Intelligent Autonomous Robotics Final Report

Sean Wilson and Daniel Wren
School of Informatics, University of Edinburgh
s0831408@sms.ed.ac.uk, s0789266@sms.ed.ac.uk

1. Abstract

The goal of this assignment was to design an autonomous robot capable of using different sensors to navigate an open world, in order to locate and distinguish between five resource sites. The robot was designed for accuracy and flexibility. From observing biological designs and behaviours, the robot was designed with a ‘head’ of primary sensors used for localization and mapping, while other sensors such as touch and light were used to detect changes in the environment. Despite the promising results observed from mapping and docking, the robot suffered from poor localization, the accuracy of which depended on the battery’s voltage level. This could be rectified with the addition of Bayesian probabilities for each particle observed.

2. Introduction

The task was to build an autonomous robot able to locate resource sites, press the light switch within the site, recognise the frequency, and execute a movement sequence depending on the frequency emitted. The robot must visit as many sites as possible in five minutes. The landmarks within the world are static, but the robot will navigate an open world and must therefore react to moving obstacles such as other robots.

To achieve these objectives, the robot must sense the environment, determine its location and plan its route to the next resource site. This is achieved through the use of a Fit-PC, running a Linux OS, and a Phidget interface kit. The interface kit allows for several types of analog and digital inputs, useful for creating an internal image of the world. For the robot to be completely autonomous, it will create an internal map from sensory data and determine goals through observations.

3. Methods

3.1 Robot Design

3.1.1 Locomotion

A four-wheel drive system kept at least two drive wheels on the ground and provided the ability to turn on the spot. Turning on the spot decreased the robot’s turning circle, decreasing the likelihood of colliding with unobserved objects. To do this with minimal tire slippage and opposing turning forces, the wheels were spread evenly on a circle, with the centre of the robot equidistant from each wheel. This made the robot square in shape, which further reduced the risk of collision while turning on the spot and helped to distribute weight.

The robot weighs about 2.5kg and is driven by two small motors. As such, it is imperative that the motor to wheel revolution ratio is decreased as speed is inversely proportional to torque. A 26:1 driver to driven ratio gives the robot sufficient torque and momentum.

Centre of mass and gravity are key. The weight must be evenly distributed, especially on a four-wheel drive vehicle as all wheels must remain on the ground to ensure grip. Weight was evenly distributed by locating the battery between the front wheels, while the Fit-PC and circuitry was placed between the rear wheels. The robot was designed with as low a centre of gravity as possible.

3.2 Sensory Positioning

Preliminary research into the accuracy and behaviour of sensors revealed that the environment is best observed through a hierarchy of sensory inputs. From observing biological life forms, it is apparent that in order to observe an environment effectively and naturally, the sensors must have a certain degree of freedom. This allows the robot to map and localize without changing direction, and reduces localization error.

As such, the IR and sonar sensors were attached to a servo motor at the front of the robot. The equipped servo allows a 50° turn, accurate to 1° in both directions.

At the top of the sensory hierarchy, the most frequently used, is infra-red (IR). The IR sensor has an accuracy between 8cm and 200cm (see Appendix G). One IR sensor faced horizontally forward, complimenting the sonar readings and recognising the shapes of sites. The second IR sensor, placed on a 45° tilt, was used to detect tall, close objects such as walls.

The sonar sensor was inaccurate at providing fine detail such as sloping walls. The wall returned the same or similar values (within 10 units) for each reading. Sonar also picked up noise from surrounding objects, even during simple and controlled preliminary testing. Due to the nature of the environment, the sonar could only be used to roughly locate sites at long range (over 200cm). The sonar had an accuracy of 10cm either side of the true distance.

At the bottom of the hierarchy is the most reliable but least utilised: digital switches used as bumper sensors. Being digital (on/off), they only trigger when pressed, making them highly reliable. The sensors were situated in the four corners of the robot, attached to a front and rear bumper spanning the width of the robot.

Light sensors were required to identify the frequency of LEDs at resource points. To ensure a clear view, two light sensors were positioned at the same height as the LEDs. The light sensors were originally placed at the front of the robot (see Appendix C), but this was too far forward and did not detect light changes while close to the button. Consequently, the sensors were moved to a central position on top of the robot.

