UNIVERSITY OF Southampton

Design Journal Summary
FEEG6013 Group Design Project

51
Robot Localisation

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

ID Number: 26121093
Name: Oliver Arent Heilmann

Primary Supervisor: Prof. Paul White

Co-Supervisors: Prof. Martyn Hill

Submitted on: 09/05/2019

SUMMARY OF INDIVIDUAL CONTRIBUTION

The idea behind building a localisation system usable for the international Eurobot robotics competition was conceived over summer, during Alice's and my time at the Eurobot 2017/18 world final in France. Speaking with peers, the project idea gained traction and eventually morphed into the final project specification.

Having already taken part in the Eurobot competition, as well as becoming a teaching assistant to that module the following year, it was initially my job to get other team members up to speed with the project scope. As other team members had less experience with this competition, I took lead of earlier meetings and highlighted key design considerations that eventually became large sections of the GDP.

Over the course of this project, my key roles included:

- **Motor, Drives & Sensor Research**: Having already spent significant amounts of time researching sensor types (Infra-red, ultrasonic, LIDAR etc.), it made sense for me to carry this on. After conducting a critical analysis into wheel selection, I arranged to borrow two sets of MD25s and driver sets from the Part II Systems Design and Computing module lead (saving us £250.00 worth of equipment). Motors and drive eventually becoming Caroline's role.
- **Funding Acquisition & Deliverables:** Being assigned the external liaison role, I sourced external funding for our project. By February 2019, we had secured funding up to **£2,400.00** from **Boeing**. Over the course of the project, I also submitted and delivered presentations, Milestone forms and managed all aspects of the purchase orders.
- **Sensor Selection and Implementation:** My technical role involved programming and testing for the all the sensors used in our robotic system.
- **Second Year Student Participation:** I was also the external liaison between my GDP group and the second year Eurobot students who volunteered to test their robot on our testing table. This involved coordinating between my team and all participants, as well as organising timeslots.

Albeit not lead roles, I worked with team members on the overall organisation, functionality and structure of the programming for the robot. Furthermore, I designed and manufactured robot parts surrounding the sensors.

ROLES, ACTIVITIES AND OUTPUTS

As this GDP was self-proposed, we were not provided with any source material, previous project iterations or design reports cataloguing previous experiences. As a result, every part of our project started at square one, with every aspect needing to be extensively researched before starting design and build phases. This project also lent itself well for each group member to have a lead technical role. Over the duration of this project, my technical role involved sensor selection, design, manufacture and implementation. It was critically important to characterise the performance of each sensor in order to develop a set of data that could be used to form the covariance matrix in the Kalman Filter; the success of this GDP was therefore directly related to my activities and project outputs.

GLOBAL POSITIONING SYSTEM

A huge issue surrounding odometry (the localisation method used by second year Eurobot students) is compounding error; the positional accuracy decreases with time from sequential movements. In an effort to mitigate (or significantly reduce) this, it was decided to build a global positioning system (GPS).

Research Phase:

During my time at the international Eurobot competition in France, I spoke to a number of top teams about the positional uncertainties of their GPS. Despite some teams spending >£5000.00 on their systems, without fail, uncertainties of around 4-7 cm were quoted. Several published research papers also catalogued design and build

procedures of Eurobot beacon systems with similar performances [1], [2]. Regardless of method, the result always seemed to be the same. My calculations demonstrated that there was a clear threshold for LIDAR sensors for example:

$$T = \frac{1}{f} = \frac{1}{SS} = Time\ for\ 1\ Rotation\ (TR) \qquad Eq.\ (1)$$

$$TR \times SF = Samples\ per\ Rotation\ (SR) \qquad Eq.\ (2)$$

$$\frac{2\pi r_{max}}{SR} = Gap\ Between\ Samples\ (at\ 3\ m) \qquad Eq.\ (3)$$

Where *SF= Sampling Frequency*, *SS= Scanning speed* and r_{max}*= Maximum distance from robot to stationary beacon).* **Table 1** was populated using these calculations. It can be seen that $\pm$ 4 cm is unachievable for affordable LIDAR sensors such as RPLIDAR A1. It is also worth noting that, while N301 LIDAR sample spacings are only 1 cm, the uncertainty measurement for distance is 3 cm. The combination of these two uncertainties also rendered the sensor too inaccurate for our application (and costs far too much).

Name	Cost (£)	Sample Spacing at 3 m (cm)	Distance Uncertainty (cm)
RPLIDAR A1	£93.54	4.7	$\pm$ 2.0
N301 LIDAR	£1840.00	1.0	$\pm$ 3.0

Table 1: An example of two, commercially available, continuous rotating LIDAR sensors. Distance uncertainties taken from respective technical specifications [4].

Infra-red triangulation system alternatives have also been developed with measurement angles 0.6404^0 (translating to 3.4 cm measurement uncertainty at 3 m from beacon) [3]. For both LIDAR and Infra-red methods, the aforementioned uncertainty readings are for only one beacon; in order to calculate the position of the robot, three measurements must be taken. For a robot using a scanning (spinning) receiver to measure its' distance to the surrounding beacon towers, time differences between measurements must also be taken into consideration during calculations. As one can see, building a GPS system with high accuracy (within the mm range) is nontrivial. Ultrasonic trilateration systems on the other hand do not have the same sources of error, however, manufacturing is very complex and requires significant background noise cancellation, filtering and demodulation to work effectively; a published paper stated their ultrasonic beacon system had errors of 1 – 10 cm for example [1]. After spending considerable time browsing the web for small scale, compact and commercially available triangulation/ trilateration systems, I eventually came across an ultrasonic beacon system. The product dimensions and design were within the regulations of Eurobot and the system promised to deliver a positional uncertainty as small as 2 cm.

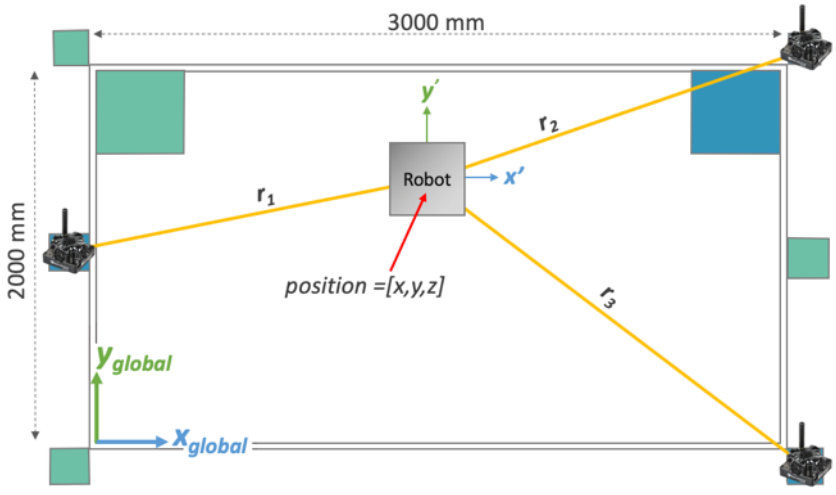

Figure 1: Robot with the Marvelmind Ultrasonic beacon system positioned on a basic Eurobot playing table. Basics of triangulation illustrated.

