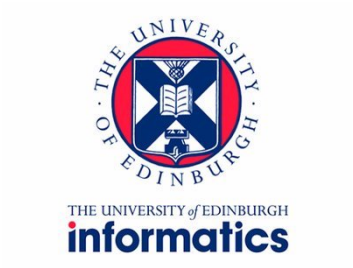
Robotics: Science and Systems

Practicals - Final Report

Construction, modelling and implementation of a LEGO robot for autonomous localisation



Lukas Schäfer
School of Informatics, University of Edinburgh
Semester 1, 2018/ 2019

November 22, 2018

Abstract

This report outlines the mechanical design, construction as well as software implementation of a LEGO robot to perform an autonomous localisation task where highlighted points of interest (POI) need to be identified and communicated in a known map.

Using a four-wheel differential steering mobile robot design combined with waypoint navigation, obstacle avoidance techniques based on IR, sonar sensors as well as bumpers and particle filter localisation, reliable POI detection and orientation in the arena was achieved.

However, the probabilistic localisation was not without its inconsistencies and the robot failed to achieve the determined task in the given time limitation.

Contents

Chapter 1

Introduction

The task of this practical project involved the construction of a robot using LEGO and the implementation of appropriate software in order to perform an autonomous exploration in a known and static environment.

1.1 The task

The robot starts at a fixed deployment location with known orientation inside the map, illustrated in figure 1.1, of about 428cm by 320cm containing walls as obstacles. Throughout the map points of interest (POI) will be spread out at arbitrary locations. Each POI is represented by a reflective tape on the floor of approximately $25cm \times 25cm$.

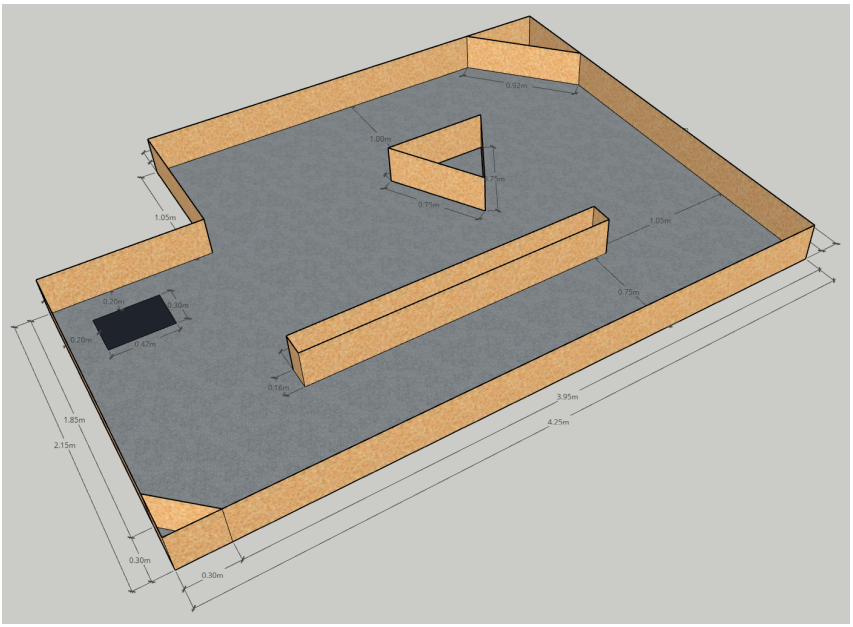


Figure 1.1: Illustration of the given map with highlighted deployment location (black mark) - taken from `https://uoe-rss.github.io/project`

For the final task, the robot has to move around the map and find three POIs. Whenever a POI is detected, the robot is supposed to stop with its centre of mass in the middle of the POI and simulate a communication with a satellite. Therefore, the robot is expected to align an antenna pointing towards a satellite dummy placed on the ceiling in a

known location. After finding and communicating all three points of interest, the robot has to move back towards its initial position to finish the task. For the entire process, there is a time constraint of 5 minutes.

1.2 Our approach

In order to achieve this task, we designed a four-wheel differential steering mobile robot with various sensory technology to be able to avoid obstacle collision and reliably localise itself in the known environment. Arguably the biggest challenge was reaching a sufficient precision of odometry and measurement estimates requiring gradual changes in the modelling and mechanical design throughout the entirety of the project.

In the following chapter, we will outline our approach in more detail considering mechanical design, involving actuators, sensors and the general physical architecture, as well as the software implementation from the general control program over odometry tracking and modelling up to localisation and navigation techniques.

