# 1. Introduction

For this project, many different modules are combined to create a functional robot that can perform the desired routine as per the specifications; therefore a bottom-up strategy will be employed to help target the source of the error, which is especially important when the robot is running multiple threads at the same time. The test cases designed will try to minimize the chances of an error happening on D-Day, but also try to make sure it would not cause the testers to overspend its time budget on it.

# 2. Test Items

The following items will be the focus of the robot:

* Basic cases
  + Odometry
  + Navigation
  + Bluetooth
  + Ball launcher
  + Localization
* Intermediate cases
  + Obstacle avoidance (ultrasonic)
  + Obstacle avoidance (touch sensor)
* Advanced cases
  + Defender role
  + Attacker role

# 3. Test cases

## Basic cases:

### Odometry

|  |  |
| --- | --- |
| Test case ID | 1.1.1 |
| Case name | Rotate |
| Purpose | To determine if the robot can accurately know where it is pointing at after rotation is performed |
| Description | The robot will turn to various angles and see if it can turn back to its origin |
| Prerequisites | N/A |
| Steps | 1. Make sure the robot is pointed to the 0 degree for this test 2. Turn to 60 3. Turn to 180 4. Turn to 350 5. Turn to 80 6. Turn to 0 |
| Actual results |  |
| Expected results | Robot can point to the various points accurately, and also take the smallest angle to turn; odometry should properly reflect angles turned |

|  |  |
| --- | --- |
| Test case ID | 1.1.2 |
| Case name | Walk straight |
| Purpose | To determine if the robot can accurately walk a predefined distance straight |
| Description | The robot will walk various amounts of distance, and see if it can keep it straight |
| Prerequisites | N/A |
| Steps | 1. Make sure the robot is pointed to the 0 degree for this test 2. Walk 30 units forward 3. Walk 30 unit forward |
| Actual results |  |
| Expected results | Robot can walk straight, so that its angle is perpendicular to where it started; the odometry reported should be reasonable accurate |

### Navigation

|  |  |
| --- | --- |
| Test case ID | 1.1.3 |
| Case name | Travel To |
| Purpose | To determine if the robot can travel to desired location |
| Description | The robot will walk various amounts of distance, and the offset from the origin will be used to determine if the robot is accurate enough |
| Prerequisites | N/A |
| Steps | 1. Make sure the robot is pointed to the 0 degree for this test 2. Walk to (0, 30) 3. Walk to (30, 15) 4. Walk to (30,30) 5. Walk to (0,15) 6. Walk to (0,0) |
| Actual results |  |
| Expected results | Robot will take the direct angle to travel to the points, while adjusting on its way. It shouldn’t be too off from the origin |

### Bluetooth

|  |  |
| --- | --- |
| Test case ID | 1.3.1 |
| Case name | Receive coordinates |
| Purpose | To determine if the robot can receive the Bluetooth coordinates using the given program |
| Description | The robot will receive a pair of coordinates and display it. |
| Prerequisites | N/A |
| Steps | 1. Start the Bluetooth program and robot 2. Transmit a pair of coordinates 3. Display it on LCD to acknowledge 4. Repeat one more time |
| Actual results |  |
| Expected results | Robot will display the proper coordinates on the LCD. |

### Ball launcher

|  |  |
| --- | --- |
| Test case ID | 1.4.1 |
| Case name | Receive coordinates |
| Purpose | To determine if the robot can receive the Bluetooth coordinates using the given program |
| Description | The robot will receive a pair of coordinates and display it. |
| Prerequisites | N/A |
| Steps | 1. Start the Bluetooth program and robot 2. Transmit a pair of coordinates 3. Display it on LCD to acknowledge 4. Repeat one more time |
| Actual results |  |
| Expected results | Robot will display the proper coordinates on the LCD. |