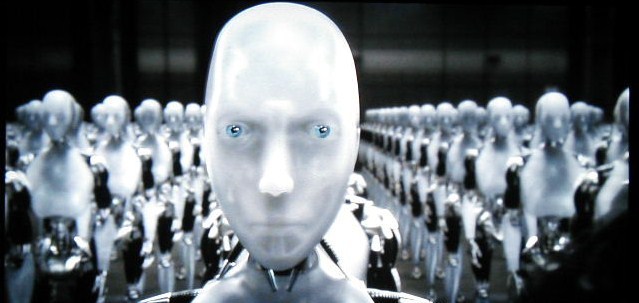
3/5/2016

Paul Dockar, Dennis Kim, Kimi Izzo, Michael Smith, Nick Rabey

Lemongrabs

Assignment 2 Report

Mechatronics 2 - 48623

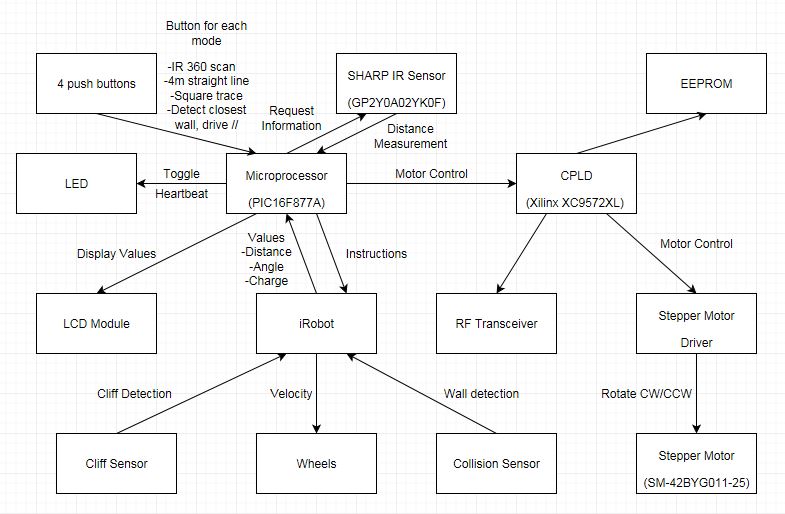
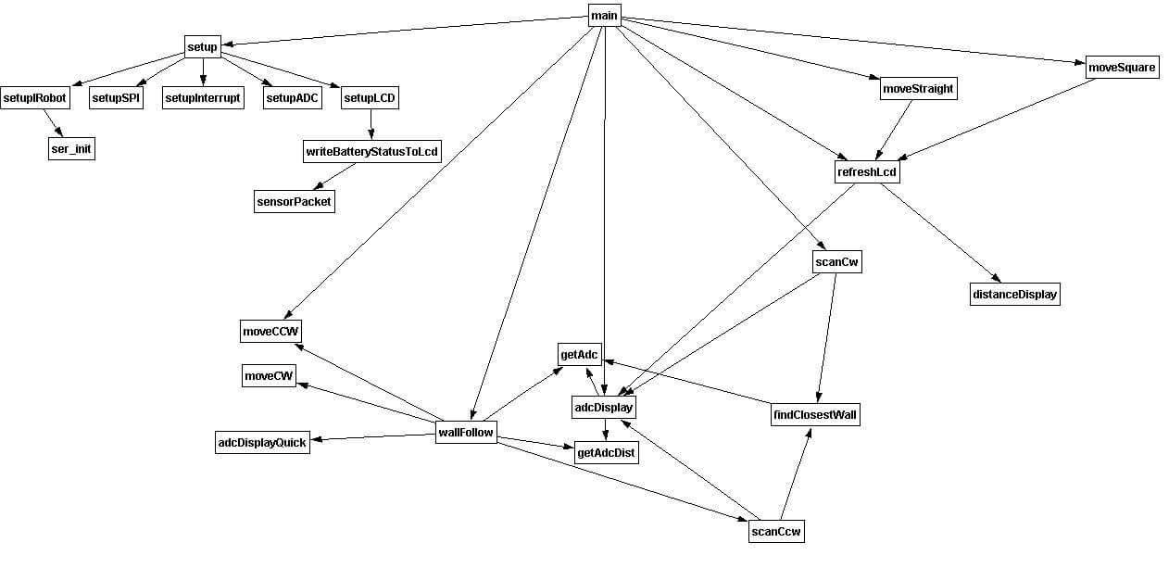