In order for the robot to identify when it is situated within a resource site, two underside light sensors were placed on the front corners of the robot. This enabled the robot to detect a difference between the black site floor and the grey open world.

3.3 Movement

When turning, the function ‘turnToFace’ will be called taking an integer: the number of degrees to turn. This angle will be a local bearing with regards to the robot’s current direction. This will be converted into a true bearing with regards to the mapped world (see localization and mapping). With this function, it is easy to keep the robot’s bearing up to date in relation to the internal map. The robot takes 4.85 seconds to turn 360° on a full battery. As the battery’s voltage decreases, the turn becomes less accurate. Due to the nature of voltage driven motors, this was unavoidable.

3.4 Object Avoidance

Object avoidance was broken down into two categories:

1) Object detected within the area of a potential site,
2) Object detected outside potential site range.

If an object is detected within a set radius of its goal, the robot will perform a sweep in search of a V-shaped or inverted V-shaped object. If a V-shape is detected close-by, it can be assumed that the goal coordinate was not located at the centre of the resource site. The goal coordinate is updated to become the centre of the newly detected V-shape, and the robot continues its search. If an inverted V-shape is detected, it can be assumed that the robot has arrived at the rear of a resource site and so the robot will manoeuvre around to the front of the resource site and continue its search.

Outside of the range the robot will turn parallel to the object, rotate the IR sensors to face the object, and manoeuvre until it is clear. At this point the robot will continue pursuing its current goal.

If the angled IR detects close readings above the centre of the V-shape this is identified as the walled edges of the world and the goal coordinates are recorded as not a resource site. If the robot repeats the same manoeuvre loop five times, the particle is no longer considered a potential site.

3.5 Site Location and Behaviour

3.5.1 Site Identification

Downward-facing light sensors allow areas to be identified by the floor colour. Preliminary testing confirmed that a single central sensor did not provide accurate readings and did not detect if only one corner of the robot drove over the resource site.

Two light sensors (one on each front corner of the robot) provided improved readings and recognised only one corner of the robot occupying the resource site. A bulb was placed on the underside of the robot, equidistant from each light sensor (Appendix B). The bulb nullified the effect that natural shadow, cast by the robot, had on the readings. While the robot is in a resource site, readings for both sensors are recorded and are used for the threshold between open space and resource site. By setting the threshold each time, it was possible to eliminate the effects of bulb intensity, and natural varying light conditions on the robot's ability to identify the resource point.

3.5.2 Locating the light switch

Once within the resource site, the tilted IR, aided by the servo motor, takes 20 readings in a 100° arc. The IR sensor is reclined by 45° (Appendix D) in order to locate the gap between the two resource point walls, directly above the light switch (see appendix E). With the 20 readings from the IR sweep stored in an array, the robot proceeds to analyse the array in order to calculate the correct direction of the light switch. The algorithm pseudocode for this calculation is described in appendix J.

The algorithm stops if a gap is bounded by walls, as this is taken to be the location of the switch. If a gap is not bounded on both sides, it is assumed to be the sides of the resource site. In this case, the algorithm continues until a bounded gap is located or all readings are checked. If two gaps are located, both unbounded, the robot will turn to face the gap with the closest forward IR reading. Once the gap location is found, the turnToFace function rotates the robot towards the gap.

3.5.3 Recognising a switch press

As the robot navigates resource points, a thread is listening for activation of the front touch sensors. Once one of the touch sensors is triggered, the robot stops and waits for a large reading from a front light sensor. If no reading is detected, the robot assumes it has collided with the wall instead of the switch. In this case the robot will reverse and adjust for a better angle of approach.

3.5.4 Recording the frequency

Once these preconditions have been satisfied, the robot stops its motors and spends six seconds recording the number of flashes emitted every two seconds. Preliminary tests showed that the forward-facing light sensors gave different readings, so recordings for each sensor were taken separately. Two light sensors were used in order to apply sensory fashion, taking the readings from the sensor with the lowest variance and recording the mean from each two second period taken to be the frequency.

3.5.5 Executing the correct movement sequence

With the frequency, the correct motor controls are executed according to the task specifications. The robot's four-wheel drive allows it to rotate on the spot with precise executions of the required movements that show which frequency the robot's sensors read.

3.6 Localization and mapping

Although the resource sites were fixed, it was less computationally complex to treat the world as non-static and thus create a map through sensory readings. The most suitable method was particle filter SLAM (Simultaneous Localization And Mapping).