Aware that creating a high performance beacon system is complex, I conducted extensive testing to ensure that the company's (Marvelmind) claim of 2 cm accuracy was indeed correct. After reading the 80 page manual in detail and looking at several reviews and online troubleshooting threads, as well as coming into extra funding from Boeing, the group decided to purchase this system (Entry 1,2).

Initial Testing:

Figure 1 shows the basic principles of the Marvelmind trilateration system; time of flight measurements would be taken from stationary beacons to calculate distance from hedgehog. Preliminary testing began before the completion of the testing table. For this reason, I created a mock-up, to scale, beacon test method (see **Figure 2**). As all beacons had to be level with one another, I also constructed beacon support mounts for experimentation; these can also be seen in **Figure 2**. Over a period of 30 s, the mobile beacon (hedgehog) was left stationary; a Raspberry Pi was used to collect the output positions. Plotting these results on a time against $x_{coordinate}$ axis showed that the beacons did not have a consistent sampling rate (as indicated by the regions of higher and lower sample densities in **Figure 2**). This inconsistency was determined to be as a result of the intermittent radio frequency communication between the beacons. I altered parameters such as scanning bandwidths and radio profile which improved the performance of the spacing by 124% (calculated by comparing average sample spacing over experiments).

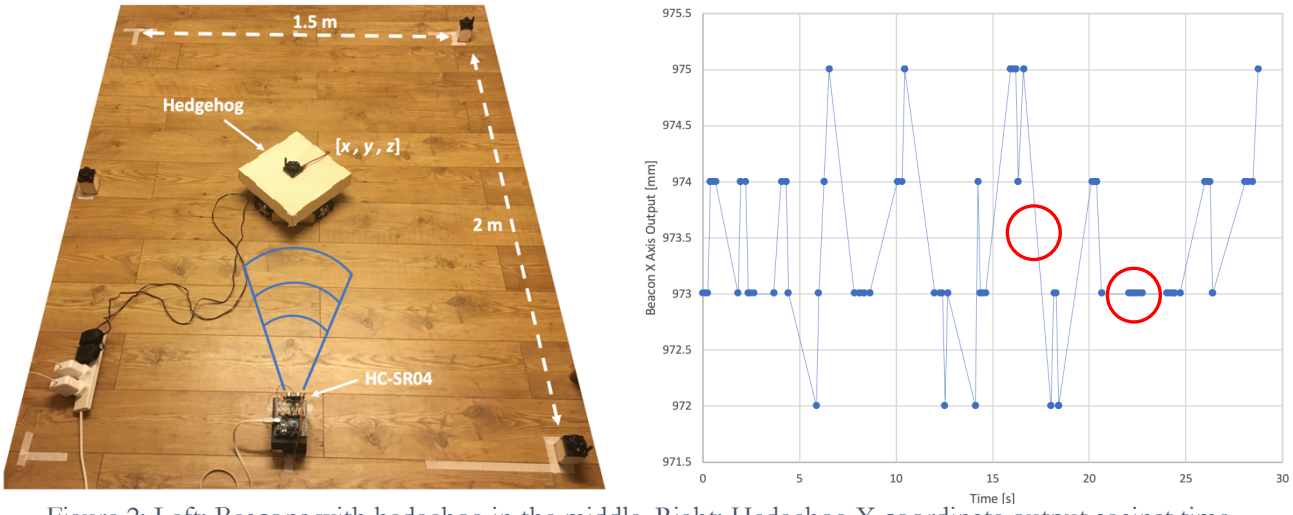

Figure 2: Left: Beacons with hedgehog in the middle. Right: Hedgehog X coordinate output against time

Using the stored values, standard deviations and ranges were also calculated for *[x,y,z]*; see **Table 2**. These results showed that the beacon outputs had similar performance in both *x* and *y* directions. Furthermore, the standard deviations indicated that the beacon performance was within the 2 cm uncertainty claim. At this period in time, the test method had not been constructed so I was unable to perform any meaningful moving/ dynamic tests with the beacons.

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	0.8	3.26
Range (mm)	3	4	13	3	4	14

Table 2: Shows the hedgehog output 'x' position data to the Raspberry Pi over a period of 30 seconds. Dots indicate discrete output.

Marvelmind also stated that the sampling frequency rate could be changed from 1 Hz to 16 Hz. Testing in the Anechoic Chamber (at the University of Southampton) highlighted that the sampling rate would not go above 8 Hz however. **Figure 3** represents a 16 Hz experiment; despite this however, it is clear to see that the hedgehog emitted pulses with spacings of 0.125 s between them. Correspondence with Marvelmind highlighted that the 16 Hz rate was achieved by filling in beacon readings with an inbuilt IMU. This however required calibrating with bespoke software before every use and was therefore deemed too convoluted a process to be useful for our project (Entry 3,4).

