CSE 4360 / 5364 - Autonomous Robots

Behavior-based Fire alarm Robot

Team11 Project-2-Fall-2018 Report

--Team Members--

Erik Andrup

Jay Denton

Nathan Huang

Timothy Vo

Due Date: 11/28/2018

**Part 1: Robot Design**

**Section 1: Building Robot**

Parts used:

The Programmable Brick, Two Large Motors, The Rotation Sensor, two touch sensor (one is not plugged in to make sure robot hits wall evenly), light sensor, ultrasonic sensor, and others parts that put everything together.

Instruction: Reference website: <https://www.lego.com/en-us/mindstorms/about-ev3>

The robot is built in a simple two-wheeled version. The left wheel with Motor connects to the port OUT\_B; the right wheel with another Motor connects to the port OUT\_C;

Sensors**:**

setAllSensorMode(TOUCH\_PRESS,COL\_COLOR,US\_DIST\_MM,GYRO\_ANG);

**IN\_1** - Touch sensor on front of robot to know if robot hits wall

**IN\_2** - light sensor on color mode facing down on front of robot to know if we reach the candle ( will readblue)

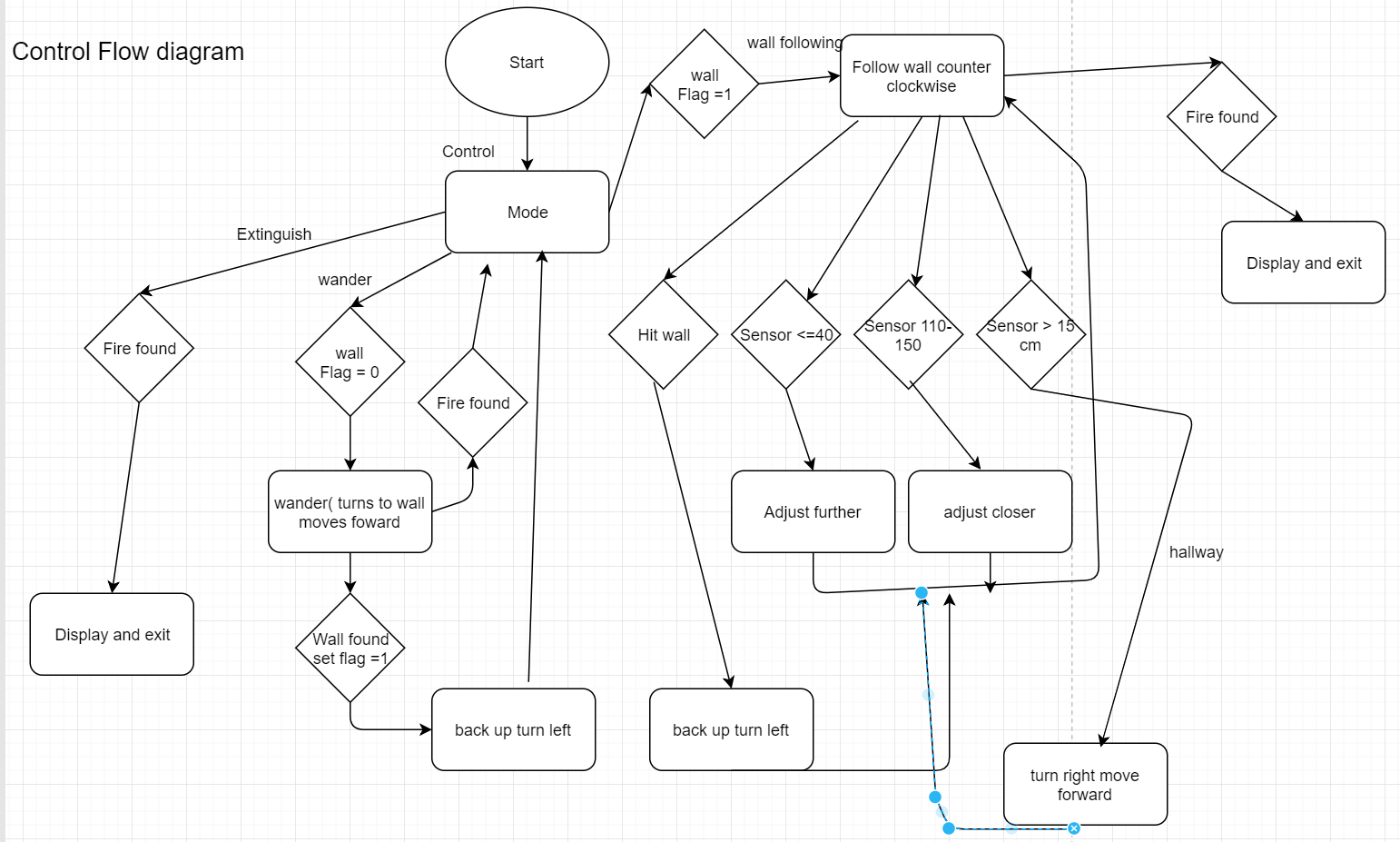
**IN\_3** - Ultra-sonic sensor mounted on right side of robot to read distance on right to assist in wall following