The robot's location is the origin (0,0); when the robot moves, each saved particle is translated with respect to the robot's movement. Each time the coordinates of the particles are updated, the error increases.

When the robot starts, the direction it faces becomes the true 0/360 bearing of the world; all calculations are based upon this. When the robot rotates, its bearing is updated by using trigonometry. A basic naive implementation was used with the pseudocode below in order to update and correct the centre particle of resource points:

```
1) For new possible resource site particle
2)      Compare against all previously saved
        particles
3)      If new particle is within set radius of
        saved particle
4)              Declare as the same possible site
5)              Calculate the average between old
                and new and save coordinate
6)      Else
7)              Save particle as a new possible
                site
```

Each particle is calculated by what is observable from the current location as such the centre will be different, depending on the observable portion of the site. With every added particle, the centre of each site becomes more accurate through convergence. Later, the Bayesian probability between estimated and actual location should be implemented to reduce the error given by the inaccuracy of moving and rotation.

3.7 Site location and information

At the start of a run, the robot performs a 360° sweep, using sonar and forward IR to record all potential resource sites (see Appendix F). Using raw IR readings would not allow for accurate location of sites. It was found through several readings (see appendix H) that the equation of our IR readings to get a linear distance was:

```
y=(0.0001*(50000*(x^(100/61))
+1.38199*(10^9)))/
(x^(100/61))
(Here y is the IR reading and x is the
distance in cm.)
```

A potential resource site is defined as three or more consecutive distance readings (up to 20 readings) within 10cm of each other at a range less than 200cm. Note that one reading spans 5°. This makes the maximum size of a potential site 60cm (at a distance of 200cm), which is the width of the smallest site. 200cm was decided as the threshold as the IR readings become exponentially more inaccurate (see Appendix H).

When a potential site is located through a sweep, the site location is recorded in a 2D array as an x and y coordinate in relation to the robot's location and the true bearing of the world. The angle between the robot's bearing and the location is recorded alongside the distance from robot to location and the site's state. The state of a potential site is regarded as either:

- 0 - The potential site has not been visited
- 1 - The site has been visited and a frequency recorded
- 2 - The location has been visited and was not a resource site

When a site has been visited, the state of the coordinates are updated. When a current goal's state is no longer 0, the robot will do a 180° turn, facing the opposite direction of the coordinate and perform a 270° sweep, updating the world map with new particles. The closest unvisited potential is then set as the new current goal.

4. Results

4.1 Movement

The remaining battery greatly affected the accuracy of movement. The function itself was successful in keeping the robot's bearing within the world accurate, but if the battery started to falter then the robot would not complete its full rotation. This was only a serious issue when moving long distances where a small angle could make a large difference or when multiple bad rotations would stack up and heavily skew the robot's bearing within the world.

4.2 Object avoidance

This function is heavily dependent on the coordinate system to maximise efficiency. If the robot predicts its location further away from the goal coordinates than is true, the robot treats site walls as obstacles and thus does not search for V-shapes. This issue arose more often, the further the robot had travelled and rotated.

4.3 Site location and behaviour

The following statistics are based on 20 attempts.

Locating the bearing of at least two sites from the 360° sweep had a 90% success rate. In some cases. the robot perceived the back of a resource site as its front. In this scenario, the robot could manoeuvre to the front of the site and begin docking as normal.

Once within a site, locating the gap above the switch resulted in almost 100% success. This would take a variable amount of time, but the robot would reach the switch eventually. The delay was due to the width of the robot sometimes impeding its advance and a bumper detecting a wall.

Recognising the frequency had an 84% success rate and, assuming the frequency was correctly identified and the battery was adequately charged, executing the correct movement sequence was always accurate. Through previous calculations, the time required to turn 1° was determined and accurate 180° or 360° turns were possible.

Sometimes the robot pressed the switch but due to the battery in the switch dwindling, the robot's light sensors would not pick up the flashes and it would perceive the switch as a wall.

4.4 Site location, localization and mapping

The mapping system correctly identified three resource site particles with a 65% success rate over 20 test runs. Particle matching in from further sensory data was unsuccessful due to large errors between estimated and actual localization when taking readings. 5% of the time the robot would identify a visited resource point as a new potential.

