Robot Implementation

P15194468

Sam Fuller

2019

# Architecture

The robot implements a finite state machine architecture. A finite state machine consists of several behaviours that can only happen one at a time and each behaviour must complete in full before changing state, they usually have a controller to determine the next behaviour to run. A finite state machine is perfect for this project because there are a set list of tasks that need to be completed individually and in order. To implement this all of the tasks have been separated out into MATLAB formulas, the controller simply performs one behaviour after another until the program is

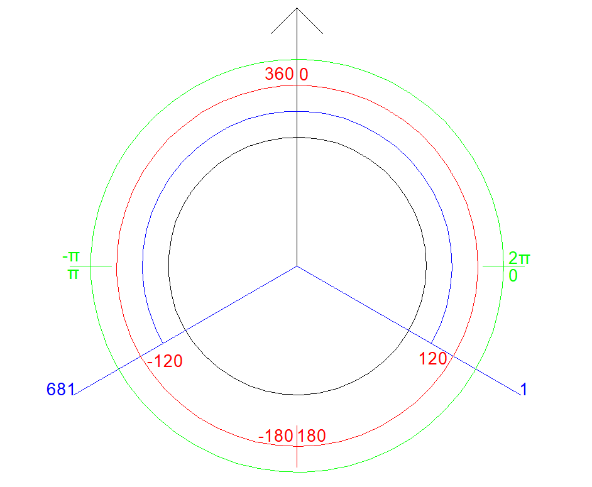


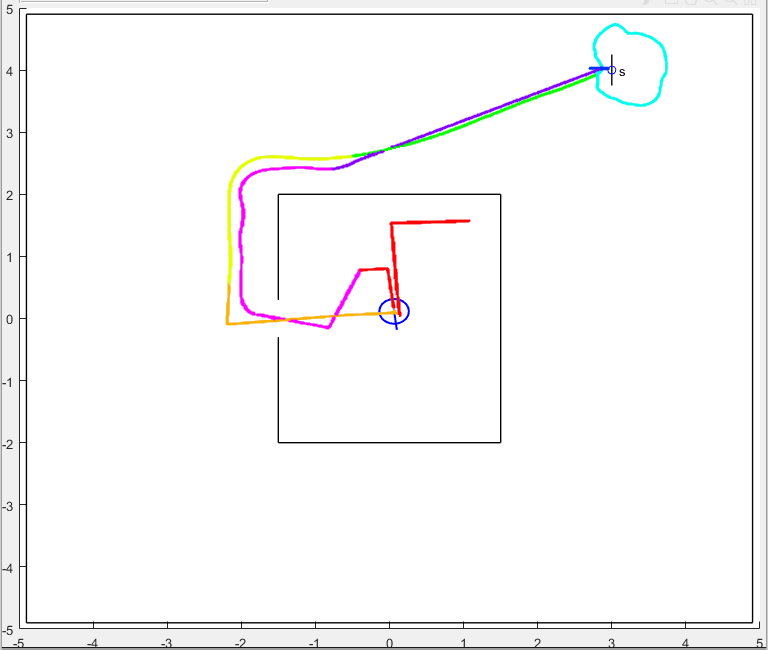
Figure 1

In the IRobot create simulator (used to create the robot) many different angles and measurements are used, figure 1 was created to help translate between all of the different angles. Throughout this report the following colours will be used to differentiate between:

Radians (in green) are used when getting an angle from the simulator, they are always relative to the map with 0 being on the right.

Degrees (in red) are used when passing an angle to the simulator, they are always relative to the robots heading and count anticlockwise with +90° being on the right of the robot and -90° on the left.

Lidar (in blue) is used to describe the individual lidar measurements, unlike degrees and radians the lidar values only take up from +120° to -120° from the robots heading with 1 being at +120° and 681 at -120°.