**IN\_4** - Gyrosensor for more accurate turns

**(Side Look)**

****

**Part 2: The Control System**

****

**Overview**

This project is to implement the behavior coordination mechanism to the EV3 Robot. Two main behaviors were implemented, which are Wander and Wall Following.

In a short summary, our robot begins in wander mode until it finds a wall. Then it goes to the wall found and adjust itself to parallel with it by its right side, where the sonar sensor is at. Then the wall following mode is activated. When following the wall, the robot is going straight forward within a safe range, between itself and the paralleling wall, most of the time. We set the safe range to be between 4 cm and 11 cm. If the sonar tells the robot, it is too close to the wall, say less than 4cm, then the robot will move away further. If the sonar tells the robot, it is moving away from the wall, say over 11cm, then the robot will move closer.

The robot won’t stop until it hits the wall in front or found a hallway or detect the fire region. At the end, it should detect the fire and distinguish it.

**Part 3: Code Structure**

1. **Overview**

There are two files, which are **“Fireman.cpp”** with the main() function, and the **“Control.h”** header file.

C++ **goto** statement is widely used in the main function A goto statement provides an unconditional jump from the current code to a labeled statement in the same function.

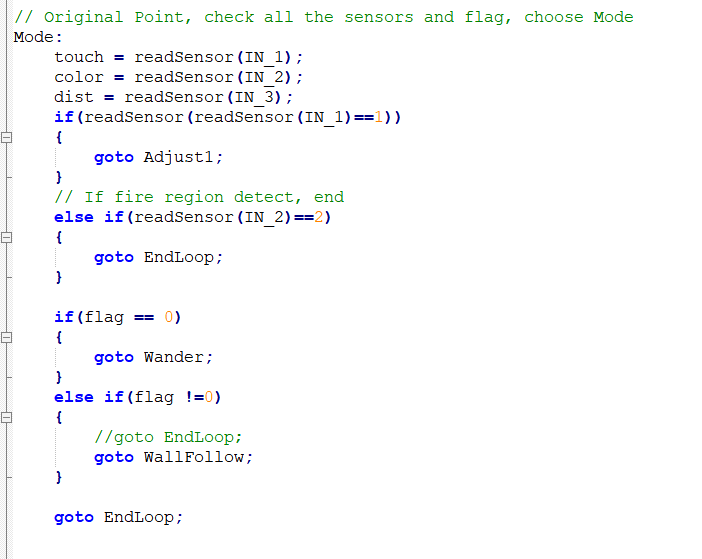
There are 7 labeled statements in the main function:

1. Mode
2. Wander
3. WallFollow
4. Adjust1
5. Adjust2
6. Adjust3
7. EndLoop

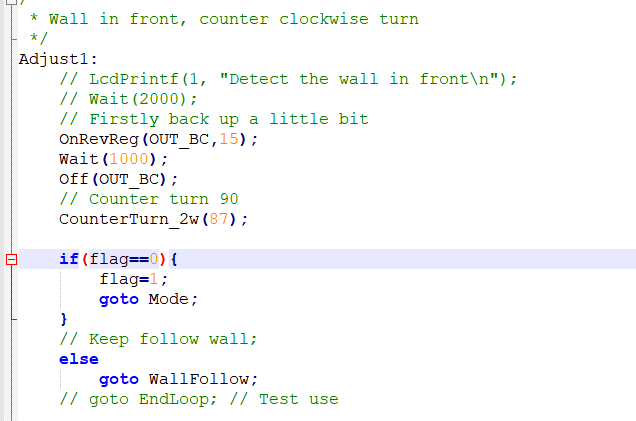
\*Commented Test-use-only labels: CheckTouch, CheckColor, CheckDist.