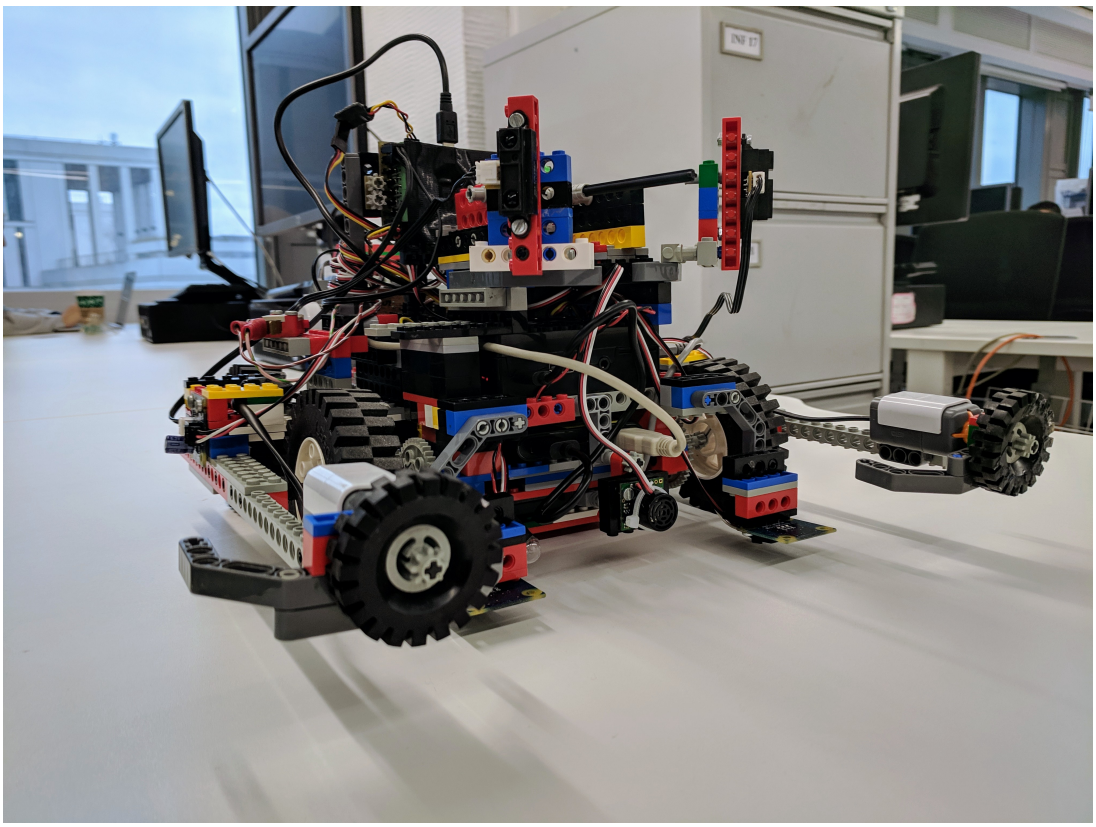


Figure 1.2: Picture of our final robot design

Chapter 2

Methods

2.1 Mechanical design

2.1.1 General architecture

The robot built for this project is almost exclusively constructed using LEGO. Out of its four large wheels, two are located on each side of the robot and powered using the same motor. Therefore, turning can be achieved by differential steering through the difference

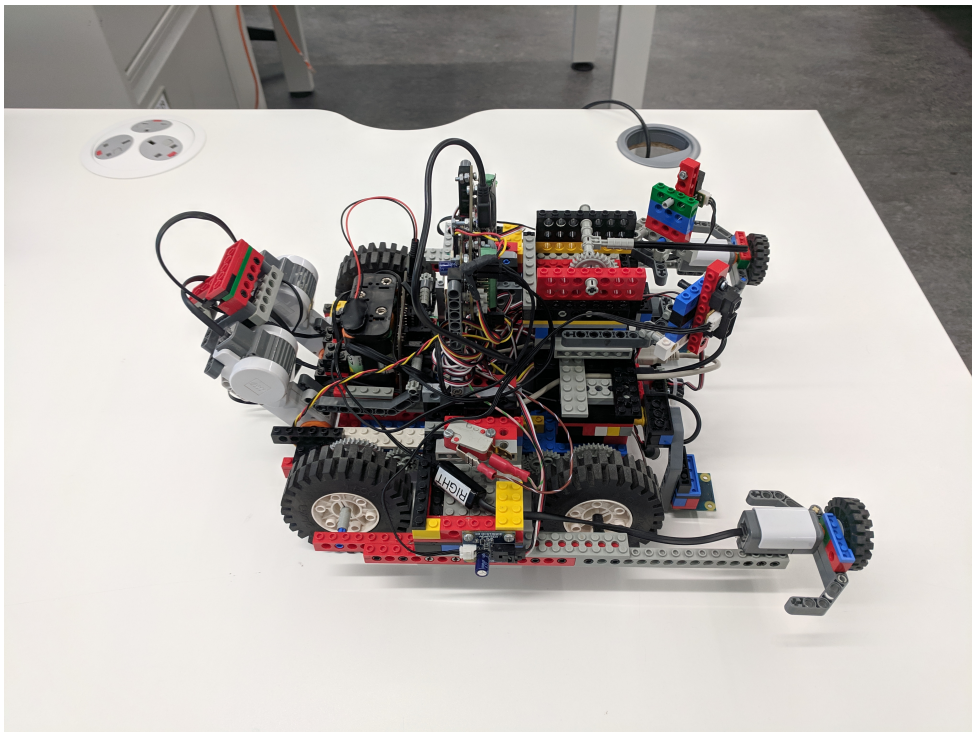


Figure 2.1: Top-side view of the entire robot

in directions of wheels on the left and right side. The robot in total is about 36cm long and 26cm wide.

All wheels are connected to the central corpus of the robot which mounts all the electronic components necessary to power and control the robot hardware. The core of the robot is a Raspberry Pi 3. Besides this computer, the robot mounts a Phidget 888 Assembly as the interface for any sensor and control information, a servo-motor-board, a

USB hub for connectivity, a motor-board powered by an AA battery pack and a powerbank for power supply of the USB hub and Raspberry Pi itself. The placement of these parts can be seen in appendix A.

One of the main challenges in designing the physical architecture of the robot, is to incorporate all these components on the LEGO robot without disrupting the balance or weight distribution or introducing moving parts which could all introduce inconsistencies in robot movement. We therefore place all these heavy components centred in the middle of the robot as close to the wheel axes as possible. This approximately even distribution of mass in the centre of the structure is essential to avoid friction and imbalances.

2.1.2 Gearing and actuators

In order to power the differential steering configuration, the wheels to each side are powered using an individual LEGO NXT motor.