# Behaviours

## Find Centre

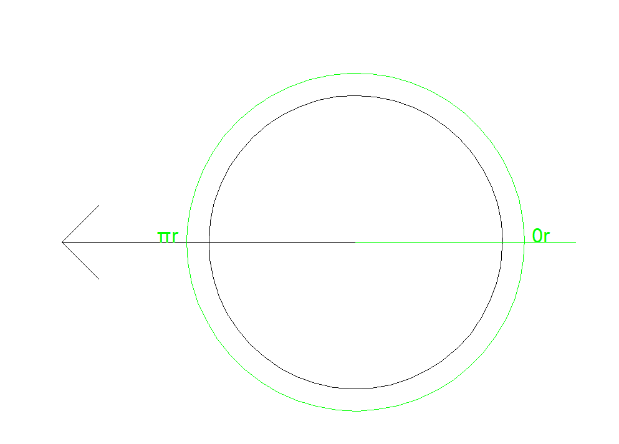
The find centre behaviour is split in to two parts, first the robot aligns itself horizontally, then the robot turns 90° and aligns itself vertically. One limit of finding the centre this way is that the robot cannot initially start this behaviour on the same level as the exit to the room, this will send a maximum reading to the sensors and not the actual distance to the edge of the room.

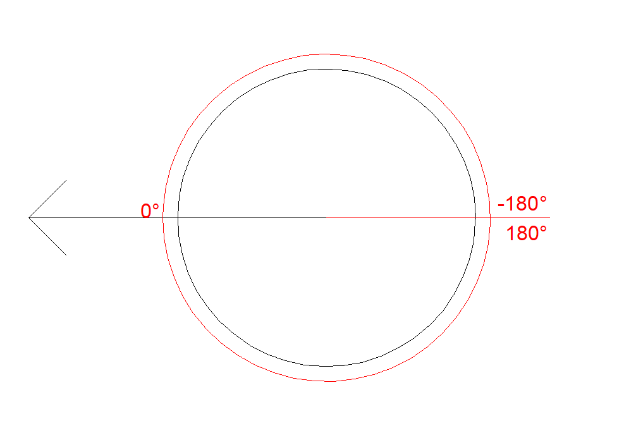
Each alignment compares forward and backward sensors and drives towards the larger value of the two until within a small tolerance. The tolerance is needed because the robot will practically never reach the exact centre of the two walls.

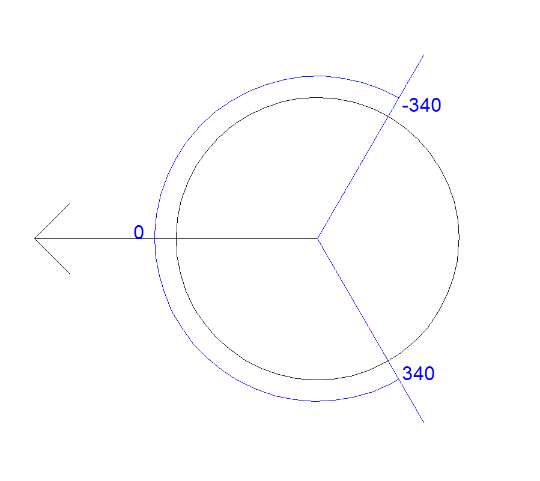
## Leave Room

The leave room behaviour assumes the centre of the room has been found, the robot is turned towards the door and the top and bottom lidars are compared, if the difference is greater than a given tolerance, the robot turns towards the larger measurement. Once close enough to the door the robot calculates its heading a final time, and simply moves forward until it has passed through. A PID controller is not used because the door is only slightly larger than the robot, initially a PID controller was tested, however this would cause the robot to turn erratically when passing through the door.