# Introduction

This report details the processes and results of programming iRobot Create to:

1. Flash a heartbeat LED at 1Hz.
2. Continuously display the IR sensor range (converted into cm) on the LCD.
3. On a button press, scan the IR sensor clockwise 360⁰ and return the sensor to face the nearest obstacle
4. On a button press, drive Create 4m whilst continually displaying total linear distance moved on the LCD screen
5. On a button press drive Create in a square shaped trajectory, whilst continuously displaying total linear distance travelled.
6. On a button press, scan for closest wall and drive parallel to that wall, maintaining a constant distance.

# Block Diagram

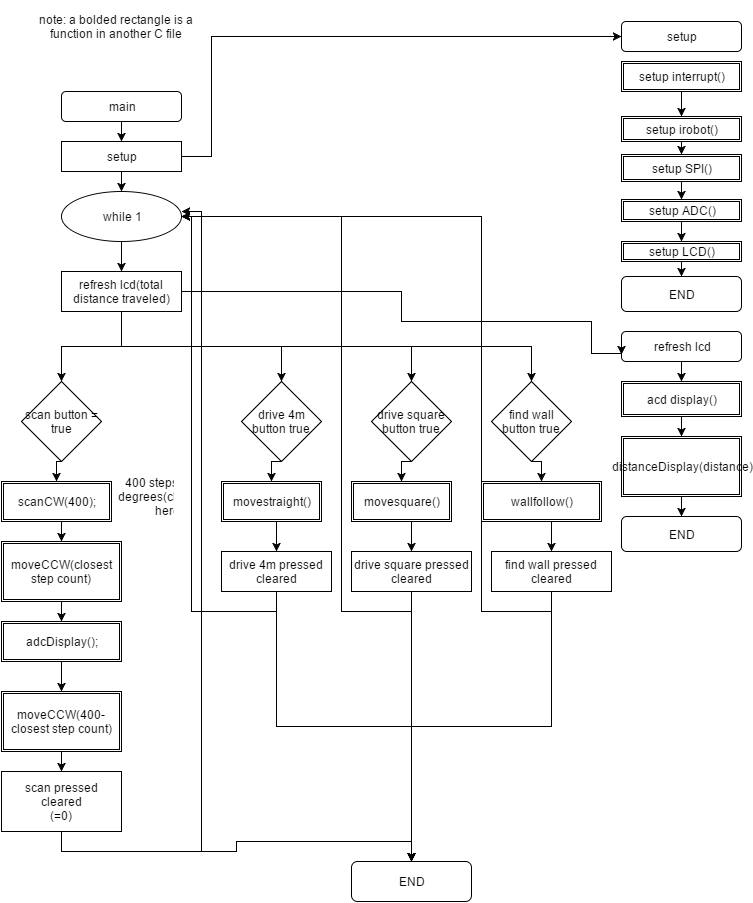
**Figure 1 – Hardware Block Diagram**

**Figure 2 – Software Block Diagram**

# Code flowchart

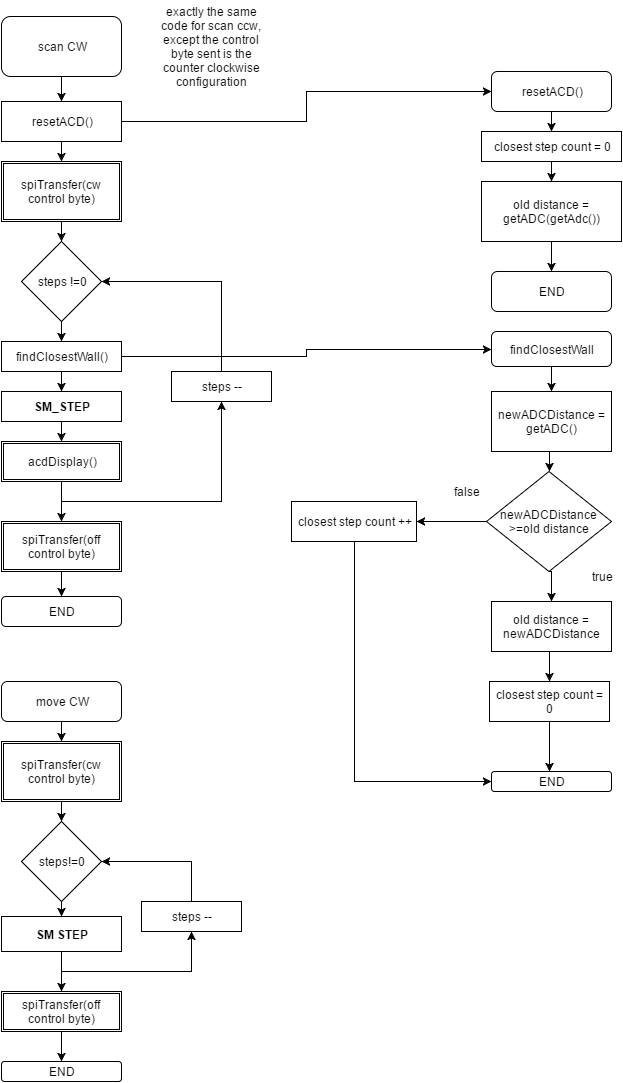