5. Discussion

It was evident that in order to increase the accuracy of the robot’s localization and mapping, Bayesian probability would need to be implemented to correct the difference in estimated location and actual location.

Due to inconsistent motors, the robot would sometimes fail to complete a 360° turn, or continue turning after 360°. This could be corrected by gearing up the servo motor to turn more than the standard 110°. If the servo motor was able to turn a full 360°, the robot would be able to map its surroundings with increased accuracy and speed.

The mapping system was effective, but as can be seen from the readings in Appendix H, accuracy drops exponentially at distances greater than 200cm. Sonar could be used to ensure that sites at further distances are located but as seen in Appendix I, the sonar tends to pick up noisy readings frequently. Although sonar may not give accurate readings, potentials could be recorded and further analysed by the IR sensors when closer.

On the day of the competition, a major flaw in the robot’s programming emerged. Throughout testing, this issue did not arise and therefore was not corrected. The problem lay in the way the threshold between the resources dark floor and open floor was created. During the competition, when disconnected from the visual output, the threshold readings were very dark (a very low value). Thus, when the robot entered any dock, including the dock the threshold was set in, the underside light readings did not fall below the threshold. On reflection, a more robust method would be to measure the drop in light readings when entering a zone. For example, if the light readings suddenly dropped by 20 and stayed roughly 20 below normal, this would indicate a change in surface.

Once the robot identified that it was situated within a dock, the corresponding function and tilted IR guided the robot towards the light sensor with satisfactory results. The main issue was the width of the robot. For the 90° docks, the robot would sometimes catch the edges of the bumpers on the wall before the light switch. This was rectified by the robot repositioning itself in front of the light switch for another attempt. By streamlining the design, it could be possible to touch the light sensor on the first attempt every time.

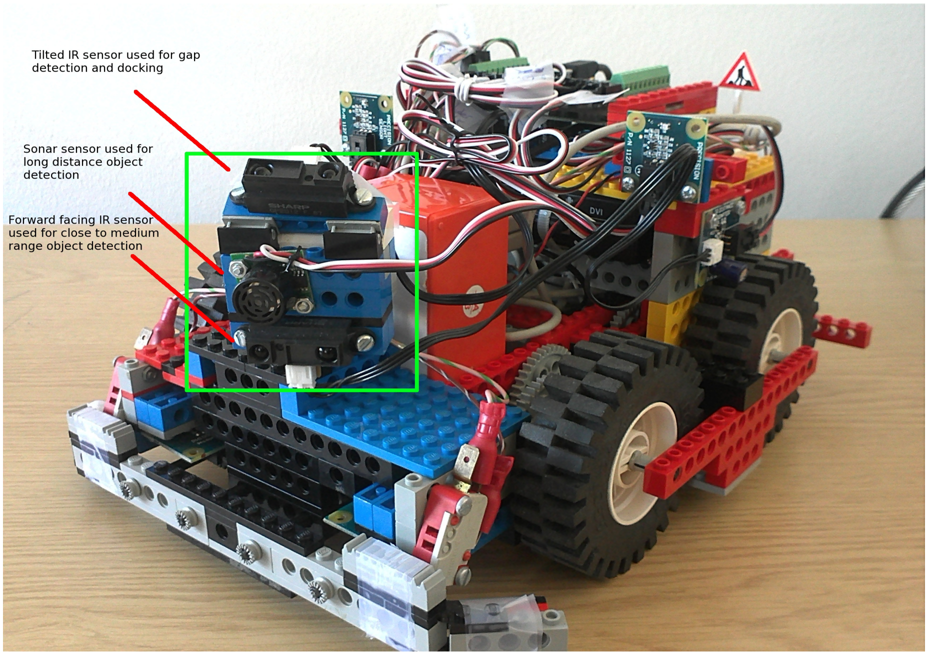
6. Conclusion

Conclusively, the robot completed its tasks adequately through effective searching, sensing and docking behaviour. The localization and mapping system, though not perfect, allowed for an internal map which enabled the robot to plan its route from the beginning, evolving that plan as new data was processed and new potential resource sites were visited. Through all test runs, the robot’s record within the designated time frame was 4 resource sites visited with 3 being correctly recognised.

