine an optical sensor, providing local sensing, and an ultrasonic beacon system, providing global sensing, to give overall positioning.

KALMAN FILTER OVERVIEW

The Kalman filter is common in robotics literature. It is a recursive estimation model, where knowledge of error in the state, which initially starts out as a covariance matrix, is improved with each time step, as the filter combines measurements with a predefined state model [13].

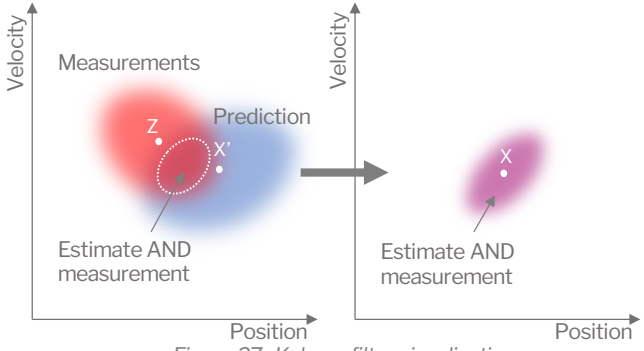


Figure 27: Kalman filter visualisation

Figure 27 shows a visualisation of a typical Kalman filter. Both the measurement and prediction models are represented by Gaussian distributions, each has a mean and covariance, shown by the red and blue regions. The intensity of the colour represents the likeliness of the value being at that point
These regions are multiplied to leave a final optimal estimate, denoted X, which has its own associated mean and covariance

KALMAN FILTER PROCESS MAP

The below process map details the stages in calculating the estimate. The colours of the boxes match the corresponding gaussian regions in the graphs above.

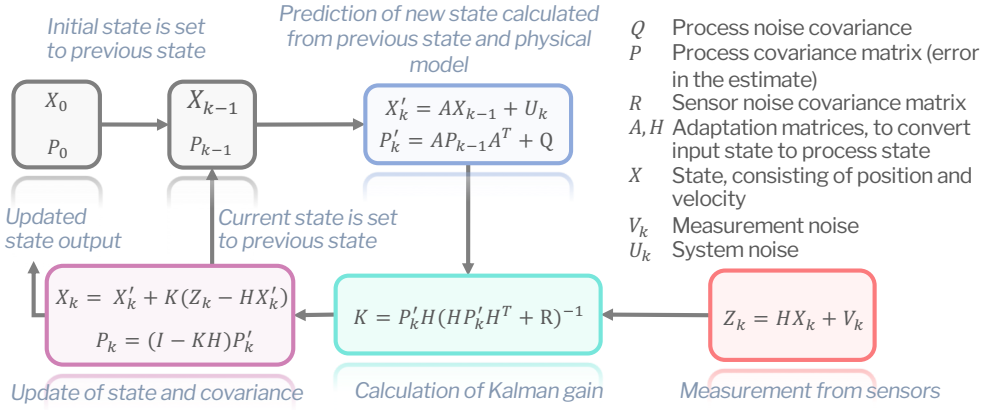


Figure 28: Steps and associated calculations per iteration of the Kalman filter, adapted from [12]

KALMAN FILTER STATE MODEL

The filter is designed in this project to estimate the global (x, y) position of the system with respect to the origin (set as the lowest left corner of the test board). The state space model of the moving robot can be modelled using the equations to the right.

$$X_k = AX_{k-1} + U_k,$$

The filter uses this model to predict the movement of the robot. The model has an associated error appended to it when used in the filter (as shown in Figure 28) and is combined with the measurement model, described below.

$$A = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad X_k \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$$

KALMAN FILTER MEASUREMENT MODEL

The measurement model uses the x and y positions read from the optical sensor. The optical sensor is reset at the beginning of every movement to reduce the effects of drift.

$$where \quad Z_k = HX_k + V_k$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