## Main

Main.c contains an infinite while loop which checks all button flags to perform functions, and constantly refreshes the ADC distance measurement on the LCD. Main also contains the setup function, which calls each individual setup for all components of the IRobot system; LCD, ADC, iRobot, SPI and interrupt serices.



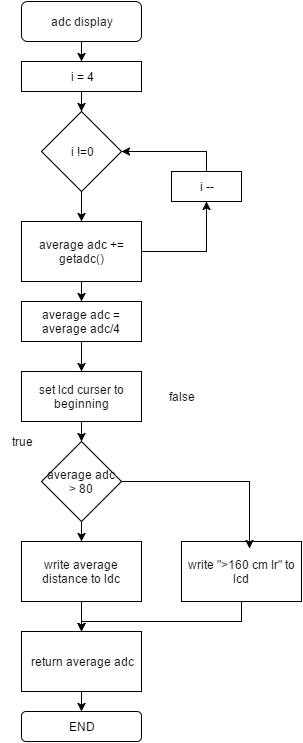
## Stepper

Stepper program contains an important function; Scan IR sensor (mounted to stepper motor), which finds the closest wall, which uses the ADC to measure distances per half step, and 2 important variables, one to store the closest distance measured, and one to count the amount of half-steps to turn the stepper motor back to this position after a 360 degree turn.