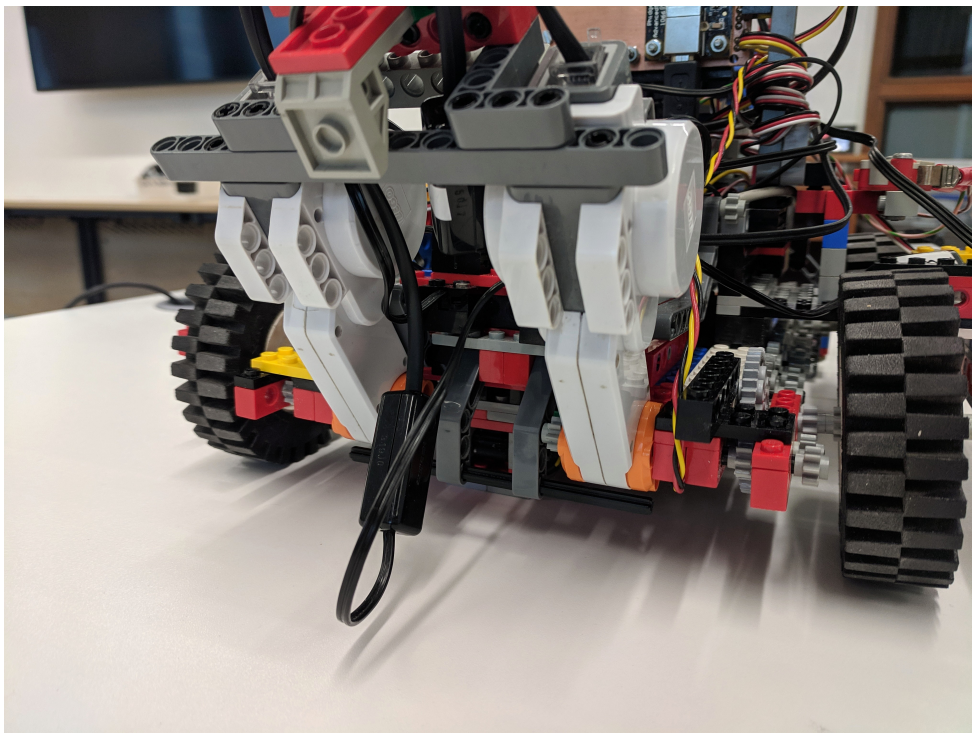


Figure 2.2: Backside with motor attachment

An essential challenge regarding the gearing of the wheels is managing the trade-off of torque and speed. Initially, we build a motor-wheel connection translating the speed 5:3, so that five rotations of the motor axis would result in three rotations of the powered wheels. However, we found that this results in a considerable speed causing small, but unpredictable sliding whenever stopping or changing the direction. Therefore, we modified the gearing in two iterations to a 5:1 translation further decreasing speed for higher torque to improve robot motion stability.

Both motors are placed at the backside of the robot connected to a gear-chain at the side of the robot, illustrated in figure 2.2. These gears translate the axis rotation from the motor to both wheels at the respective side of the robot.

Wheel stabilisation

During our experiments, we noticed that the mass, centred in middle of the robot, was sufficient to cause bending so that wheel and motor axes were not properly aligned. In order to minimise such tilting, we added supporting beams on both sides of the robot which stabilise the axes, keeping them in parallel to the ground, and minimise the lever-effect otherwise introduced by the mass distribution.

2.1.3 Antenna alignment

In order to align the antenna towards the satellite with high precision, we use the servo-motor for vertical movement. In addition to its precision the servo-motor has the advantage that it directly rotates to a given input orientation, so that no calculations are required for the rotation.

To horizontally align the antenna towards the satellite position, however, we rotate the entire robot. As this rotation is executed using differential steering, we need to compute the rotation of the robot based on its current position, its orientation as well as the satellite's position. Such an angle can be computed using simple trigonometry calculations.

2.2 Sensing

The robot needs to be able to avoid obstacles in the map, localise itself and navigate around while constantly searching for points of interest. All these tasks require information about the robot's surrounding which can be obtained by various sensory technology. Our robot incorporates two infrared (IR) sensors, a sonar sensor, two whsiker-like switches, a hall effect sensor for odometry as well as two light sensors to detect the reflective POIs.

2.2.1 Sonar sensor

The sonar sensor is placed at the front of the robot with its axis coincident with the longitudinal axis of the robot. It provides distance measures with respect to a narrow cone in forward direction. While this centred distance measurement is certainly useful for obstacle detection, we found that its estimates are varying significantly making it hardly usable for the localisation.

Even after collecting sonar distance estimates with respect to real distance measurements and fitting a linear model to it (given that the sensor already tracks the distance, we found that a non-linear model would lead to unreasonable estimations), this variance remained. Let d_S be the distance measurement of the sonar sensor and d be the estimated distance, then $d \approx 1.124 \cdot d_S + 0.2836$[1].

2.2.2 IR sensors

The sonar sensor is unable to detect any obstacles to the sides of the robot due to its placement. Therefore, we added two IR sensors at the front oriented at an 45° angle to the longitudinal axis of the robot, pointing towards the left and right respectively.

[1]For the underlying data of the fitting and a plot, see appendix B.