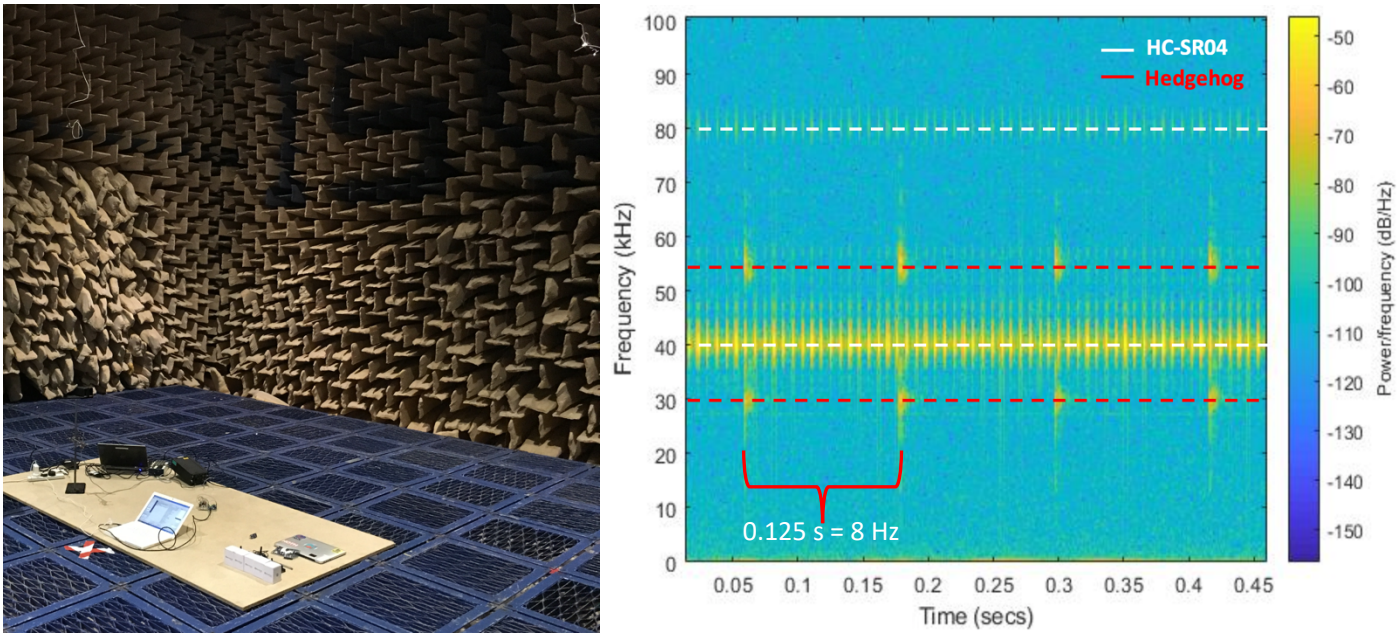

Figure 3: Left: Anechoic Chamber before experimental setup. Right: Ultrasonic Frequency plot for Beacon and HC-SR04.

Background Interference:

Eurobot competitors commonly use HC-SR04 ultrasonic sensors for obstacle avoidance. The transducer in this sensor pulses at a frequency of 42 kHz, which also produces a resonance at 80 kHz (see **Figure 3**). In order to ensure that it would not interfere with the beacon system, all previously conducted tests were also carried out with the HC-SR04 operating in the background. **Figure 2** shows the experimental setup with the HC-SR04 facing the hedgehog. **Table 2** showed that the beacon system performance was not compromised by the HC-SR04 sensor. This agrees with the information shown in **Figure 3**; the sensor bandwidth are shown not to overlap (peak power frequencies ~8 kHz apart) (Entry 5).

Sampling Update Rate:

As already demonstrated, the highest attainable sampling update rate is 8 Hz. In order to simplify the Kalman filter programming, the filter would be run no faster than the slowest sensor (to ensure constant matrix size). For this reason, it was critical that the Raspberry Pi could resolve a result from the incoming beacon data at 8 Hz.

During testing, it was discovered that the Raspberry Pi could not 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 increased to 70% of its total allowable limit. As a result, new beacon positions resolved 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) methods were tested; these improvements did not increase the CPU performance enough however. Finally, introducing a separate intermediary Arduino processor dramatically reduced the computational load on the Pi from ~40% to ~5%. This

also allowed for sampling rates of 8 Hz or more to be sent from the beacon to the Raspberry Pi (via the Arduino); this is shown in **Table 3**.

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

Table 3: 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.

Predictive Algorithm:

A rate of 8Hz is very slow for computational processes. As the Kalman filter would only be able to run at the rate of the slowest sensor, I decided to try and improve the beacon sampling rate beyond the 8 Hz limit. As our sensor fusion algorithm also used inputs from the IMU and optical flow sensors, predicting results for one sensor would allow for greater use from the other two sensors (of course, this does not consider sensor confidence weightings). My intention was to test the performance of the entire robotic system with and without the increased sampling rate (8 Hz beacon **vs** 16 Hz beacon with estimates). Discovering that the computational load on the Raspberry Pi ended up being too great, even without the beacon system incorporated, meant that I was not able to conduct these tests; this will be discussed in more detail in Section **Performance Drop**.

For simplicity, the robot only moves in straight lines; this 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); see **Figure 4**. As this method depends purely on previous movement data, it goes without saying that the prediction improves with time (as more samples are collected). I did not test sampling rates beyond 16 Hz as the number of predictions would then supersede the number of real results; a 1:1 ratio (or less) was deemed suitable.

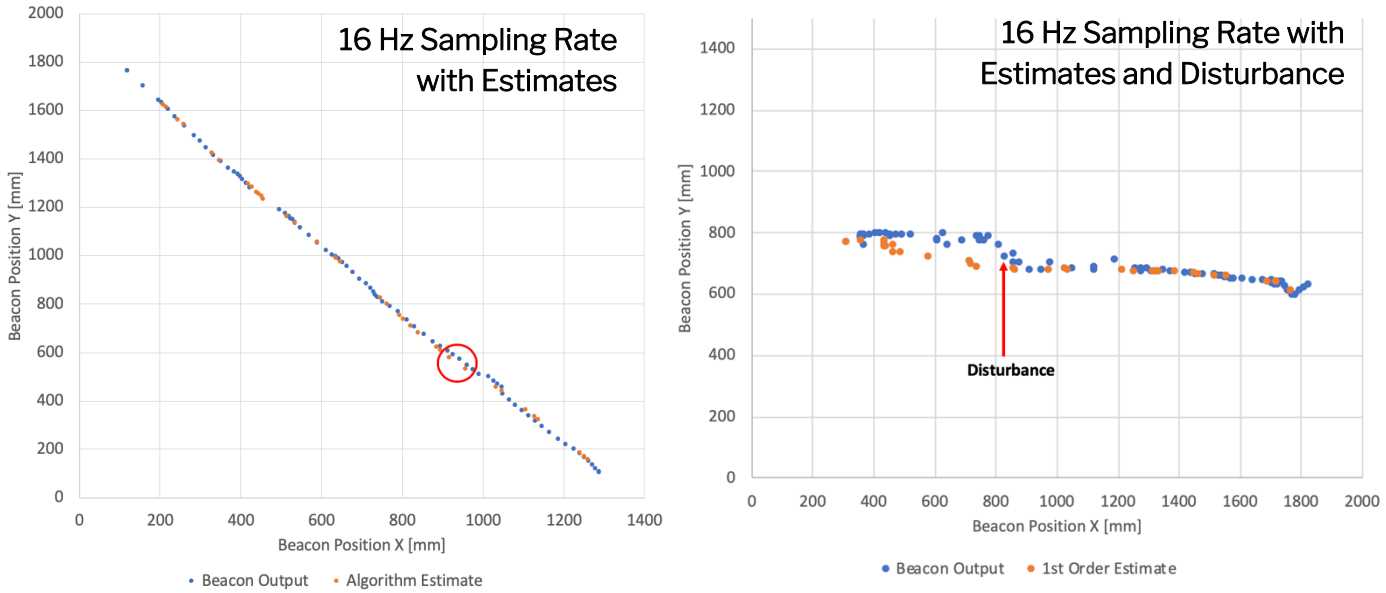

Figure 4: Left: Beacons' position as it moves. 16 Hz sampling rate (double beacon limit). Blue dots indicate a real updated position by the beacon, orange dots indicate the algorithm estimate. Right: Same as left but with disturbance ¾ way through movement.