KALMAN FILTER IMPLEMENTATION

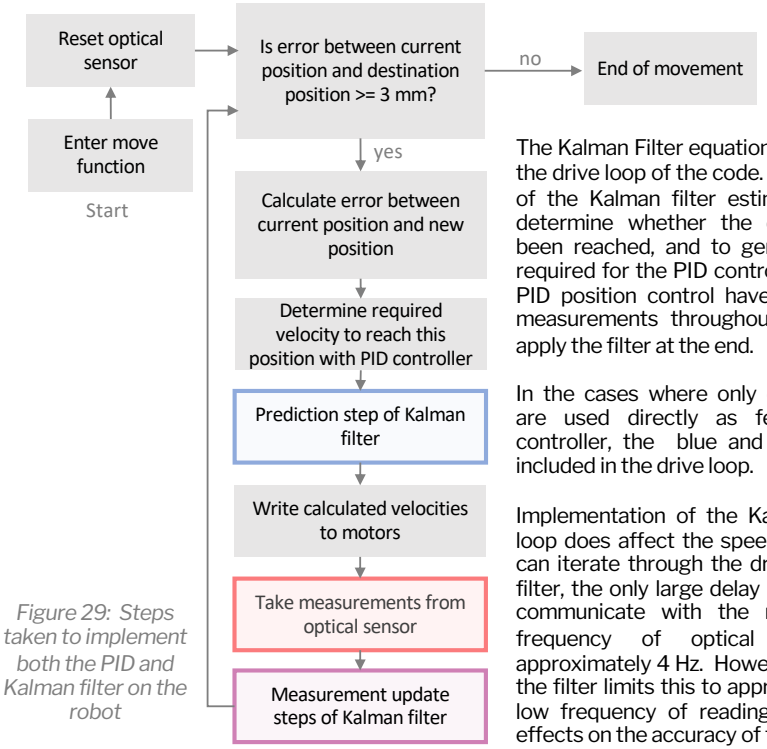


Figure 29: Steps taken to implement both the PID and Kalman filter on the robot

The Kalman Filter equations are calculated within the drive loop of the code. This allows the output of the Kalman filter estimation to be used to determine whether the destination point has been reached, and to generate the error term required for the PID controller. Systems without PID position control have the ability to collect measurements throughout the movement and apply the filter at the end.

In the cases where only optical measurements are used directly as feedback to the PID controller, the blue and purple steps are not included in the drive loop.

Implementation of the Kalman filter within the loop does affect the speed at which the system can iterate through the drive loop. Without the filter, the only large delay is the time required to communicate with the motors, meaning the frequency of optical readings can be approximately 4 Hz. However, implementation of the filter limits this to approximately 1.6 Hz. This low frequency of readings may have negative effects on the accuracy of the filter.

GRAPHICAL USER INTERFACE

MOTIVATION

To create an interactive and accessible system for future Southampton Eurobot teams, or for projects requiring validation for a localisation system, a simple user interface is desirable. The aim was to generate a graphical environment that can receive an input from filtered positional data from the robot, and also generate a command output to send to the robot. The interface should represent the competition board, and manoeuvre a virtual robot character in the same position as the real input being received. Lastly, the output should command the robot to a position. A visualisation tool is also in-line with the project aims of making the output well-suited for demonstrations such as open days and the *Engineering Design Show 2019*.

TOOL COMPARISON

Three technologies to create the visualisation were considered:

Technology	Description	Advantages	Disadvantages
OpenGL	OpenGL: C++ based application programming interface (API) for rendering vector graphics.	• GPU hardware acceleration: much faster graphical processing. • Design flexibility.	• Low-level and C++ : very large learning curve. • Difficult to interface with python source files.
Godot	Godot: 2D and 3D pre-built game engine and game development environment.	• User-friendly. • Quick and easy to start designing, with a drag and drop interface.	• Event scripting written in C# : language none of us have any experience in. • Less flexible. • Difficult to interact with python.
pygame	pygame: python library for making multimedia applications.	• Written in python. • Simple syntax.	• No GPU hardware acceleration: slower • No 3D capabilities.

Table 9: Comparison of possible GUI technologies

INITIAL DESIGNS

OpenGL and Godot were experimented with, where coordinate systems and grids to represent a floor plane were developed (Figure 30). Neither were completely suitable due to their difficulty in interacting with external python source files.