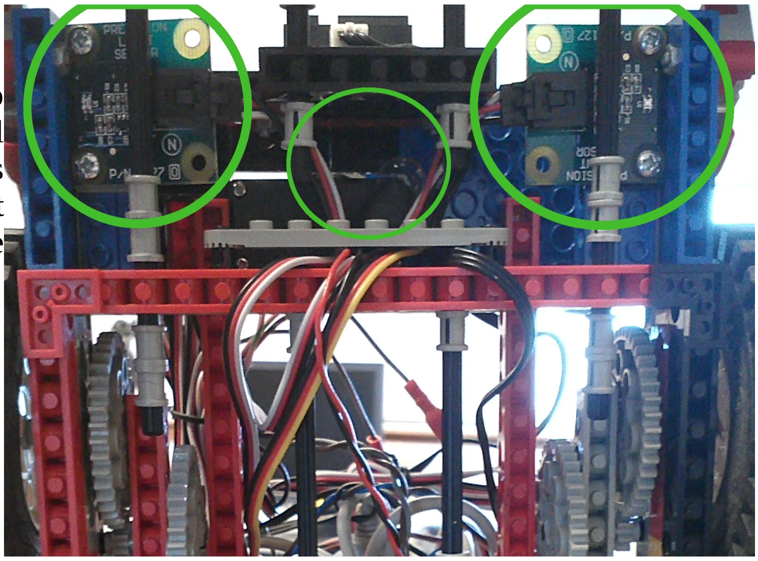
7. Appendix

Appendix A

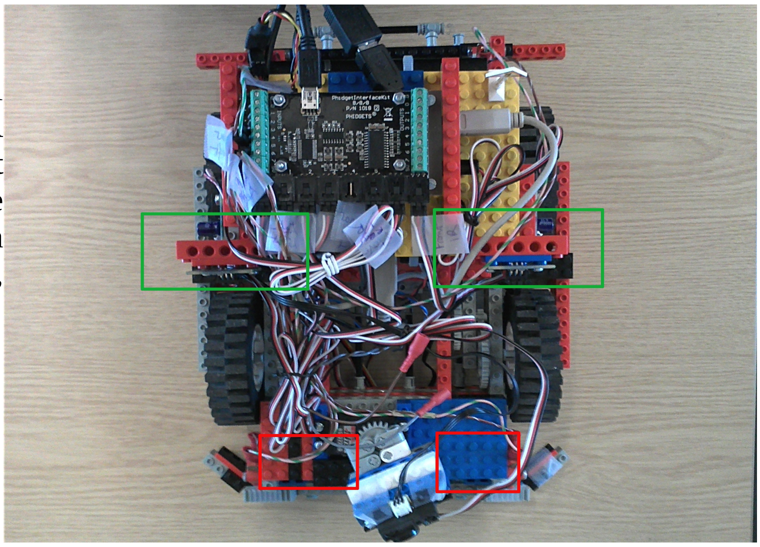
Appendix A shows the sensor ‘head’ attached to the servo motor at the front of the robot



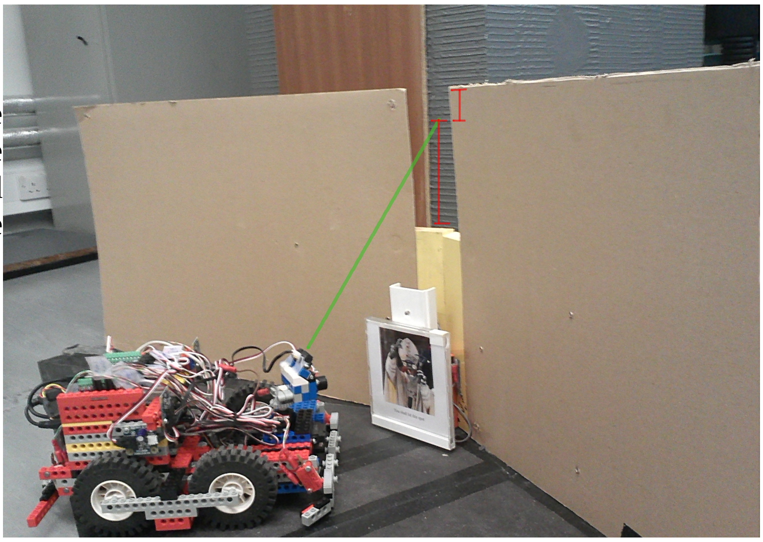
Appendix B
Circled in this image are the two underside light sensors and the central bulb. The light sensors are located as far to the sides as possible to ensure that the smallest overlay into the dark zone can be detected.