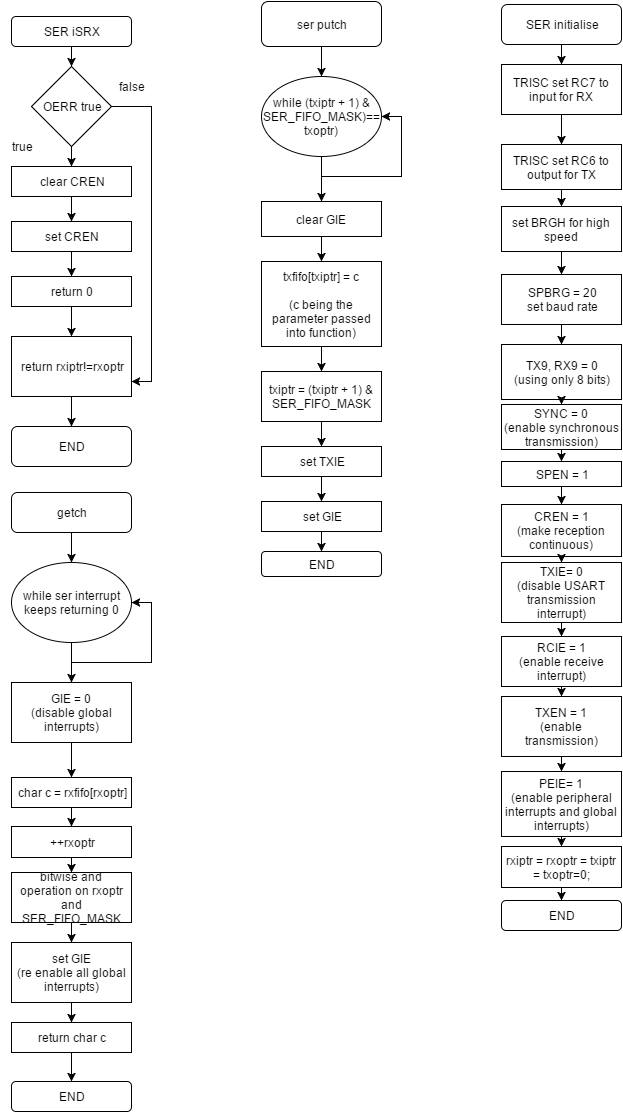


## ACD

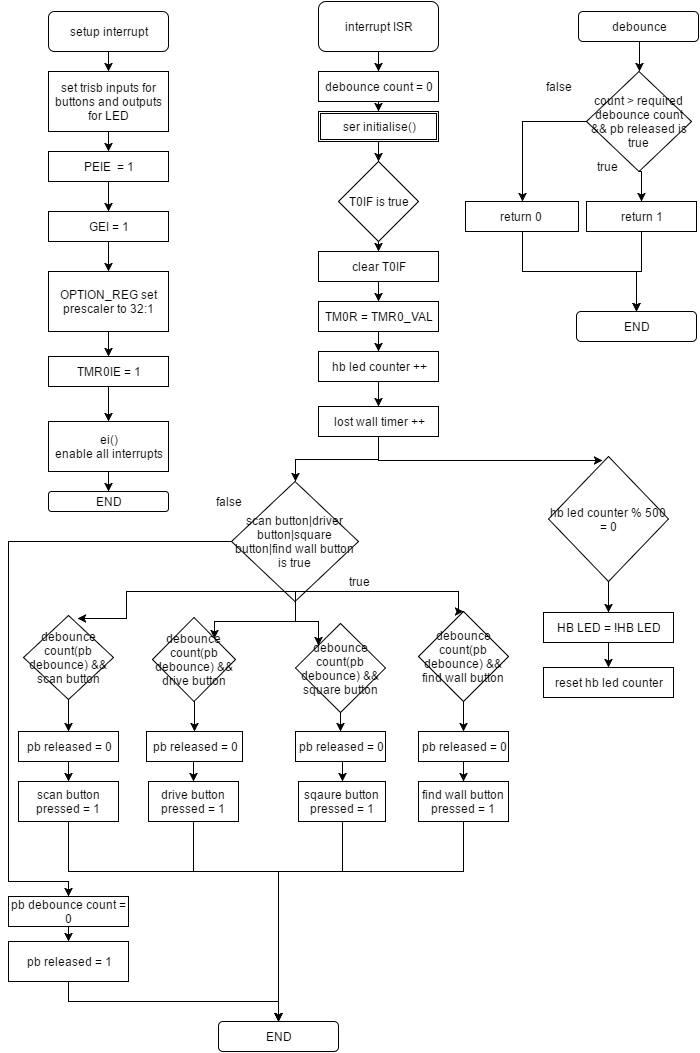
ADC contains a function to accurately measure the ADC reading four times, then average this reading and convert into cm’s for use in other functions and the LCD display.