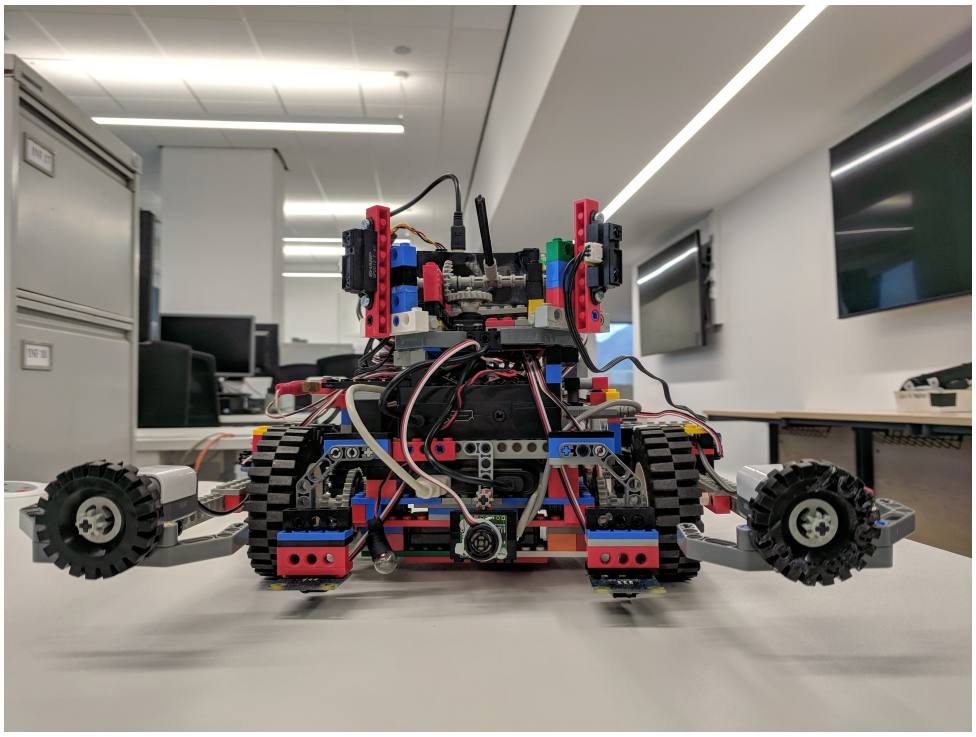


Figure 2.3: Robot front with IR sensors, sonar sensor, light sensors and light bulb

As the IR sensors provide voltage measurements, we need to translate those to a distance estimate. It can be generally noticed that the sensors are inaccurate above a distance of about 80cm and below 10cm, but provide a precise measurement in the given interval. This information can also be found in the information sheet of the IR sensor used[2]. Therefore, while fitting a model, illustrated in appendix B, to estimate the distance using the IR measurements, we only considered distances in $[10, 80]$. The final model is the following where d is the estimated distance and V_{IR} is the measured voltage: $d = \frac{6408}{V_{IR}} - \frac{4991}{V_{IR}^2} - 2.669$. This model is based on the assumption that the measured IR voltage is inverse proportional to the distance, which is stated in the information sheet.

2.2.3 Bumpers

In addition to the visual sensors, we also added a switch connected to a bumper as an extension to each beam at the robot's sides. These are able to prevent collision with objects undetected by the other sensors.

At the beginning, these switches had a tube connected. However, we found that this structure was unstable and led to some connections remaining undetected. We replaced the tube with a more stable bumper to have reliable detection of any frontal contact.

2.2.4 Hall effect sensor

All the sensors mentioned so far provide measurements about the robot's surrounding. However, such information is sometimes unavailable or not trustworthy due to e.g. large open space making it alone insufficient. Therefore, we also require information about the odometry, i.e. the robot's movement.

[2] `https://engineering.purdue.edu/ME588/SpecSheets/sharp_gp2d12.pdf`

In our project, this is achieved by a hall effect sensor detecting rotations. We connect its axis with the smallest available gear to the gear-chain on one side of the robot, illustrated in figure A.8, in order to get frequent and precise estimates regarding the rotation of the robot's wheels. However, discrepancy between the measured rotation and the movement of the robot can be introduced due to slipping or stuttering of the wheels.

Using this odometry information, we can build models of motion for the estimated displacement and therefore position of the robot. Let θ_t be the orientation at time t and η the hall-count since the last model update, then the following can be defined:

- $x_{t+1} = x_t + K_d * \eta * \cos(\theta_t)$ where x_t is the x-location at time t, $K_d = 2.6$ is a fitted constant representing the distance travelled (in cm) for each hall-count during forward or backward displacement
- $y_{t+1} = y_t + K_d * \eta * \sin(\theta_t)$ where x_t is the x-location at time t and $K_d = 2.6$

We can also estimate the rotational displacement as $\theta_{t+1} = \theta_t + \eta * K_r$ where K_r is the expected rotation (in degree) for each hall-count. K_r^l for left rotation is set to 8.3 while K_r^r for right rotation is set to 8.6.

2.2.5 POI detection

Lastly, the robot also mounts two light sensors. These are located at the front left and right of the robot about 1cm above the ground and essential to detect the reflective tape of POIs. To be able to deal with varying lighting conditions, we mount a light bulb close to the light sensors which makes the reflective tape considerably more noticeable. However, as the light bulb could not be centred in the middle between both light sensors, we use different sensor measurements as thresholds to detect POIs for the two sensors. Both thresholds were set in a trial and error process.

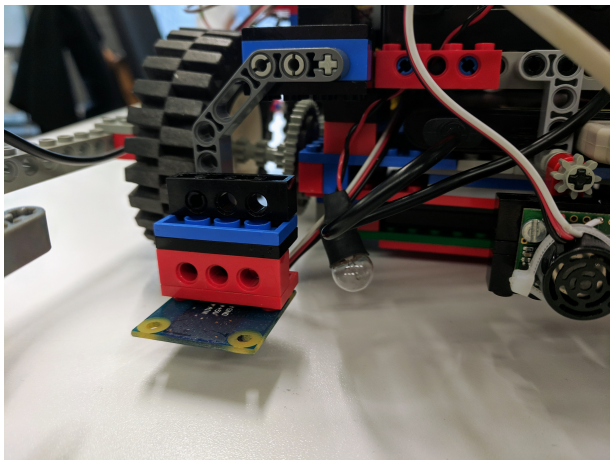


Figure 2.4: Light sensor and light bulb

2.3 Software

Besides designing and building the mechanical robot, it is also necessary to implement the complementary software to make use of all the physical components.