Appendix C
The red squares indicate the original position of the forward facing IR sensors. Through preliminary testing it was found that more accurate frequencies were captured when relocated to the secondary position, highlighted in green.

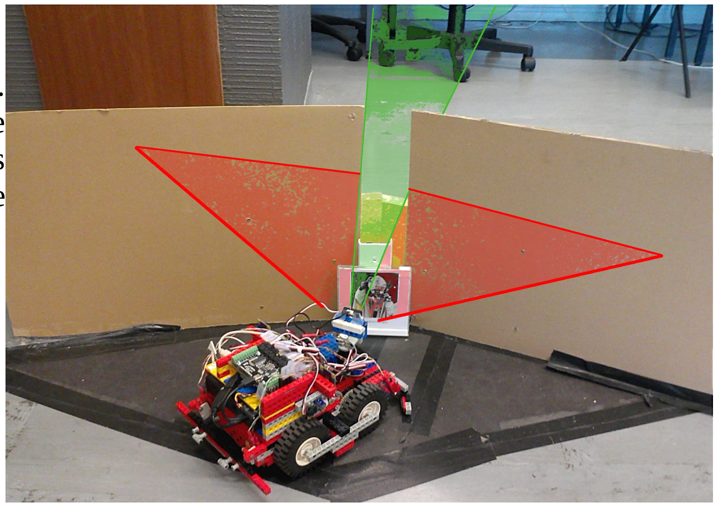


Appendix D
The 45 degree IR sensor is shown here detecting the gap between a resource site's walls. A 45 degree tilt was found to give enough clearance above the light switch, even at close range.



Appendix E

This image depicts a typical IR sweep. Wall-like readings are detected for the majority of the sweep. A bounded gap has been highlighted in green. This gap will be taken as the correct direction to pursue.