## SER

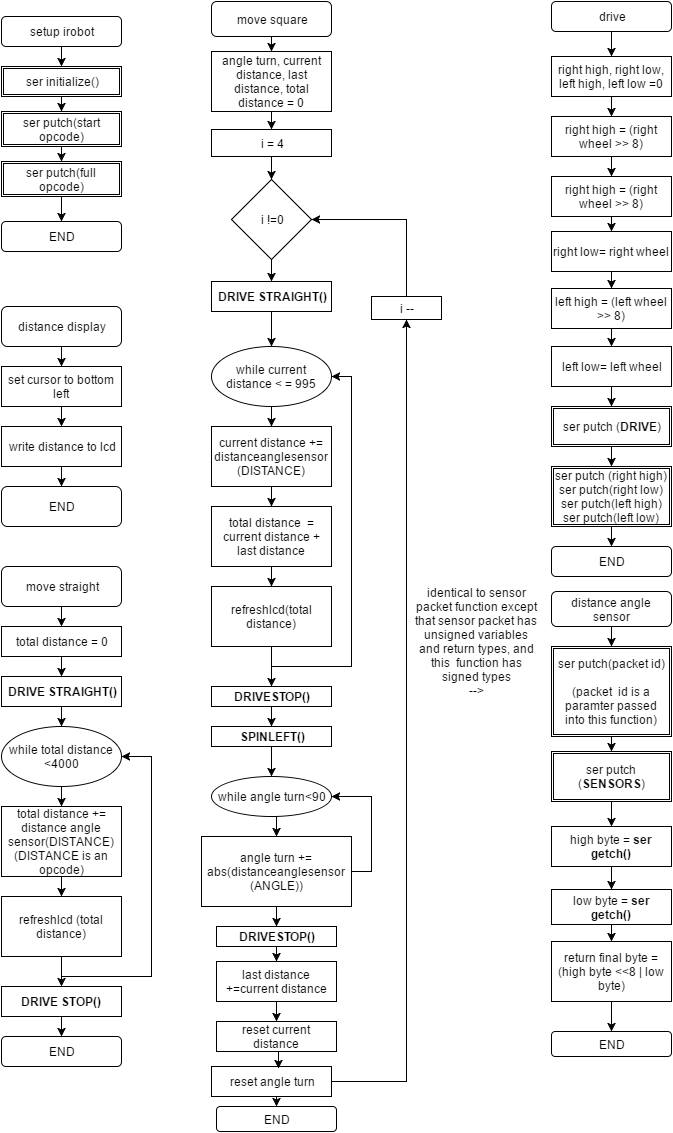


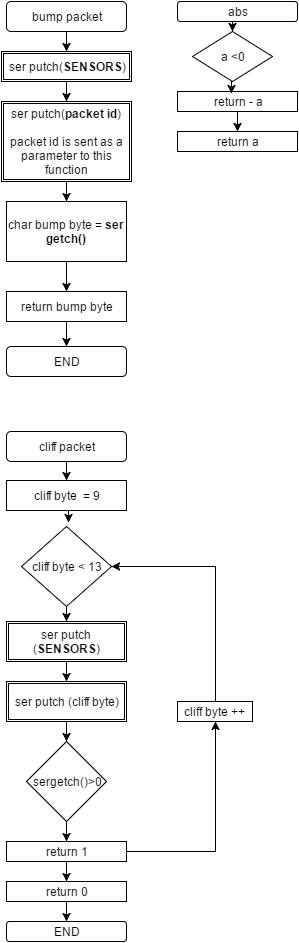
## INTERRUPT

The interrupt service contains all debouncing functions for the push buttons, the timer for heartbeat LED (set to 500ms), and the lost\_wall\_timer counter, which is extensively used in the wallFollow function, allowing the robot to make quick 180 degree turns around the maze.

## ROBOT

IRobot contains 3 of the 4 push button functions. These 3 are moveStraight, moveSquare, and wallFollow. moveStraight is a simply function which sets the wheels to move forward at 200mm/s, whilst polling the distance travelled from the iRobot using the ser\_getch functions. moveSquare is similar, but contains a for loop, and turning 90 degrees every 1m travelled to achieve a square like pattern.

