**Section 1: Robot Design**

The robot adopts a two-wheel differential drive configuration, having two drive wheels under independent control, one at either side of the robot. The ball component of a LEGO ball and socket joint implements a simple, low-friction castor at aft of the robot.

The ultrasound sensor is mounted on a rotating turret atop the robot at its 2D centre of mass. Consequently, all ultrasound measurements are taken relative to the centre of the robot. The controller forms the body of the robot, with the batteries being accessible from the underside for ease of replacement.

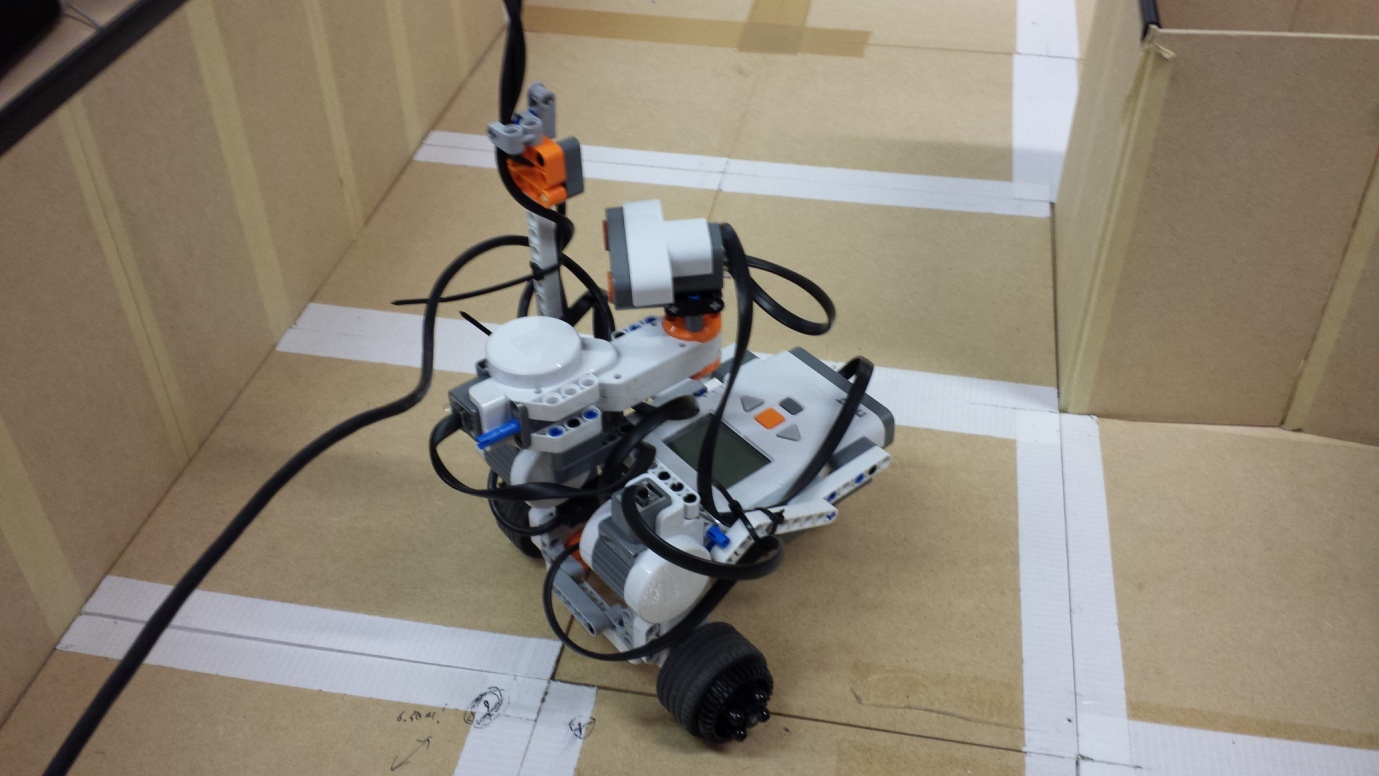


Figure 1: The Robot

The RWTH Mindstorms NXT Toolbox developed by Aarchen University is used to communicate with the robot. Calls to the Toolbox are encapsulated in a class named Bot which provides high-level functionality equivalent to the BotSim class, for translating and rotating the robot and performing an ultrasound scan. The Bot class also exposes low-level functionality for rotating the ultrasound sensor by a specified angle, obtaining ultrasound measurements at the current angle, and generating an audible tone.

**Move**

The move function translates the robot by the specified distance in centimetres. It works by computing the tacho limit or number of rotations which should be applied to each motor in the same direction. he tacho limit is calculated by dividing the desired translation distance by the wheel circumference, which was obtained by measuring the wheel diameter using a micrometer, then multiplying by π.

**Turn**

The turn function rotates the robot by the specified angle in radians. It works by computing the tacho limit which should be applied to each motor in opposite directions (contra-rotation). The tacho limit is calculated by multipling the ratio between the wheel radius and the robot radius by the desired angle of rotation. The robot radius was taken to be the distance between the centre-points of the two wheels and was measured using a 30cm rule.