Testing also highlighted that this method did not respond well to non-linear motion such as drift or pushing the sensor normal to its path (**Figure 4**). As this prediction algorithm was linear based, it was expected not to perform

well with non-linear data. As can be seen from **Figure 4**, the initial disturbance is picked up by the real beacon data, however, the prediction algorithm does not respond proficiently; the difference between the real and predicted results do not converge until the end of the robot's motion. In order to combat this, I changed the algorithm so that it would only use the last 30% of results taken during the motion. This improved the response to perturbations and drift, however, at the expense of using larger datasets. In order to further improve on this, I employed a support vector regression algorithm (commonly used for predicting stock market data for example), however, the increased computational load on the Pi resulted in significant proportions of the real position results being missed entirely. As the entire reasoning behind this algorithm was to provide supplementary results, this method was not suitable for purpose. Having used a more powerful processer, this method could have been implemented for improved prediction results however.

Summarising this section, the beacon was tested and prepared for implementation on the robot. Additional performance enhancing algorithms were tested and made ready for integrating into the robotic system; these would be used to improve the overall positional accuracy of the robot.

Performance Drop:

Despite promising beacon testing results, 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. External factors such as ultrasonic echoing, radio frequency and electronic interference were considered as possible culprits; CPU load and Raspberry Pi voltage levels were also considered.

The beacon was designed to sit above the top plate of the robot chassis (as seen in **Figure 5**). However, the frequency of readings was much lower than expected when compared to initial testing completed on the prototype chassis. Testing the beacon performance 1 m away from the robot chassis not only improved the sampling rate, but also the latency of the readings, suggesting the robot design was causing interference. Further investigation highlighted that the radio frequencies between the beacons and modem were being blocked by the aluminium top plate. By replacing the middle and top plates of the robot with acrylic, the beacon was then far enough away from aluminium to not cause interference. This only slightly improved sampling rate, suggesting there was another factor affecting performance.

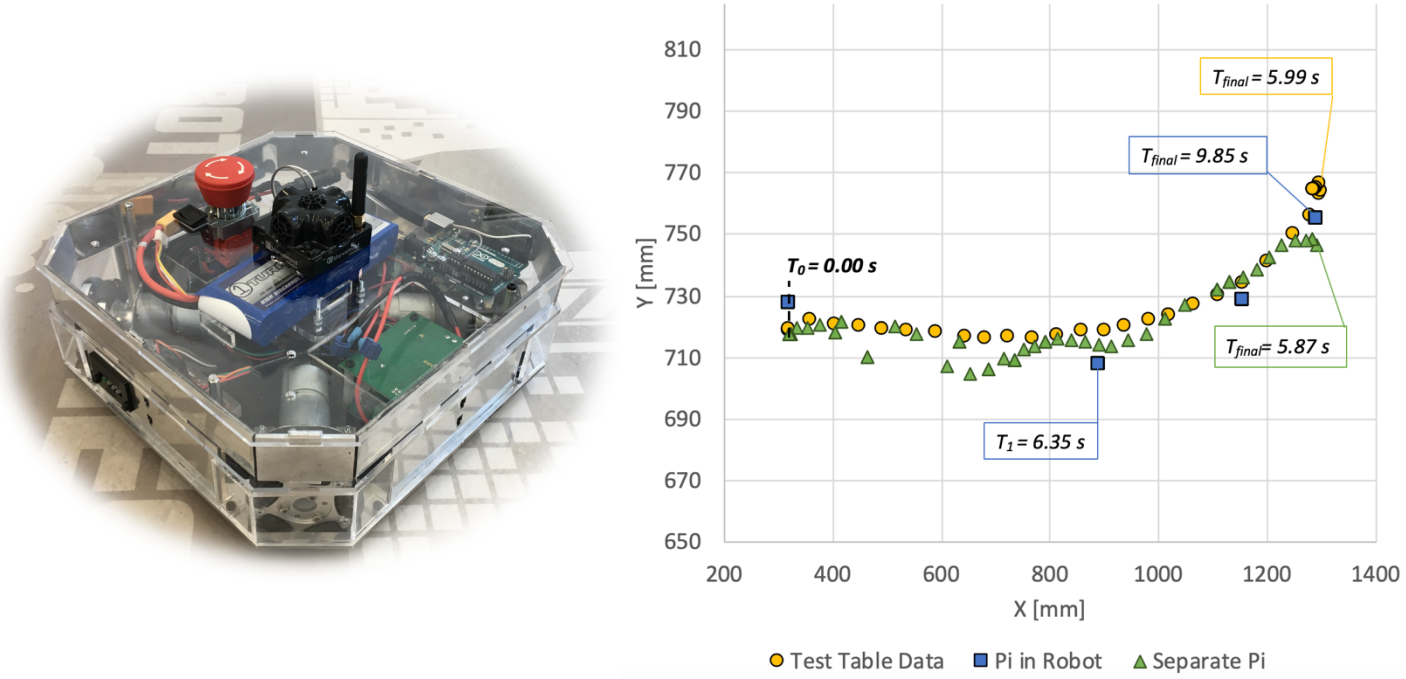

Figure 5: Left: Robot with acrylic mid and top plates. Right: Robot movement recorded by test method; two separate Pi's recording beacon data.

Signal latency and low sampling update rates still remained, evidenced in **Figure 5**. The path taken by the robot (data recorded by the test table marked in blue) has a duration of 5.99 s. The orange data points show the frequency

and timing of the beacon readings when connected to the robot. Here, the first beacon reading was received by the system at 6.35 seconds, 0.36 s after the robot had finished its path. Additionally, only four position updates were recorded by the beacons, which is too few to be used in the Kalman filter, requiring multiple readings per second.

It was suspected that high computational load could be the cause of the infrequent readings. Beacon results were also collected on a separate Raspberry Pi to determine if this was the case. As is illustrated in **Figure 5**, there are significantly more data points (green markers) when using a separate Raspberry Pi. The final beacon reading was recorded 5.87 s after the robot started moving, which closely corresponds to the time at which the robot stopped moving, at 5.99 s.

These results showed that, the current robot is not capable of processing the beacon data fast enough to be useful in this system, but that the beacons would be an appropriate sensor if connected to a system which could manage the computational load.

INNERTIAL MEASUREMENT UNIT

Orientation may be determined via odometrical means, with static reference points, or by gyroscopic measurements. Currently, the robots made by second year students determine heading by measuring the difference in wheel rotations; this method is prone to errors such as wheel slip, encoder inaccuracies and motor step size. Incorporating a reliable orientation system with minimal vulnerability to aforementioned factors was critical. After careful design consideration, I eventually selected the Adafruit BNO055 inertial measurement unit (IMU) (**Figure 6**) due to its embedded sensor fusion capability; this is discussed in more detail below.