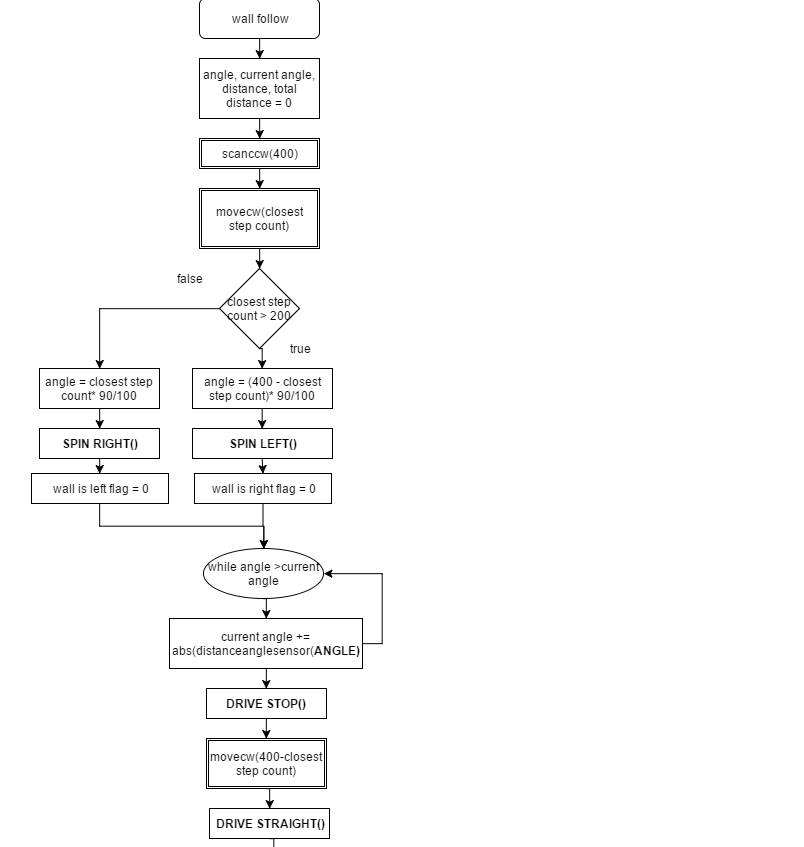


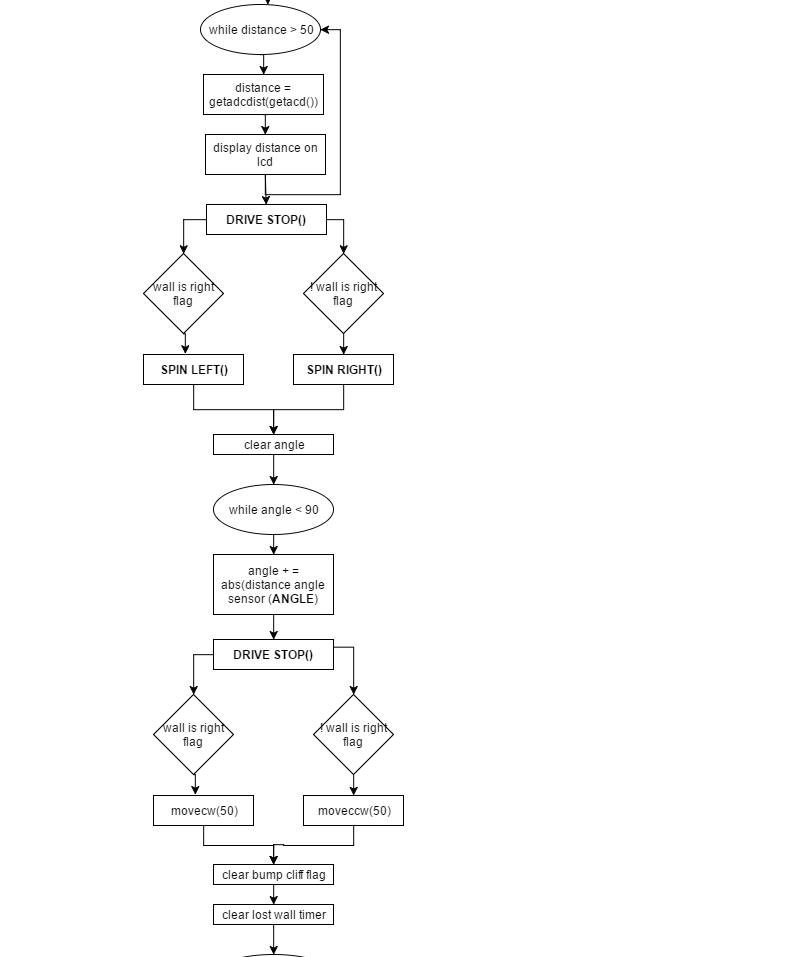


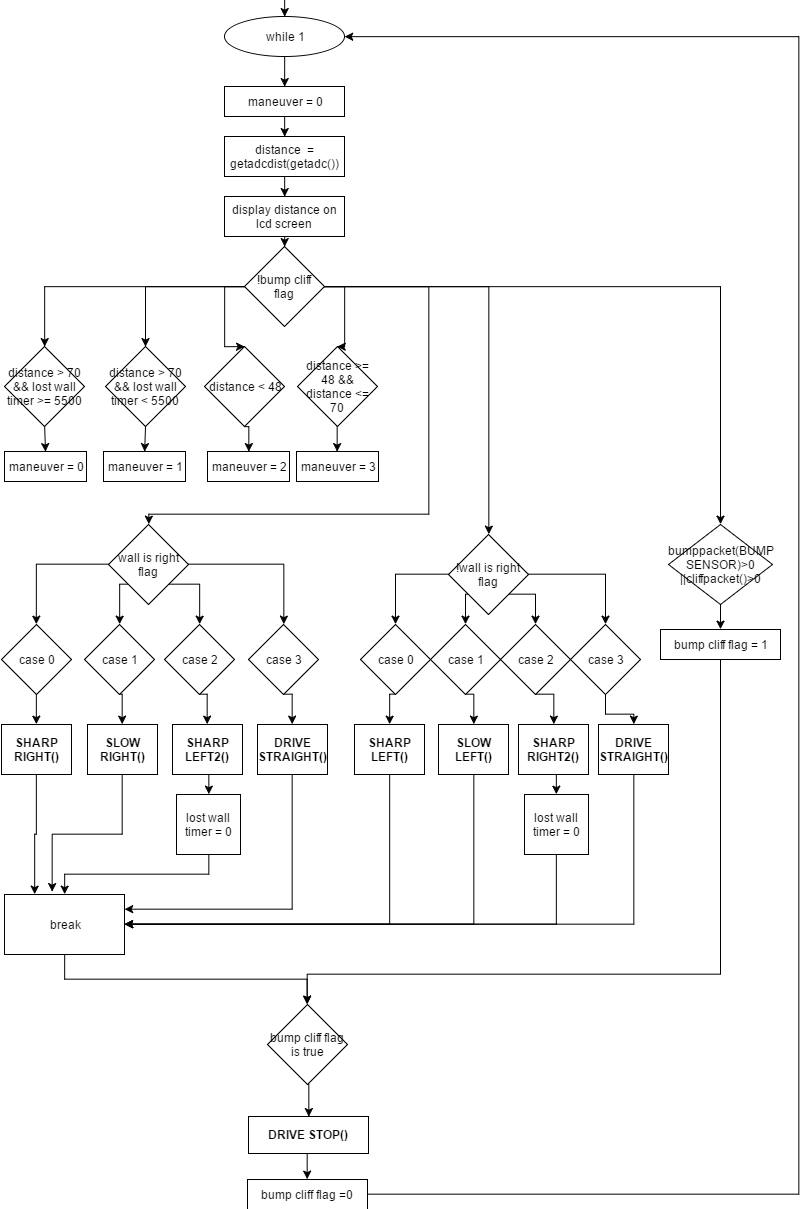
## ROBOT WALL FOLLOW FUNCTION

The iRobot wallFollow function is the largest for the program. It performs an IR scan 360 degrees to find the closest wall (previously mentioned), and then orientates towards the wall, driving to a 50cm distance from it. The PIC sets a flag depending on initial position of the robot to the wall. If the closest wall was found to the right of the robot, a flag is set, which sets the robot to follow the wall to the right of it, and vice-versa if on the left.

The robot will then turn 90 degrees depending on the closets wall flag, and start driving forward. The IR sensor is rotated 45 degrees from the forward position of the robot, and the IR sensor is then frequently polled to determine the distance from the wall. Multiple single line if statements, and a switch statement determines which manoeuvres the robot will perform. If the IR sensor reads the wall is too far away for example, the robot will slowly turn back towards the wall.







# Characterisation Graphs

* Straight Line Manoeuver for different speeds (actual distance for 5 attempts):

|  |  |  |
| --- | --- | --- |
| Speed(mm/s) | Distance(cm) | Command Distance(cm) |
| -100 | 404 | 400 |
| -100 | 403 | 400 |
| -100 | 401 | 400 |
| -100 | 401 | 400 |
| -100 | 403 | 400 |
| 100 | 398 | 400 |
| 100 | 404 | 400 |
| 100 | 402 | 400 |
| 100 | 403 | 400 |
| 100 | 399 | 400 |
| 200 | 398 | 400 |
| 200 | 397 | 400 |
| 200 | 403 | 400 |
| 200 | 400 | 400 |
| 200 | 401 | 400 |
| 300 | 403 | 400 |
| 300 | 395 | 400 |
| 300 | 399 | 400 |
| 300 | 404 | 400 |
| 300 | 405 | 400 |
| 400 | 430 | 400 |
| 400 | 422 | 400 |
| 400 | 427 | 400 |
| 400 | 424 | 400 |
| 400 | 432 | 400 |