Appendix F

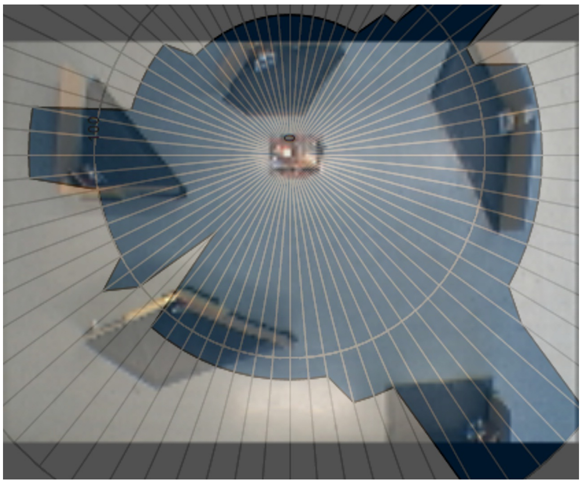


Figure 1

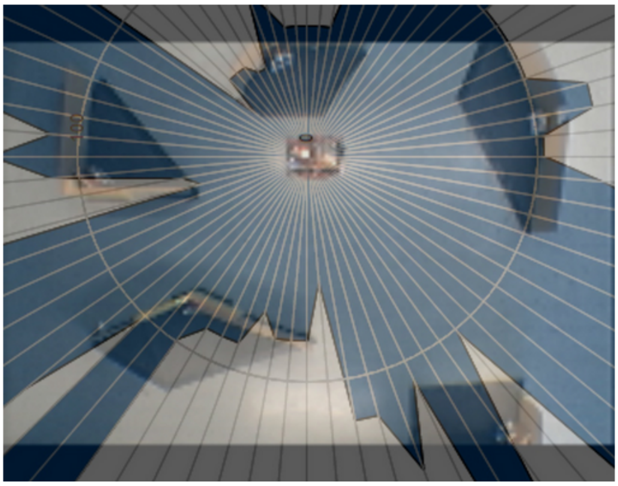


Figure 2

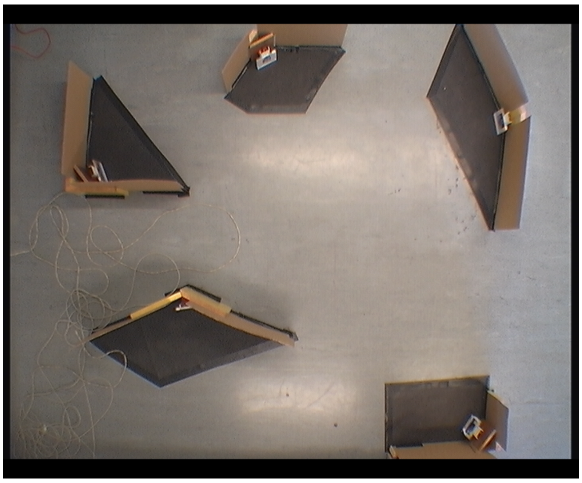


Figure 3

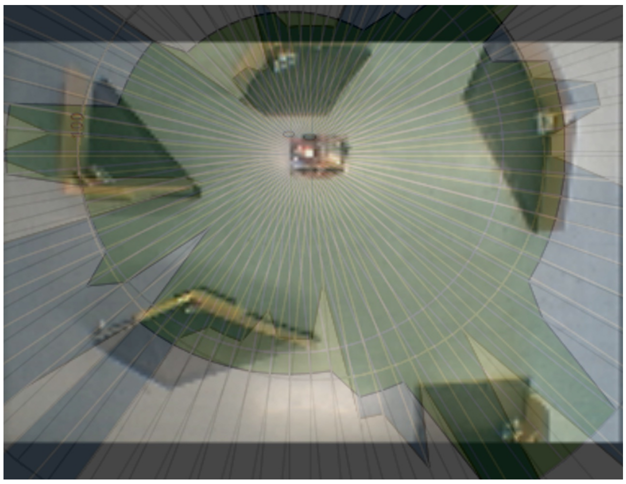
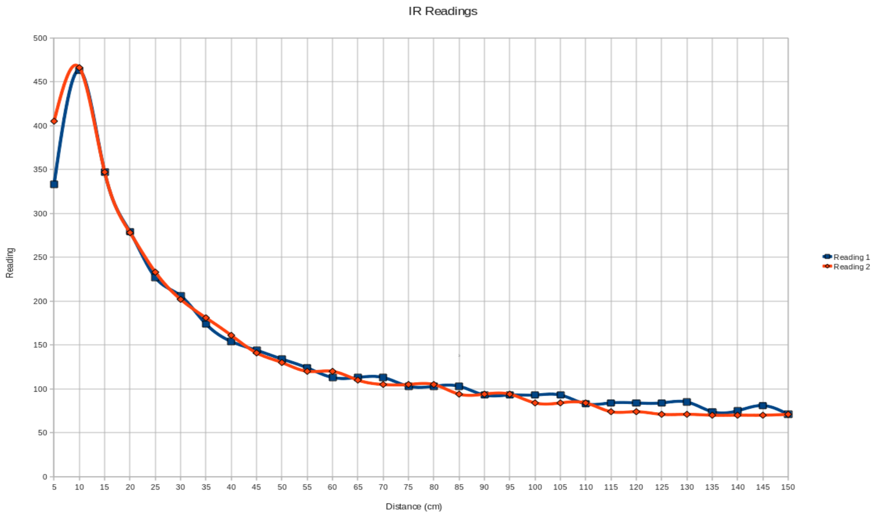