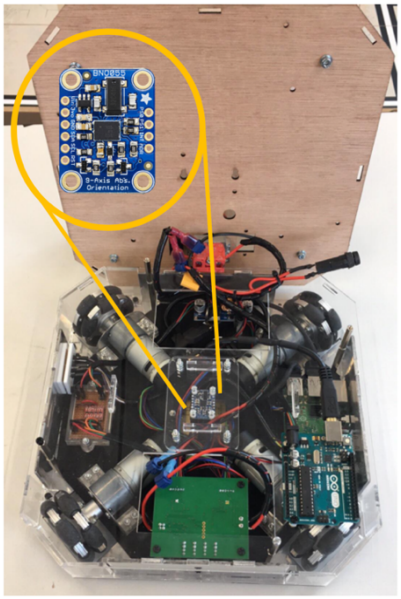

Figure 6: Centrally positioned IMU.

Design Considerations:

Intelligent design can often save time in other areas such as in programming and manufacture. With regards to selecting our orientation method, I set several key design parameters that had to be met. These parameters eventually influenced the robot design, manufacture and programming (which I was responsible for):

- **Sensor Placement:** Ensuring the robot's centre of rotation aligns with the centre of the orientation method reduces the amount of post processing calculations required.
- **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 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.
- **Limited Computation Load:** Appropriate sensor selection would limit the amount of computations required by the Raspberry Pi.

Sensor Selection:

As mentioned, the Adafruit BNO055 sensor was selected because of its unique onboard sensor fusion capability. IMUs typically employ a gyroscope, accelerometer, magnetometer and temperature sensor, however, do not

perform and any sensor fusion operations. This sensor was small enough to fit into the centre of the robot, adequately reduced dependency on all other sensors (not having to use two beacons or optical sensors for orientation for instance), and did all of the necessary computation on board.

Clock Stretching:

Normal Raspberry Pi I2C communication does not allow for clock stretching, this can cause several issues during data parsing between itself and other sensors. During periods where the slave (sensor) is not able to cooperate with the clock speed set by the master (Pi), the baud rate is reduced (clock stretched). If the masters clock rate is not reduced, then large sections of the data can be corrupted or even missed entirely.

The Raspberry Pi UART protocol facilitated clock stretching however, for the Pi 3B+, the inbuilt Bluetooth connectivity was assigned to their pins. In times of high computational load, UART communication would compromised resulting in loss of data transfer from the sensor. In order to fix this issue, Bluetooth was disabled and the UART pins reassigned for the BNO055 sensor.

Out of all the small scale IMUs that I came across, the BNO055 was the highest performer. For this reason, it was advantageous to integrate it into the robotic system. Employing the UART protocol therefore allowed us to collect accurate heading measurements with minimal cost to the Raspberry Pi computing power. This did however take significant time as I had never used UART communication before, nor had I needed to disable core microprocessor functionality.

Calibration:

As documented in the Design Report, the BNO055 sensor required calibration before use to ensure accurate orientation measurements; for this reason, I wrote a program that would take the user, step by step, through the calibration process. Upon running this code, the Pi would start storing readings from the embedded gyroscope, accelerometer and magnetometer. As not all future users would know how to use, or have access to the Raspberry Pi, I needed visually represent the calibration state. For this reason, I incorporated four LEDs into the robot design. LEDs would light up as the embedded sensors calibrate; the image below is an illustration of this process. As well as this, I also did all the relevant soldering and manufacturing required for the integration into the robot.

BNO055 Performance:

As was the case for the beacon system, the IMU performance had to be characterised; this was important for the selection of values for the covariance matrix in the Kalman filter. Testing identified that sensor drift was insignificant (**Figure 7**). The conclusion was made on the basis of the readings not changing at all when left stationary f or a period of 60 seconds.

Dynamic tests were also conducted; the robot was programmed to spin on the spot while the Raspberry Pi would collect the orientation data from the IMU. After six consecutive revolutions, the sensor readings stayed consistent with no signs of showing drift (this can be seen in **Figure 7**). A mean difference of 2^0 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 $n = number\ of\ samples$. 2^0 equates to 2.28 mm variance between readings (for a robot footprint of 265 x 265 mm). This post processing step was quite laborious as it first involved normalising all 6 of the revolutions so that they could then be plotted over one another as shown in **Figure XX**.

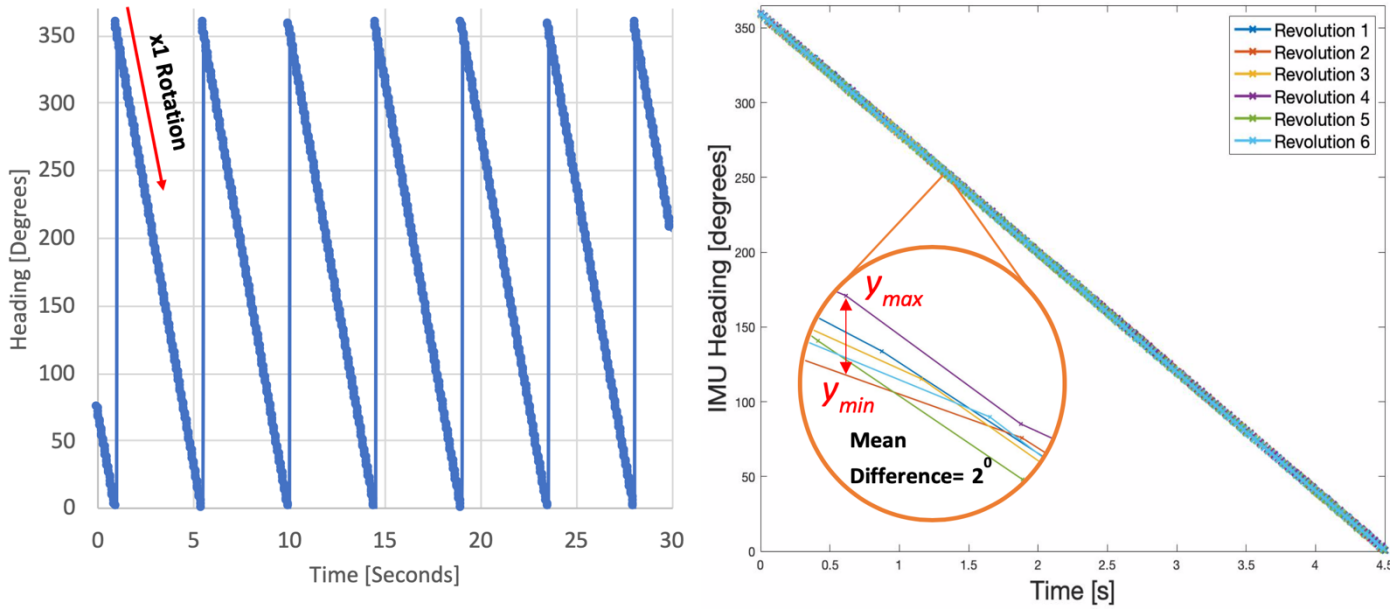

Figure 7: Left: Heading against time. Robot constantly rotating on the spot. Right: Matlab program to calculate Mean difference.