**Ultrascan**

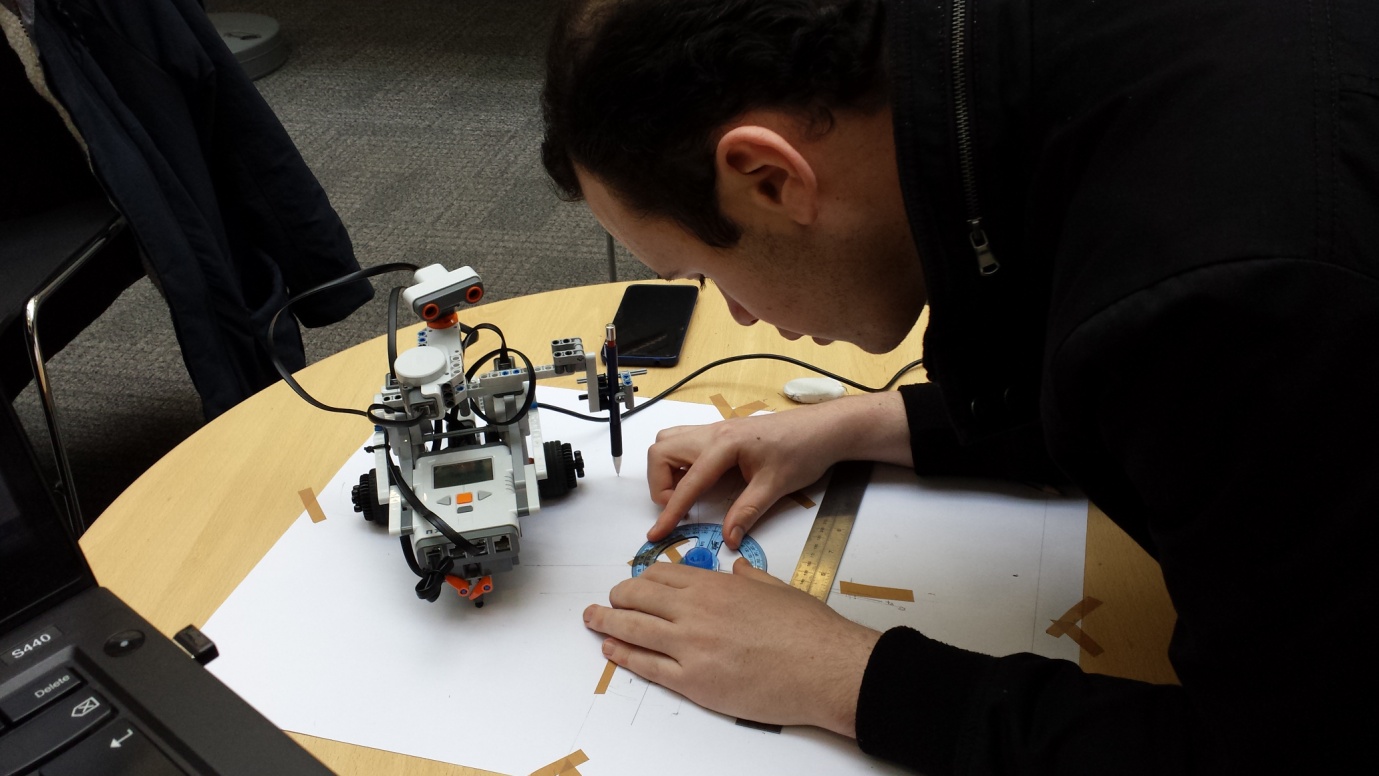
The ultrascan function rotates the ultrasound sensor in increments of the specified angle in degrees over a 360° range. To prevent the sensor from becoming entangled in its own cable, it is first rotated by 180° clockwise, then anticlockwise in increments of the specified angle, and finally back to 0°. At each increment, the sensor is interrogated for the distance to the nearest obstacle. Once the scan is completed, adjustments are made to ensure that the distance vector matches that returned by BotSim i.e. in respect of the positions of the elements for each angle, thus ensuring interoperability with BotSim for the purpose of comparison with the particle scans.

**Section 2: Calibration**

A calibration procedure was undertaken to quantify the robot’s rotational, translational and sensor error. Calibration was conducted in an area outside the computer labs over the course of several days.

**Rotation**

Rotation calibration was performed on a coffee table inside the Merchant Venturers Building. The robot was placed on a sheet of paper, and a penholder was constructed from LEGO and attached to the robot, so that it would trace its own path as it moved. The robot was then commanded to rotate by an angle in the range ±270°, and the actual angle of rotation was measured using a protractor. Five measurements were taken at each angle.