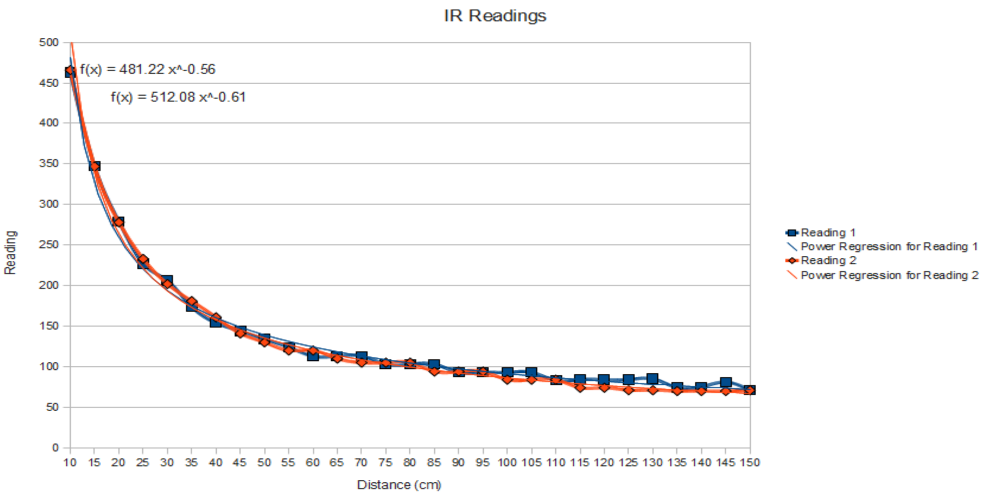
Figure 4

Figure 1 – This shows an example of mapping via sonar. The blue area are the readings recorded by the sonar. As can be seen, the sonar does not give very accurate distance measurements.
Figure 2 – This shows an example of mapping via IR. As can be seen, the IR gives much more accurate distance measurements but after a range of 100cm, the accuracy starts to decrease.
Figure 3 – This shows the world with no robot.
Figure 4 – This shows an overlay of both IR (blue) and sonar (green) mapping.

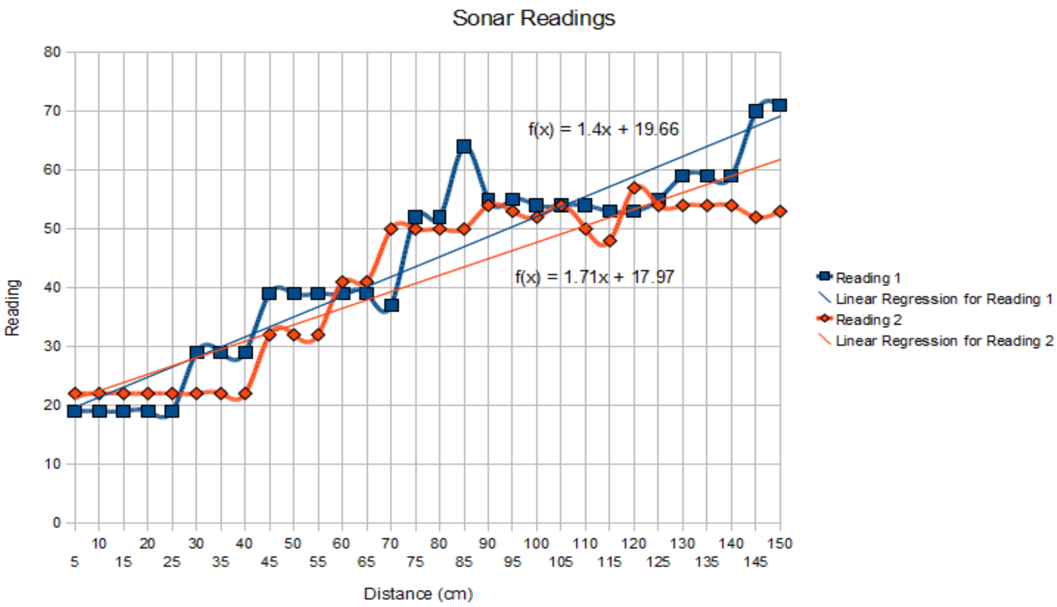
Appendix G
This graph shows the inaccuracy of the IR sensor at a range of 8cm or less. Two test were carried out, taking readings every 5cm between 0cm and 150cm.



Appendix H
Here is a modified version of appendix G with the inaccurate readings removed. This was done in order to obtain the power regression curves needed to convert IR readings to a distance metric.



Appendix I
This figure shows the readings recorded by the sonar sensor over two tests, recording every 5cm. As can be seen by readings at 80cm and 110cm, the sonar often picks up stray readings, thus giving inaccurate linear equations.



Appendix J

```
variable threshold //distinction between wall and open space

For each 5 degree reading
      If reading is a wall
            Increase wallCount

            If wall is after a gap
                  gapLocation = midpoint of gap in degrees

                  If gap is at the far left of sensing range
                        reset the gapSize to 0

                  Else if wall on either side of gap
                        goto turning method

      If open space detected
            Increase gapSize

            If gap is on far right (last reading)
                  gapLocation = midpoint of gap in degrees

      If no walls detected (wallCount == 0)
            proceed closer to the resource point

      If no gap detected (wallCount == 20)
            reverse back from the wall
```