# 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
  + Ball loading
* Intermediate cases
  + Obstacle avoidance (ultrasonic)
  + Obstacle avoidance (touch sensor)
* Advanced cases
  + Defender role
  + Attacker role
* Optional functions
  + Odometry correction

# 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, facing the correct orientation |
| 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. Also the robot needs to point at indicated angle when it is at the destination. |
| Prerequisites | N/A |
| Steps | 1. Make sure the robot is pointed to the 0 degree for this test 2. Walk to (0, 30,0) 3. Walk to (30, 15,45) 4. Walk to (30,30,100) 5. Walk to (0,15,250) 6. Walk to (0,0,0) |
| Actual results |  |
| Expected results | ???? |

### Bluetooth

|  |  |
| --- | --- |
| Test case ID | 1.3.1 |
| Case name | Receive information over Bluetooth |
| Purpose | To determine if the robot can receive the Bluetooth coordinates using the given program |
| Description | The robot displays the received information transferred via Bluetooth |
| 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 information on the LCD |

### Ball launcher

|  |  |
| --- | --- |
| Test case ID | 1.4.1 |
| Case name | Launch ball |
| Purpose | To determine if the robot can launch a ball through the target with a minimum of one bounce, from a range of given distance |
| Description | The robot will receive a pair of coordinates and shoot the ball to the target. The test will try to find the min/max angle for each foot |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | Robot will shoot the ball into the target |

### Localization

|  |  |
| --- | --- |
| Test case ID | 1.5.1 |
| Case name | Sonar falling edge case |
| Purpose | To determine if the robot can get an approximation of where north is. |
| Description | The robot will use the ultrasonic sensor to see where the walls are and find north. It will run it at several locations in the tile, with different orientation |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | Robot can successfully find north within an error of 10 degrees |

|  |  |
| --- | --- |
| Test case ID | 1.5.2 |
| Case name | Light sensor localization |
| Purpose | To determine if the robot can localize itself using the lines between the tiles |
| Description | The robot will start at random locations in the tile, facing North and run the localization routine multiple times |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | Robot can successfully find its location on the board and adjust its odometer |

## Ball loading

|  |  |
| --- | --- |
| Test case ID | 1.6.2 |
| Case name | Ball loading test |
| Purpose | To determine if the ball loading mechanism works reliably |
| Description | The robot will be retrieving a ball from the ball dispenser multiple times |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | Robot can retrieve the ball form the dispenser multiple times |

## Intermediate cases

## Obstacle avoidance (ultrasonic)

|  |  |
| --- | --- |
| Test case ID | 2.1.1 |
| Case name | Ultrasonic test |
| Purpose | To determine if the ultrasonic sensor works properly for obstacle avoidance at varying locations |
| Description | The robot will detect the obstacle using its ultrasonic sensor and avoid it. It should avoid the obstacles from the sides and the front. |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | The robot can avoid the obstacle without touching it and causing it to lose its odometer’s accuracy |

## Obstacle avoidance (touch)

|  |  |
| --- | --- |
| Test case ID | 2.2.1 |
| Case name | Touch sensor-left |
| Purpose | To determine if the left touch sensor can successfully detect and avoid colliding with an obstacle at varying angles |
| Description | The robot will detect the obstacle using its touch sensor and will try to go avoid it |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | The robot can avoid the obstacle without losing its odometer’s accuracy |

|  |  |
| --- | --- |
| Test case ID | 2.2.1 |
| Case name | Touch sensor - right |
| Purpose | To determine if the right touch sensor can successfully detect and avoid colliding with an obstacle at varying angles |
| Description | The robot will detect the obstacle using its touch sensor and will try to go avoid it |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | The robot can avoid the obstacle without losing its odometer’s accuracy |

## Advanced Cases

## Defender Role

|  |  |
| --- | --- |
| Test case ID | 3.1.1 |
| Case name | Localization |
| Purpose | To determine if the localization routine works properly |
| Description |  |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results |  |

|  |  |
| --- | --- |
| Test case ID | 3.1.2 |
| Case name | Localization |
| Purpose | To determine if the localization routine works properly |
| Description | The robot will detect the obstacle using its touch sensor and will try to go avoid it |
| Prerequisites | N/A |
| Steps |  |
| Actual results |  |
| Expected results | The robot can avoid the obstacle without losing its odometer’s accuracy |

## Attacker Role