The exact up and down lidars need to be calculated to give an exact difference and therefore position relative to the door. This is done in four stages:

1. The relative heading is calculated using in radians relative to the right of the map.



1. The relative heading is converted to degrees and flipped in order to get the ideal heading (relative to the left of the map.
2. The angle is converted to a lidar offset, this can be done with the formula:
3. The correct lidars are found using the offset and the two lidars at 90° angles to the heading (85 and 595)

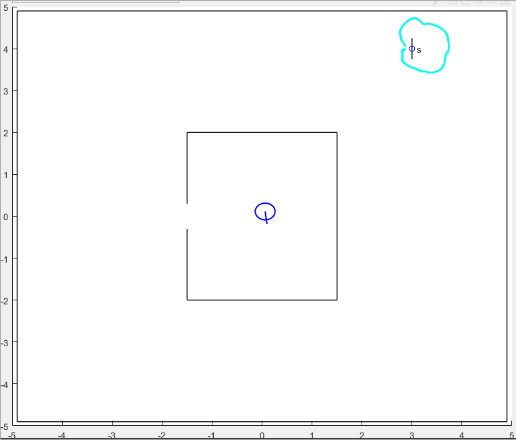
## Finding Beacon

To find the beacon the room is wall followed until the beacon is both found and within range. The wall following is a PID controller set to follow at 0.5m using all of the left sensors (1:340). The beacon needs to be within range initially because the camera creates noise, if this behaviour ends once the beacon has been spotted once, then the camera might not be able to reliably find the beacon when trying to approach it in the next behaviour.

In order to find the distance to the beacon the following formula is used:

## Travel to Beacon

This behaviour simply loops until within 0.75m of the beacon, each cycle turns the robot towards the beacon and travels forward for 0.1s.



## Circle Beacon

This behaviour uses a PID controller to wall follow, there are several differences to the wall following controller used in the find beacon behaviour:

1. Only sensors 1:200 are used, this prevents the external walls from being mistaken as a continuation of the wall and followed instead of the wall. The more forward-facing sensors are not needed as they are only necessary when approaching an internal corner, which we can be confident there are none of.
2. A special case has been added so that if no walls are seen within range the robot turns right, this helps with the very sharp corners where the robot would over shoot and loose sight of the wall.

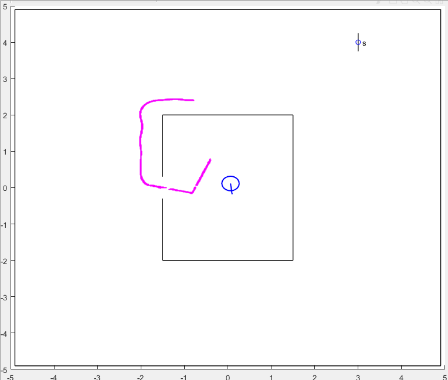
In order to keep track of when the robot has fully circled the beacon wall two Booleans are used, once the robot is facing downwards it has completed half a loop, if the robot is facing upward and has completed half a loop then it has completed a full loop. Once the robot has completed a full loop the next behaviour is performed.

## Bump Beacon

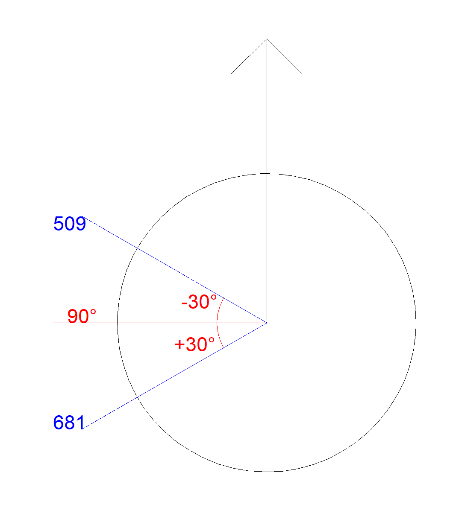
This behaviour simply turns towards the beacon, goes backward, go forwards until the robot bumps the beacon, then goes backwards again.

## Find Room

To re-find the room the angle that the robot approached the beacon was stored, the robot simply turns 180° – the angle of income in order to point towards the room. The robot then drives forward until the room is within 0.5m.



## Wall Follow

This behaviour uses a PID controller to wall follow, this wall following follows at a closer distance to the previous two (0.35m) and uses fewer lidar 509:681 (90° ± 30°)

## Find Centre

Finally the robot finds the center of the room using the same find centre function. In order to get around the limits of the function, the robot first navigates to one side of the room.