  
Figure 2: Robot Calibration in the Merchant Venturers Building

The results of rotation calibration are summarised below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Actual Angle (°)** | **-270** | **-180** | **-90** | **-45** | **45** | **90** | **180** | **270** |
| **Measured Angle (°, Mean)** | -258.9 | -172 | -84 | -38.8 | 42.4 | 74.6 | 156.9 | 250.8 |
| **Measured Angle (°, Std Dev)** | 0.7 | 1.8 | 1.2 | 1.7 | 1.1 | 3.6 | 5.6 | 2.5 |

Table 1: Rotation Calibration Summary

**Translation**

Translation calibration was undertaken on a bench inside the Merchant Venturers Building. Several sheets of paper were taped to the bench to span a distance of 1 metre and the robot was placed at one end of the span in a specially constructed harness, which anchored the robot in a fixed starting position. The robot was then commanded to translate by a distance in the range +0.5 to +1.0m and the actual translation was measured using a tape measure. Five measurements were taken at each distance.

The results of translation calibration are summarised below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Actual Distance (cm)** | **50** | **75** | **100** |
| **Measured Distance (cm, Mean)** | 47.6 | 71.5 | 95.4 |
| **Measured Distance (cm, Std Dev)** | 0.2 | 0.2 | 0.3 |

Table 2: Translation Calibration Results Summary

**Sensor**

Sensor calibration was undertaken in a lab in the Merchant Venturers Building using the robot arena. The robot’s centre of mass was positioned at a known distance between 10 and 100cm from a far wall of the arena, and the distance returned by the ultrasound sensor was measured over 5 consecutive samples.

The results of the ultrasound calibration are summarised below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Actual Distance (cm)** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** | **55** | **60** | **65** | **70** | **75** | **80** | **85** | **90** | **95** | **100** |
| **Measured Distance (cm, Mean)** | 13 | 16 | 21 | 24 | 28 | 33 | 38 | 43 | 49 | 54 | 58 | 63.2 | 69 | 75 | 79 | 86 | 90.4 | 94 | 99 |
| **Measured Distance (cm, Std Dev)** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.45 | 0 | 0 | 0 | 0 | 0.55 | 0 | 0 |

Table 3: Sensor Calibration Results Summary

**Application of Calibration Results**

The calibration results were used in two different ways:

1. **To remove the mean error from the odometry (translation and rotation) and sensor (distance) readings**

The calibration results were used to ‘calibrate out’ the mean error from the odometry and sensor readings. The measured readings were plotted against the actual readings and the coefficients of a linear regression model were calculated to define a ‘line-of-best-fit’. The linear regression models were then used to adjust the parameters passed to the robot; for example, the linear regression model for translation increased the translation distance passed to the robot so that it travelled a little further and, as a result, the travelled distance better approximated the requested distance.

The linear regression models used to calibrate out the mean errors are as follows:

* Translation: translationactual = 1.0469 translationrequested + 0.1326
* Rotation: rotationactual = 1.074 rotationrequested + 3.8933
* Sensor: distanceactual = 1.0078 distancemeasured + 0.2267

1. **To quantify the noise after removing the mean error as described in 1**

The translation and sensor error varied linearly with distance and had low variance, so the linear regression model successfully removed most of the error. However, the rotation error was observed to be less linear with a high variance, so a large error remained even after the linear regression model was applied. To capture the uncertainty in rotation and, to a lesser extent translation and the distance measurements from the sensor, the standard deviations of rotation, translation, and the sensor readings were computed after removing the mean error. These standard deviations together constitute a noise model which was input to the particles to better model the odometry and sensor readings of the robot.

The noise model is as follows:

* Translation Noise: 0.012cm
* Rotation: 5.444°
* Sensor Noise: 1.363cm

**Section 3: Task Planning**

The robot implements high-level task planning to first localise itself and then navigate to the target location. The processes it uses are described below:

* **Initialize:**

The *Initialize* process creates a Bot object to control the robot. The boundaries of the map are inflated by the robot radius, 5cm, plus an additional 5cm safety margin.



* **Localise Robot using Particle Filter:**

The *Localise Robot* process runs the particle filter which returns the ‘ghost bot’, a BotSim object containing the estimated location and orientation of the robot. The estimated location is compared to the target location to determine whether the robot has arrived at the target location within an error bound of ±2cm.

* **Get Distance and Angle to Next Waypoint using Visibility Graph:**

The *Get Distance and Angle to Next Waypoint* process retrieves the distance and angle along which the robot must travel to reach the next waypoint on route to the target location.

* **Turn Robot and Ghost Robot:**

The *Turn* process turns the robot and the ghost bot to orient them towards the next waypoint.

* **Get Distance in Front of Robot:**

The *Get Distance in Front of Robot* process obtains the distance to the nearest obstacle in front of the robot. This distance is compared to the distance to the next waypoint, computed using the visibility graph, and, if the result indicates that the robot would collide with an obstacle, the particle filter is re-run to re-localise the robot.

Figure 3: Flowchart

* **Move Robot and Ghost Robot:**

The robot and ghost bot are moved the distance to the next waypoint.

* **Signal Completion:**

The robot generates an audible alarm when it determines that it has arrived at the target location and the routine ends.

**Section 4: Conclusions**

In hindsight, better to mount wheels centrally as rear of robot sometimes collides with obstacles. Similarly, ultrasound sensor should have been mounted over wheels along axis of rotation.

Calibration should have been performed in actual arena to capture characteristics of surfaces i.e. friction, reflectivity, etc.