2.3.1 Control program

The control program, as shown in algorithm 1, is the core of the robot's software and handles the general order as well as decision making. It is therefore most descriptive of the robot's behaviour.

Algorithm 1: Control program

```
while task running do
    if POI detected then
        stop and localise;
        if POI was not found before then
            rotate and point to satellite;
        end
    end
    if hall-counter reaches localise-threshold then
        stop and localise;
    end
    if obstacle detected then
        stop and localise;
        turn away from obstacle;
    else
        if 3 POIs detected and location estimate in "go-back area" then
            go home;
            task running = False;
        else
            if waypoint reached then
                stop and localise;
                set next waypoint;
            end
            navigation go to next waypoint;
        end
    end
end
```

While it gives a concise overview of the robot's software, it also intentionally hides many details. Therefore, we will briefly describe some of the core components and decisions made of the underlying implementation.

2.3.2 Obstacle avoidance

It is to be expected that localisation in the map will be imperfect. Obstacle avoidance is fundamental for the task as it guarantees that even in these situations, the robot will avoid any collision.

In order to achieve this, we define a minimum distance of about 18cm for IR sensors and 25cm for the sonar sensor. Whenever an object below this threshold is detected, we rotate away from the respective direction while constantly checking whether one of the bumper switches is hit. If such immediate contact is detected, it is given priority and we first move away from the obstacle the robot is in contact with.

2.3.3 Localisation

The arguably most complex part of the entire implementation is the localisation. Among the many different algorithms existing in robotics for this purpose, we decided to use particle filters [Thrun, 2002] due to their continuous location estimate in contrast to grid localisation which discretises the map.

At the beginning, we initialise all particles around the known starting position and orientation of the robot. However, to distribute the particles sufficiently we add Gaussian noise with zero mean to the starting x- and y-coordinate and to the initial orientation.

At each localisation, we first update the particles orientation and position with respect to the robot movement since the last localisation, tracked by odometry, and introduce minor Gaussian noise in each update to maintain a diverse population of particles. Secondly, we compute the probability for each particle position using the expected measurement in its location, computed by a line segment intersection on the OpenCV[3] map visualisation depicted in figure 2.6, and the real measurements at the current position[4]. These are then used in a stochastic universal sampling to compute the new set of particles.

As the current location and orientation estimate of the particle filter, we compute the mean of all particle positions and orientations. While this initially seemed prone to problems whenever multiple clusters of particle would be formed, we never observed this behaviour in our experiments.

2.3.4 Waypoint navigation

To be able to reliably find all three POIs, we pre-defined a path on which the robot traverses the map. This path is represented by twenty waypoints placed so that no obstacles fall on its way. Therefore, the robot can freely move from one waypoint to the next one, only relying on obstacle avoidance to recover from potential imprecisions in the localisation. This simple navigation does not require any computationally expensive path-planning.

In order to know whether the robot already reached a waypoint, we check if it is in (at most) 10cm distance to it and behave accordingly.

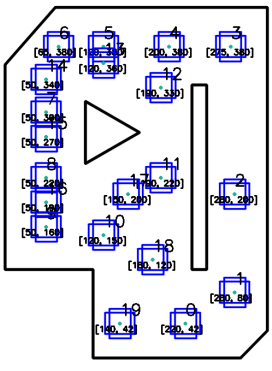


Figure 2.5: Waypoint mission visualisation

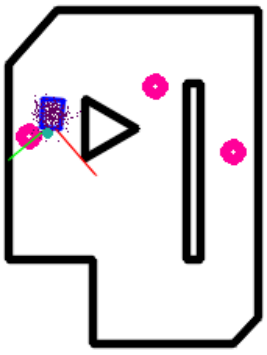


Figure 2.6: Localisation visualisation with found POIs, robot location and particles marked

[3] `https://opencv.org/`

[4] We only consider IR measurements for this localisation step due to the sonar sensor's large variance.

Chapter 3

Results

3.1 Robot performance

Overall, our robot was mostly successful in achieving the task. It was repeatedly able to detect all three POIs distributed in varying locations across the map and succeeded in returning to the original location. However, our design with respect to gearing as well as our frequent localisations after travelling a certain distance meant that our movement overall was too slow in performing the entirety of the task in the given 5min time limitation. It mostly takes the robot between 5 : 30 and 7 minutes to finish the task depending on the exact POI locations and the consistency of the localisation.

In a brief evaluation, we found that in 9 out of 11 executions of the task, the robot was capable of finding all POIs and return home, even though it took too long. However, in the remaining two runs the robot got lost due to imprecise localisation after finding the second POI. This resulted in the robot getting stuck in a corner unable to recover its orientation and position.

3.2 Challenges

3.2.1 Localisation divergence

As described above, despite the overall reliable performance of the robot, there are runs where the localisation eventually diverges from the real position. This usually happens due to an incorrect belief of orientation and can be corrected when a reliable localisation is done shortly after. However, when the robot in this situation moves in the open, where no reliable sensor measurement is available, we rely solely on precision of the localisation as well as our odometry. In this situation we noticed that our particles were not sufficiently distributing and "overconfident" in their imprecise belief leading to a mistake in localisation, which the robot usually can not recover from.