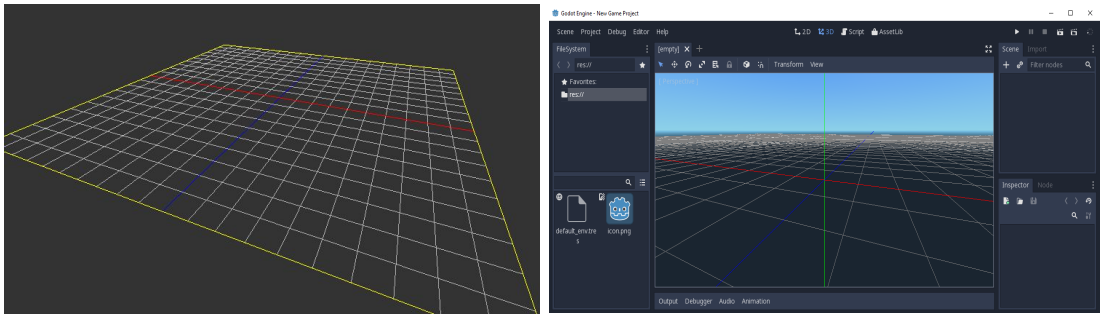


Figure 30: Left: OpenGL. Right: Godot development environment with coordinate system.

FINAL DESIGN

pygame has been ultimately chosen for its simple syntax, flexibility and most importantly, ease of communication with other python source files. The window generated is 600 x 900 pixels, which is in the same 2:3 ratio as the actual competition board. The background image is the same as the *Eurobot 2019* board design. The robot character is a function which receives a position vector and an angle for orientation, and projects a .png image file of the robot into the position provided.

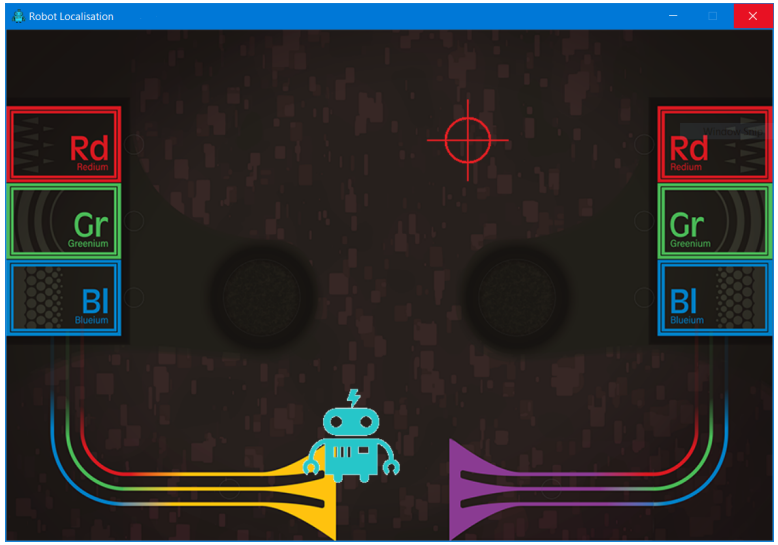


Figure 31: pygame Graphical User Interface designed to represent the robot's route

The code also includes a 'target' function, which allows the user to click a point on the virtual table. Doing so generates a cross-hair image on-screen at that location (for improved visual responsiveness for the user), and outputs the position vector at which the click occurred, relative to the pixels on the screen. This can then be scaled to represent a real position on the actual table. Figure 31 is a screenshot of the interface after a target has been clicked.

COMMUNICATION

The graphical user interface runs on the user's laptop computer. SSH (Secure Shell) Communication is possible between the laptop and the Raspberry Pi if they are on the same Wi-Fi network. The Raspberry Pi sends the position data, after it has read and processed its sensors, for the laptop to display the virtual robot character moving in time with the real one. The 'target' function output data is received by the real robot's path planning system instructing it to move to the chosen position.