After this, the maximum differences across the entire curve could be calculated. The Matlab program I wrote would, at every recorded sample, compare the maximum difference between the 6 curves. As none of these samples were recorded at the exact same time, the program would then interpolate between known points (for the other five curves) in order to determine the highest and lowest y axis values at that respective time.

Furthermore, average angle measurement error was calculated as being as 0.05^0. 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 0^0, 90^0 and 270^0; this experiment was conducted 5 times, with new calibrations for each instance to test the variability of the system. As mentioned, the characterisation of this sensor allowed my team members to develop the covariance matrix in the Kalman filter. As the sensor performance directly related to the accuracy of the final system, it was important to numerically determine the performance of this sensor.

OPTICAL FLOW SENSOR

During earlier stages of the project, it was not my responsibility to conduct research or testing for the optical flow sensor. During Semester 1, Justin had purchased the ADNS-3080 optical flow sensor and conducted some tests to show its performance. Having not received any clear update on its' status until the later stages of Semester 2, and with the urgency of the final robot build increasing, I decided to start conducting some research as well. Discovering that this fixed lens sensor was used for quadcopter flight stabilisation at a height of 2 m from the ground (i.e. not usable for our application), I sought alternatives to implement in the robot design. After some research into the optical sensors used in computer mice, I decided to purchase and test one in order to determine whether it would be suitable for our application (details catalogued in my **Design Journal 6**). Preliminary tests with an Arduino showed huge potential with the mouse experiencing no drift while stationary, as well as having a step size of only 1 pixel (translating to a minimum unit of measurement was smaller than a millimetre).

Programming and Testing:

As the Raspberry Pi is a computer, keyboards and mice can be connected to it for interfacing; my research highlighted that there is in fact a file that is created for each device connected to it. By writing a python script to

scan for the positional results in this file (outputted [dx,dy]), I was able to collect the movement results of any USB connected computer mouse. As each increment of *1* in the file directly related to a single pixel, my next task was to ensure that these readings corresponded to real distance travelled (in mm). In order to correctly scale the computer mouse sensor, I first had to convert from the computer mouse specifications, which were presented as dots per inch (DPI); this is effectively the amount of pixels that fit into an inch of space. High performance gaming mice boast DPIs as high as 12,000; this allows for higher degree of control and more sensitive mouse control (however could cost as much as £150.00). Preliminary testing highlighted that this scaling seemingly worked with measurements of objects being roughly correct (**Figure 8**); at this stage I also noticed that this sensor seemed to drift over larger movements, as can be seen on the right hand image of **Figure 8**.

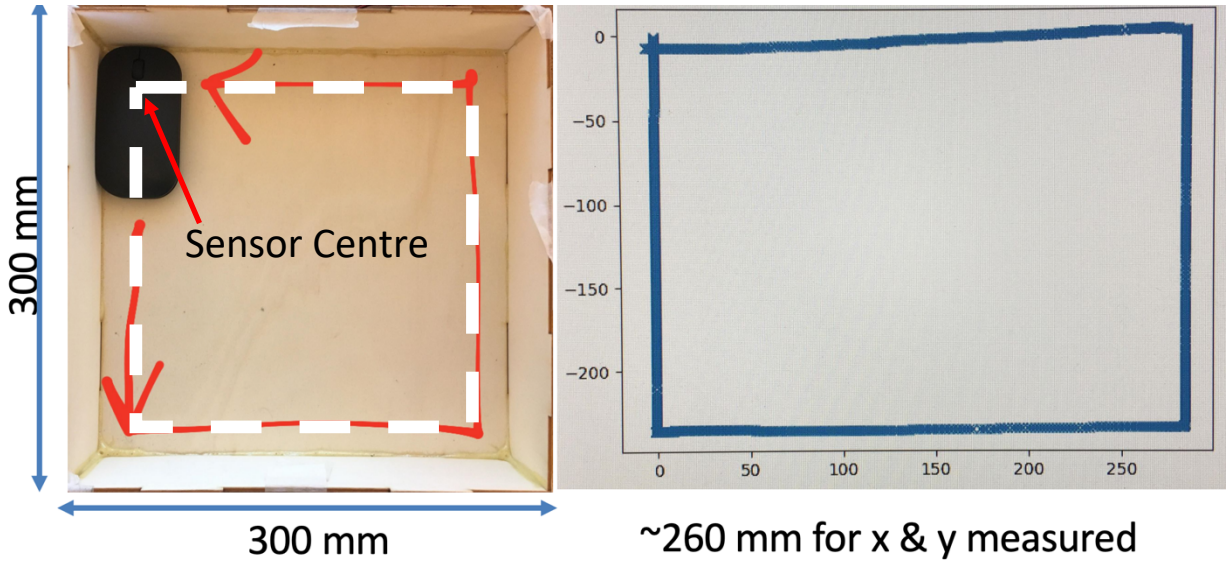

Figure 8: Left: Initial testing of computer mouse sensor. Right: Computer mouse output as moved around box interior.

The next stage of sensor preparation was to numerically determine whether the optical sensor was suitable for our robot. Together with the help of Justin, we both measured objects, of known distances, and compared the readings output by the sensor against them. **Figure 9** shows one example of an object that we measured. Plotting the