Due to a comparably large amount of obstacles in the arena, the robot has reliable measurements in most positions and is able to localise properly, but inaccuracies in robot movement can still cause the problem in some runs. This is known to be one of the core challenges whenever using particle filter localisation and remains a problem in our case.

3.2.2 Hall effect sensor variance

Another known problem is the configuration of the hall effect sensor. The sensor includes multiple magnets and each time a magnet passes the bottom of the sensor on a rotation, the binary value is flipped. Therefore, the number of such magnets as well as their position is crucial for the sensor precision.

The hall effect sensor provided contains two magnets. However, these are not, as expected, placed opposite to each other so that each 180° rotation of the sensor axis would lead to a change of value, but in a 90° angle to each other. This means that one hall-count effectively stays the same for three times the duration as the other one making any hall-count-based prediction, such as distance travelled or rotations, "jump". Such an inconsistency can harm the precision of single rotations or measurements with respect to odometry considerably and we were unable to reliably remove this irregularity.

Chapter 4

Discussion

We designed and constructed a LEGO robot capable of reliably localising itself in a known environment. It is able to follow a pre-determined path while avoiding any obstacles and detecting points of interest in the form of reflective tapes on the floor.

The robot successfully navigated around the arena, finding all three POIs and returned to its deployment position in most cases. However, our robot design led to an insufficient speed to finish the task in the given 5 minutes constraint.

For future improvements, one could implement methods to increase the chance of the particle filter localisation to recover from imprecise estimations. One such method would be to introduce random particles in a large area surrounding possible locations as described by Thrun et al. [Thrun et al., 2005][1]. Additionally, the hall sensor imprecision could be dealt with by comparing the number of iterations a binary value remains the same to analyse whether the current hall-count is a long or short one. Both these methods could possibly improve the consistency of our original methods and allow for less localising steps to reduce stopping times and let the robot perform the task at a faster speed.

[1] See section 8.3.3 Random Particle MCL: Recovery from Failures

Bibliography

[Thrun, 2002] Thrun, S. (2002). Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Appendix A