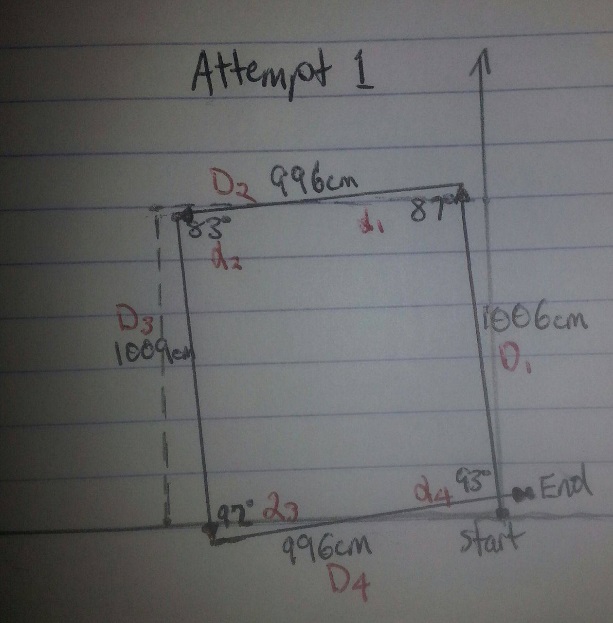
From the graph and table above we can see that the slower speeds give more consistent results that remain closest to the required 4m mark. However, during our measurements iRobot Create had a tendency to drift from a straight path at slower speeds. From this observation we can conclude that 200mm/s is the optimal velocity for the purpose of this assignment.

* Straight line manoeuver for different distances:

|  |  |
| --- | --- |
| Com. Dist (m) | Actual Dist (cm) |
| 0.5 | 48 |
| 0.5 | 48 |
| 0.5 | 53 |
| 0.5 | 51 |
| 0.5 | 49 |
| 1 | 101 |
| 1 | 99 |
| 1 | 98 |
| 1 | 100 |
| 1 | 101 |
| 1.5 | 155 |
| 1.5 | 149 |
| 1.5 | 152 |
| 1.5 | 149 |
| 1.5 | 153 |
| 2 | 202 |
| 2 | 202 |
| 2 | 201 |
| 2 | 201 |
| 2 | 202 |
| 2.5 | 250 |
| 2.5 | 251 |
| 2.5 | 249 |
| 2.5 | 251 |
| 2.5 | 250 |
| 3 | 304 |
| 3 | 302 |
| 3 | 300 |
| 3 | 301 |
| 3 | 302 |

* Square Shaped Manoeuver:
* Attempt 1

|  |  |
| --- | --- |
| **Angle (Degrees)** | **Distance Travelled (cm)** |
| A1 = 87 | D1 = 1006 |
| A2 = 83 | D2 = 996 |
| A3 = 92 | D3 = 1009 |
| A4 = 93 | D4 = 996 |

* Attempt 2

|  |  |
| --- | --- |
| **Angle** | **Distance Travelled** |
| A1 = 87 | D1 = 1006 |
| A2 = 85 | D2 = 996 |
| A3 = 91 | D3 = 1006 |
| A4 = 94 | D4 = 996 |

* Attempt 3

|  |  |
| --- | --- |
| **Angle** | **Distance Travelled** |
| A1 = 90 | D1 = 1004 |
| A2 = 83 | D2 = 997 |
| A3 = 93 | D3 = 1006 |
| A4 = 93 | D4 = 996 |

* Attempt 4

|  |  |
| --- | --- |
| **Angle** | **Distance Travelled** |
| A1 = 88 | D1 = 1006 |
| A2 = 79 | D2 = 996 |
| A3 = 96 | D3 = 1003 |
| A4 = 91 | D4 = 1001 |

* Attempt 5

|  |  |
| --- | --- |
| **Angle** | **Distance Travelled** |
| A1 = 87 | D1 = 1005 |
| A2 = 90 | D2 = 999 |
| A3 = 92 | D3 = 997 |
| A4 = 93 | D4 = 1003 |

Three Factors may have contributed to the errors in orientation of the robot:   
  
1) **Different spin rates of the motors** - causing the wheels to spin unevenly.   
2) **Battery charge** – robot tended to perform more accurately when fully charged.   
3**) Delays** – small delays meant robot would rotate slightly more than the specified angle.