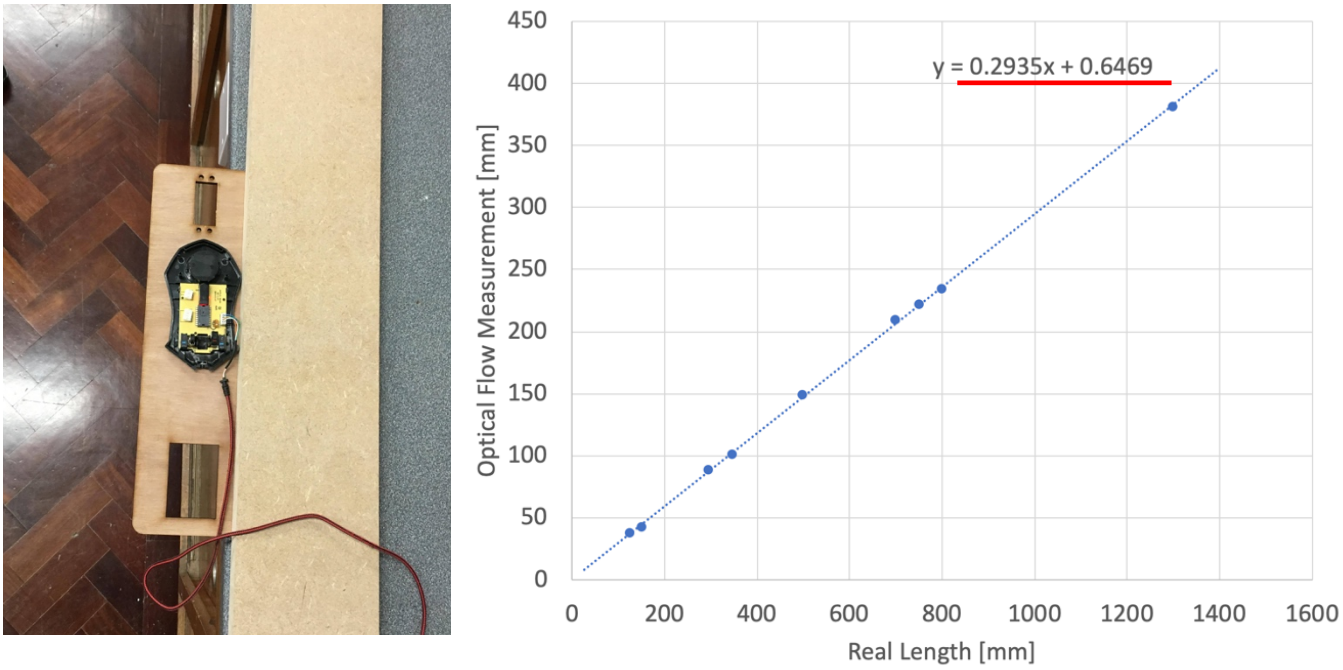

Figure 9: Left: Example of an object that was measured. Right: Real against Mouse measurements plot.

measurements against the real lengths highlighted to us that there was a linear relationship between all values. Using a least squares regression fit (**Figure 9**), we were able to numerically determine the scaling factors necessary to implement into the optical sensor program.

The gradient and constant of the $y=mx+c$ linear fit equation were incorporated into my program and all initially measured objects were tested again. All measurements were within **3 mm** of the correct values. We also tested other objects that had not been measured previously and these results still held. In effect, I had written a program for a sensor that could measure millimetre accuracy, while only costing £25.00.

Manufacture Phase:

After completing the programming, Justin then took charge of the manufacturing of the system that would eventually be integrated into the robots final design. I did however support him during this process as it was important to ensure the optical mouse was the correct distance away from the ground with sufficient force applied to ensure the sensor would always read a value (if too high off the ground the optical sensor lose focus and no longer record positional updates). Furthermore, the sensor required suspension which would allow the system to be pushed off the ground should the robot drive over a bump. I believe that our design discussions helped Justin arrive to a simple, robust solution to this problem.

PROGRAMMING ORGANISATION & OPTIMISATION

Aware that the programming element of this project was extensive, it was important to keep all the work that I produced as clear and organised as possible. Not only did this help with ensuring that no mistakes were made, it also allowed Caroline to be able to call the functions with ease. As can be seen in **Figure 10**, everything relating to the functionality of the sensors could be done with 6 simple commands.

Import the detailed sensor codes for use below

Call Setup( ) to turn on all three sensors

Call any of these functions to receive the position of the sensor

Calibration and Optic Sensor Resets needed for other purposes

```
from marvelmind_cy_mm import MarvelmindHedge
from opticalflow import Optical_Flow
from BNO055 import BNO055

####### SETUP PHASE FOR SENSORS #######
optic = Optical_Flow()
hedge = MarvelmindHedge(tty = "/dev/ttyACM0", adr=4, debug=False)
bno = BNO055(serial_port='/dev/ttyS0', rst=18)

def Setup():
    optic.start()       # Start thread
    hedge.start()       # Start thread
    bno.connect_IMU()   # Start thread
    sleep(2)

####### FETCH POSITION FROM SENSORS #######
def Beacon_fetch():
    location=hedge.position()
    return location

def Optic_fetch():
    location=optic.position()
    return location

def IMU_fetch():
    location = bno.position()
    return location

####### OTHER IMPORTANT FUNCTION #######
def Optic_reset():
    reset_optic=optic.reset()
    pass

def IMU_calibrate():
    bno.calibrate()
    pass
```

Figure 10: The sensor script.

Furthermore, I acknowledged that the computational load on the Raspberry Pi should be considered so I also streamlined my scripts by deleting unnecessary commands and keeping the codes succinct. I also compiled my scripts into Cython (a C compiler extension in Python) which slightly reduced the computational load on the Pi, and significantly increased the speed of the computing operations. I also ensured the wiring for these sensors (and LEDs for IMU calibration) were organised appropriately so as to minimise their space requirements in the robot, as well as be easy to identify if required.

KEY ACHIEVEMENTS

My personal contributions facilitated the integration and application of the optical sensor into the final robot design. The programming and testing that I conducted enabled Justin to begin developing the component to be integrated into the robot. As the chassis design was dependent on the design of the optical sensor, it was imperative for this body of work to have been conducted quickly (so as to begin testing sooner). Furthermore, the chassis required waterjet cutting in the EDMC; as several GDPs require parts to be manufactured at similar times, lead times can often be longer than expected. I believe that it was an achievement to have identified the urgency of this components implementation, and to have successfully acted on it despite my role primarily concerning the beacons and IMU. Within the space of two days, we were able to move from the research phase, to the optical sensor manufacture; additionally, the chassis design was completed and sent to the EDMC for manufacture in this time.

Furthermore, I believe that the characterisation of all sensors, as well as the improvements made to the beacons with the predictive algorithms deserve recognition. All three of the sensors discussed in previous sections were shown to be suitable candidates for our system which validates the design process taken for each of them (from research to implementation). Both the IMU and computer mouse sensors provided exceptional positional results for their respective prices. I believe this should be regarded as one of my achievements, not only for selecting the sensors, but also for developing them further with suitable code. Similarly, I believe that my work on the beacons may be brought forward to following autonomous robot/ Eurobot projects. After characterising and optimising the beacon performance, I was able to further extend this with the use of a predictive algorithm. Had a more suitable microprocessor been selected, I would have been able to incorporate this improved sensor into the Kalman filter and, as a result, contribute to the development of an even more robust and accurate system.

While non-technical, I believe that another achievement is the securing of funding from Boeing. This facilitated the purchase of extra sensors and electronics equipment that aided our design and research process, as well as saved time during testing. For example, using a touch screen to interact with the Raspberry Pi inside the robot meant that we would not have to keep reopening the robot to connect to it. Not only did this save time, but it also reduced the likelihood of us accidentally damaging internal components. We were also able to purchase an extra LiPo battery and charger. This enabled us to have the robot running constantly over the course of a testing day without recharge wait times. Battery charging would have been incredibly inconvenient as the only publicly available LiPo chargers are located in Boulderwood Campus; our testing table was in Highfield campus.