Mechanical design

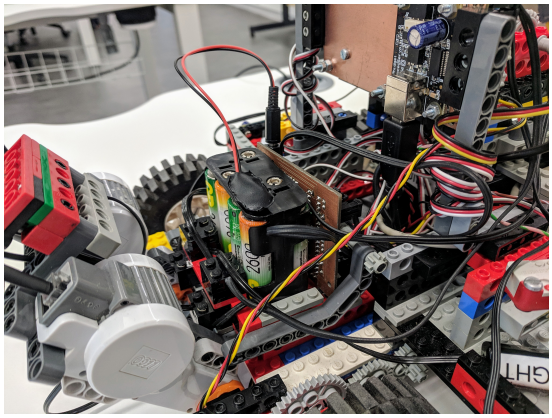


Figure A.1: Motor board and AA battery pack

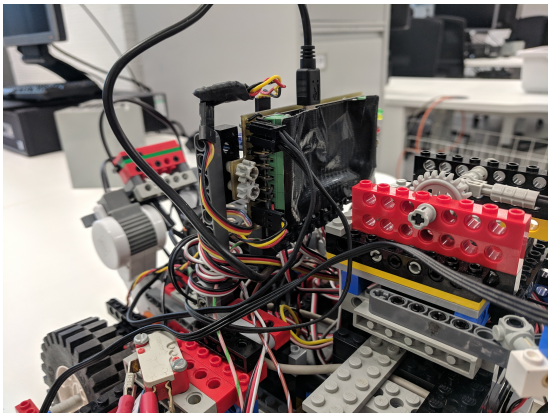


Figure A.2: Phidget 888 interface board mount

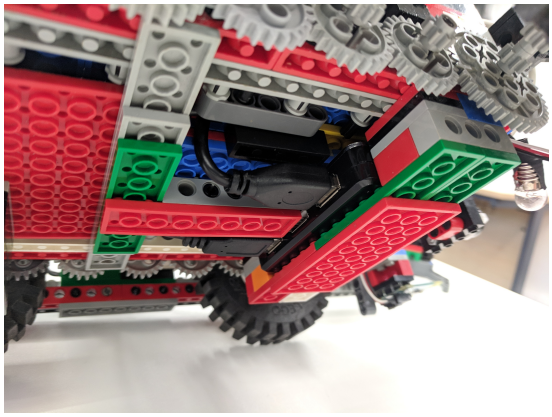


Figure A.3: Backside of USB hub mount on the robot bottom

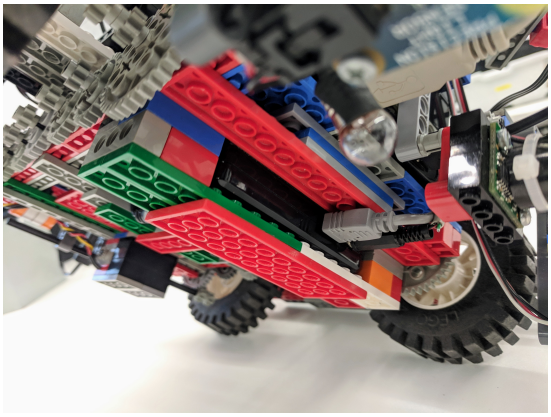


Figure A.4: Frontside of USB hub mount on the robot bottom

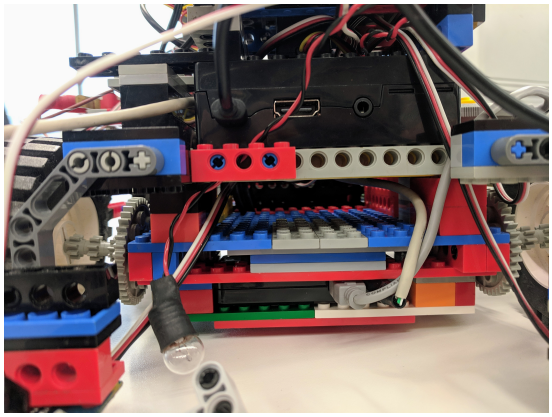


Figure A.5: Powerbank slider under the Raspberry Pi

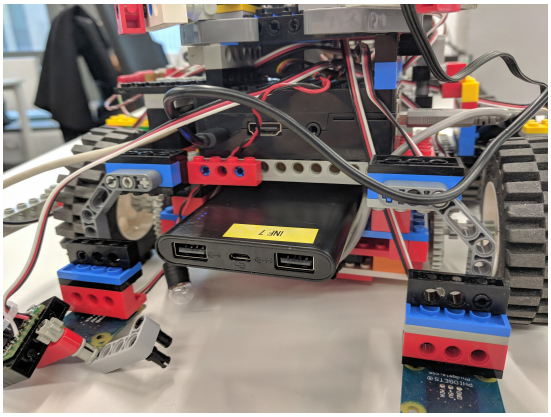


Figure A.6: Powerbank slider with powerbank

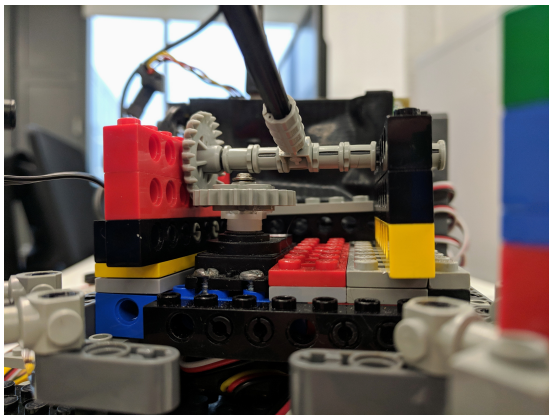


Figure A.7: Closeup of the servo motor and its antenna gearing connection

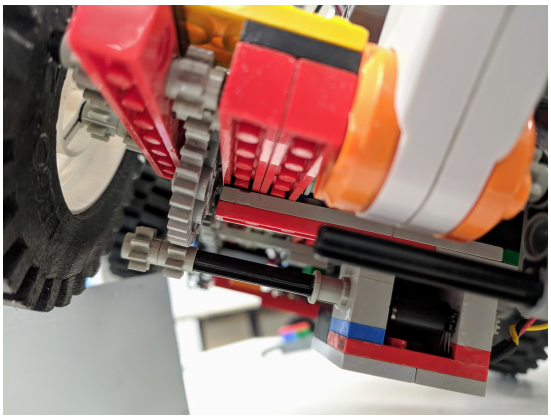


Figure A.8: Hall sensor with axis connected to gear-chain

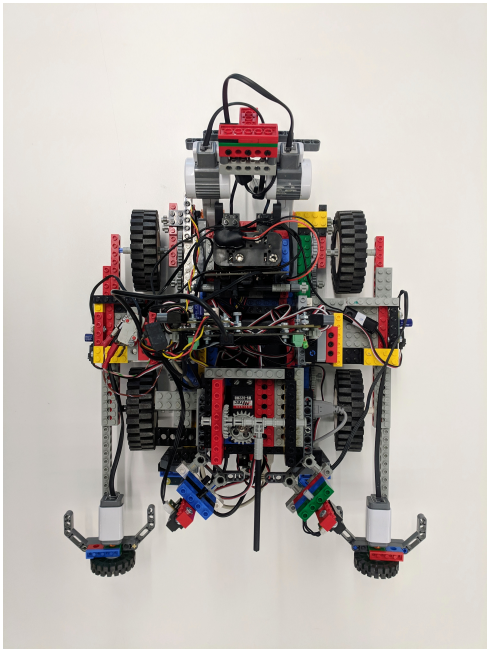


Figure A.9: Top view of the entire robot

Appendix B

Measurement models

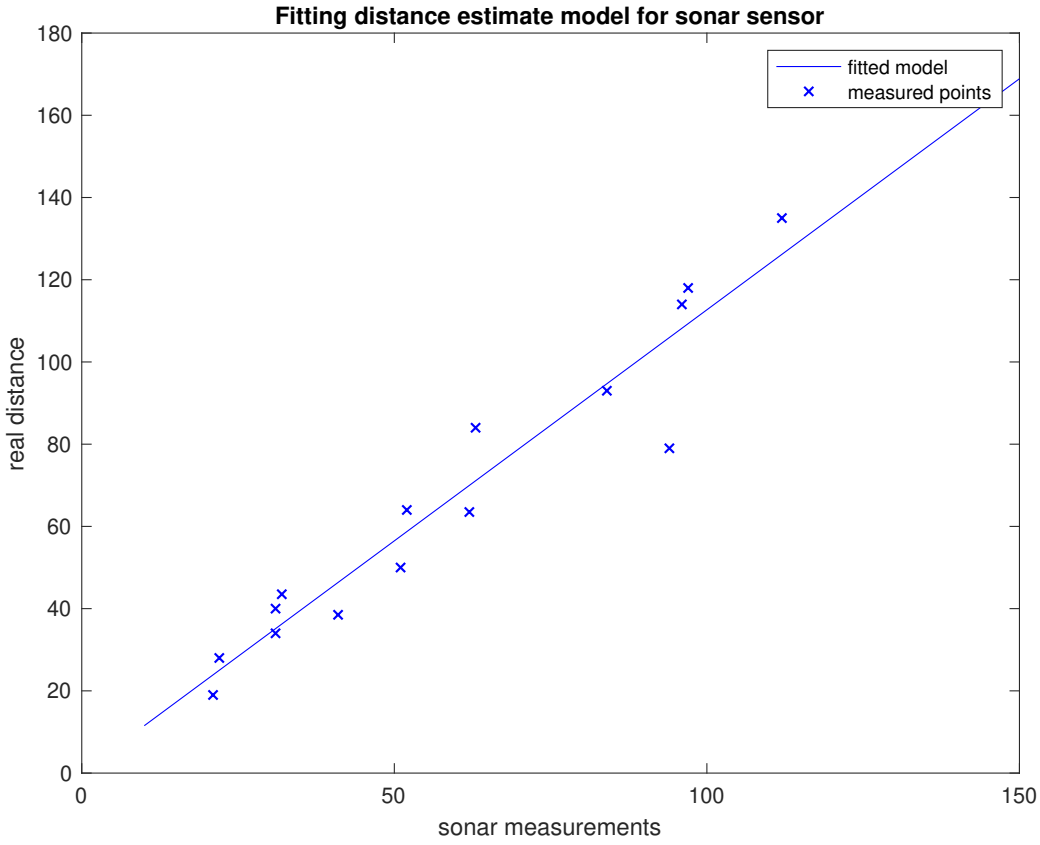


Figure B.1: Illustration of fitted model to sonar measurement data