CRITICAL REVIEW

SUCCESSES

As listed in the P2 form submitted at the start of the year, our project objectives were as follows:

- Create a low-cost final design with respect to leading Eurobot competition teams.
- Design a system that can be reused and built on by future teams entering Eurobot.
- Develop a reliable method of verifying the accuracy of the robot and comparing it to previous localisation methods.
- Design a robotic platform that can determine its position on the Eurobot table with millimetre accuracy over 90 seconds.

Comparing our final deliverables to these four goals, I would be happy to say that this project was a success. Our final robot design was indeed low cost relative to the leading Eurobot teams. We also designed a system that is reusable and able to be built upon in future years. Furthermore, we developed a very reliable testing method with which we were able to characterise the performance of our and Part II students' robots (as is explained in the

Design Report). The only shortcoming, as compared to the initial objectives, is the precision to which our robot localises. While our goal was for millimetre accuracy, we were able to achieve 25 mm accuracy. The results from our project highlight that the combination of one optical sensor from a mouse combined with PID control can significantly improve localisation performance. The innovative design and application of the optical sensor, as a replacement to the motor encoders, not only improved the accuracy of the system, but also eradicated wheel slip errors entirely. As the system did not rely on wheel rotations to localise, the robot was able to correct itself even when pushed significantly off course; this is something unique to our robot and goes beyond the objectives outlined in our initial objectives.

Our project was further improved beyond the scope of our objectives with the addition of the testing table. While we understood that a means of validation was required to test the performance of the robotic system, we did not plan to build a testing frame using vision sensing at the start of this project. The acquisition of further funding facilitated this addition and, as a result, greatly improved our ability to characterise the robots localisation performance.

In the context of Eurobot, achieving millimetre accuracy is challenging, especially when dealing time and financial constraints. For top performing teams to spend £20,000.00 on their systems and still not achieve this level of accuracy is indicative of the difficulty that is localisation. From the offset, our group claimed that this level of accuracy could be met at a significantly lower cost. While the project has not ended at a point which we are able quantifiably confirm this claim, it does strongly suggest that it is possible to achieve with only minor adjustments to our system. For this reason, I believe the project to be a success.

SHORTCOMINGS

As mentioned, our final system was not able to localise with millimetre accuracy. I believe the main reasoning for this was that we, as a group, did not adequately select a microcontroller at the start of the project. Initially, we made the binary decision between using a Raspberry Pi or an Arduino. Having only used Arduino Uno's in the past (in Part II), none of us had the experience to suggest more suitable alternatives to the Raspberry Pi. We did acknowledge that the Pi had significantly higher computational power (as well as having extensive online support) and, as a result, we chose this to be our primary microprocessor. It was only towards the end of the project, when more components were added to the system, that we realised the computational load on the Raspberry Pi was too great. Attempts to solve this issue were made (such as introducing the Arduino intermediary to the beacons), however, it was determined that the CPU usage was still too high even without the beacons and IMU. The addition of the Kalman filter was intended to improve the localisation accuracy. Due to the dramatically reduced resolving rate however (caused by increased computational load), the system performance in fact worsened.

It is difficult to say whether choosing a more powerful microprocessor would have enabled us to create a system that employed a Kalman Filter, beacon system, IMU, optical flow sensor and PID controller. Most commercially available alternatives with higher computational power are more complicated to operate and do not offer as much online support compared to Raspberry Pi's. This may have been too great a leap in experience for us and may have resulted in poorer performance rather than better. As previously mentioned, this project was self-proposed meaning that we had no previous work to build upon. Perhaps if we followed on from a previous GDP, we could have been forewarned about CPU loading issues. Had we reached the testing phase earlier, we may have been able to replace the Pi with a superior alternative and, therefore, be able to build a robot employing a Kalman Filter, beacon system, IMU, optical sensor and PID controller. Furthermore, this would have allowed for the testing of more advanced predictive algorithms (used to increase the sampling rate of the beacons); this may have further improved the performance of our localisation system.

INNOVATION

Throughout the project, options for all aspects of the design were considered. An example of a unique aspect of the design is the inclusion of a repurposed mouse optical sensor. This, alongside research into existing products and methods, allowed for the development of a bespoke robotic system fit for the Eurobot competition. Meetings with stakeholders ensured that both the robot and test method were developed in the most suitable way for their required purpose. An example of this being minimising the height and perimeter, to maximise the available space for future users. The importance of the financial aspect of the final robot, and the validation of its results were understood from the start of the project, and therefore were recognised in the design brief and objectives.

The overall cost of the robotic system amounts to £330.72 which, in comparison to other robotic systems with similar performance, is significantly cheaper. The test method amounted to £716.12 which, for a system that measures millimetre accuracy, and can be used for numerous future projects, is also very good value.

PROCESS

In order to ensure each team member was able to work on an aspect of the project simultaneously, a prototype of the robot was constructed. This allowed for the drive system to be developed in tandem with the sensor testing. By building the prototypes from MDF as opposed to Aluminium, we were able to save on material and manufacture costs, as well as ensure that our design process was more sustainable. As further funding was awarded, a test method was also constructed. Unfortunately, the development of the optical flow sensor took longer than expected. This delayed the development and manufacture of the chassis which had further implications to time available for testing. As a result, sensor fusion and PID control implementation was not tested until the final stage of the project.

Research showed there was no suitable existing test method therefore one was specifically developed. Theory was applied for choice of material, and to reduce waste additional bracing was only added where required. The aluminium extrusion has the added benefit of being versatile and reusable in other, future projects.

COMMUNICATION

This project communicates its' intent in various forms, with varying levels of detail. As differing project stakeholders required varying levels of detail, it was important to communicate our project in multiple ways. Examples of this include communication with supervisors who required high levels of detail (Design Report and Design Journals) versus Part II students who responded better to top level explanations and more visual queues (Video). The impact of our communication was improved by ensuring emphasis on all the same points throughout all deliverables. Further to this, the graphical design of our project was consistent throughout the deliverables. This gave the project a unique identity and further strengthened the connection between the forms of communication.

REFERENCES

[1] M. Peca, "Ultrasonic Localization of Mobile Robot Using," *Eurobot 2009, Ccis 82*, pp. 116–130, 2010.

[2] J. Vander, D. Hagel, M. Blaich, and O. Bittel, "Mobile Robot Localization Using Beacons and Kalman Filter Technique for Eurobot Competition," vol. 33, no. June, 2009.

[3] V. Durst, D. Hagel, J. Vander, M. Blaich, and O. Bittel, "Designing an omni-directional infrared sensor and beacon system for the Eurobot competition," *Commun. Comput. Inf. Sci.*, vol. 161 CCIS, pp. 102–113, 2011.

[4] RobotShop, "RPLidar A1M8 - 360 Degree Laser Scanner Development Kit." [Online]. Available: RPLidar A1M8 - 360 Degree Laser Scanner Development Kit. [Accessed: 05-May-2019].