real distance (cm)	19	28	34	38.5	40	43.5	50	63.5	64	79	84	93	114	118	135
Sonar measurement	21	22	31	41	31	32	51	62	52	94	63	84	96	97	112

Table B.1: Measurement data table for sonar distance model

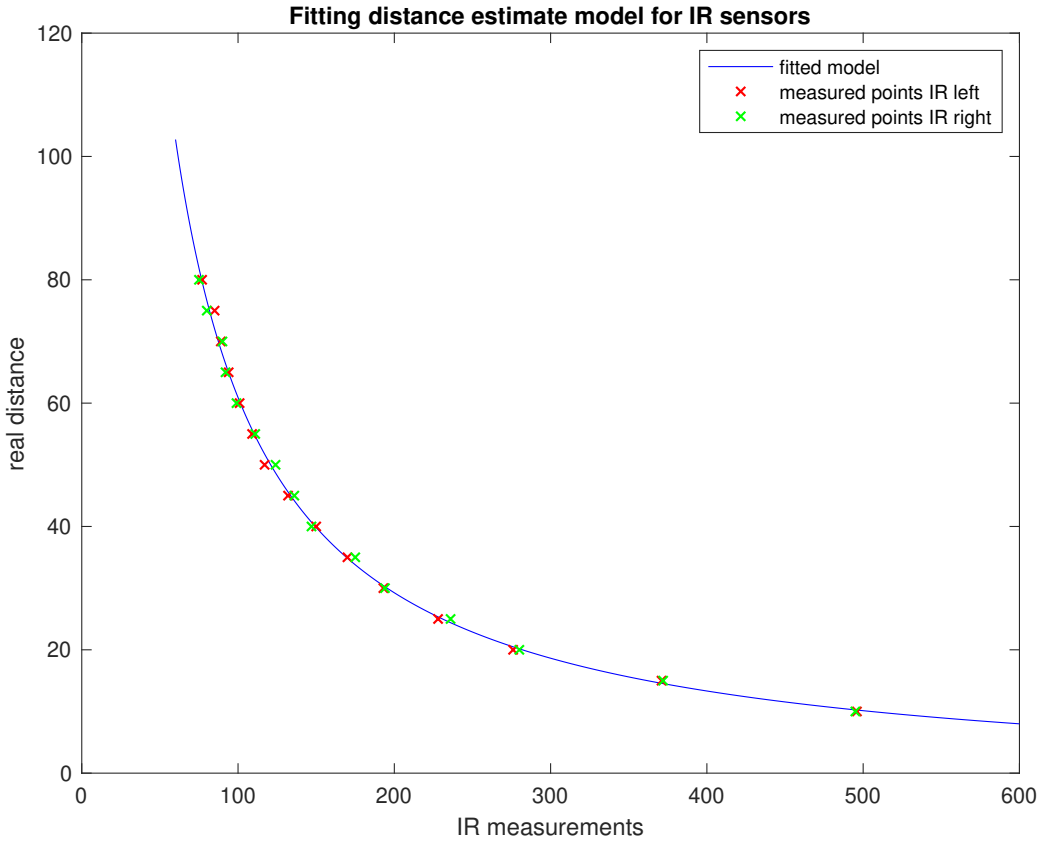


Figure B.2: Illustration of fitted model to IR measurement data

real distance (cm)	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
IR left measurement (V)	496	371	276	228	193	170	150	132	117	109	101	94	89	85	77
IR right measurement (V)	495	372	280	236	194	175	147	136	124	111	99	92	90	80	75

Table B.2: Measurement data table for IR distance model

Appendix C

Additional Sources

- The Art of LEGO Design:
  https://www.cs.tufts.edu/comp/150IR/artoflego.pdf
- Lego Design
  https://www.clear.rice.edu/elec201/Book/legos.html
- LEGO Mindstorms RIS 2.0: Gears, Pulleys, Wheels, and Tires
  http://www.ecst.csuchico.edu/~bjuliano/csci224/Slides/03%20-%20Gears%20Pulleys%20Wheels%20Tires.pdf
- LEGO Gears Tutorial
  http://sariel.pl/2009/09/gears-tutorial/
- SHARP GP2D12 information sheet
  https://engineering.purdue.edu/ME588/SpecSheets/sharp_gp2d12.pdf