There are 6 functions defined in the header file:

1. void CounterTurn\_2w(int)
2. void ClockTurn\_1w()
3. Parallel()
4. Wander()
5. void DecreaseDist()
6. void IncreaseDist()
7. **Description**
8. **Fireman.cpp**
9. **Mode statement**

****

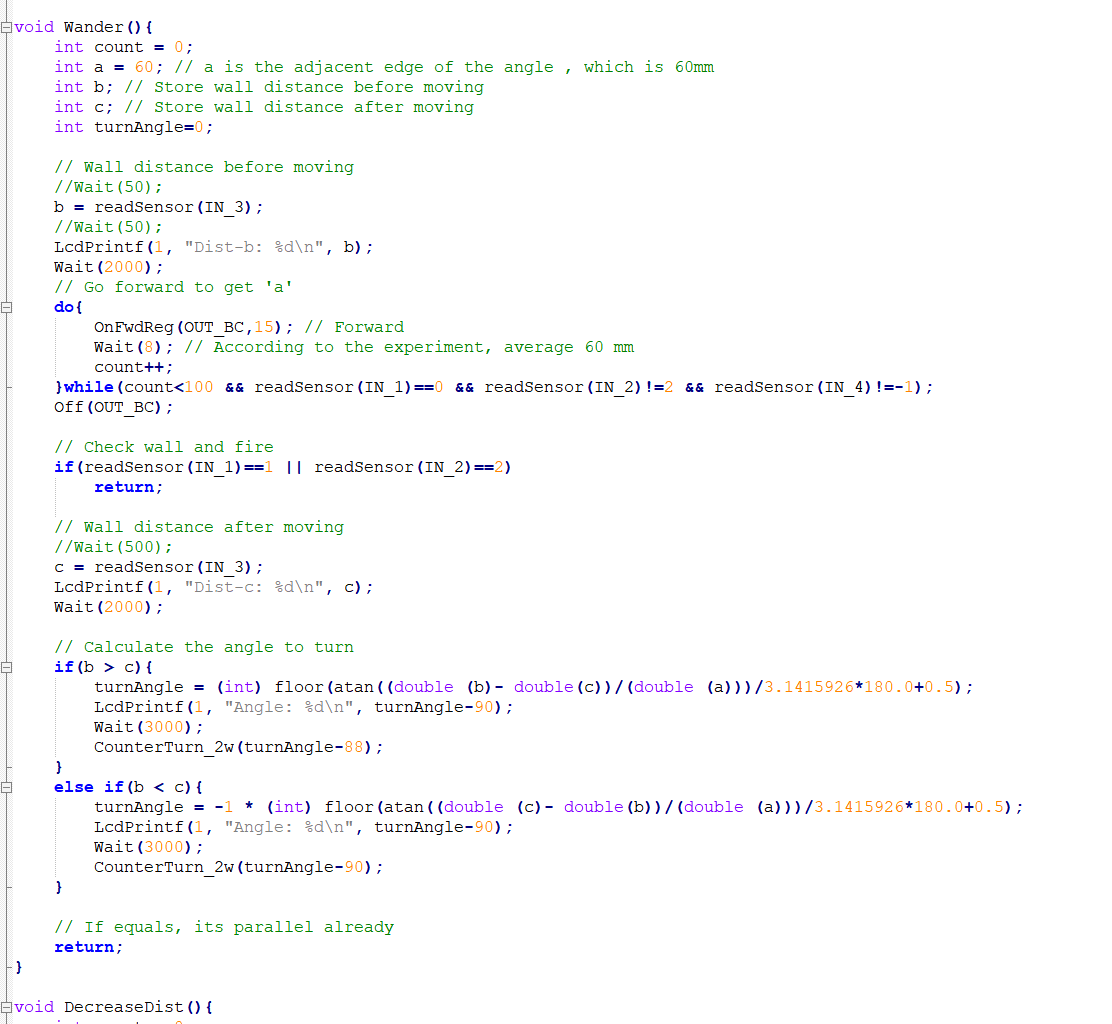
To start the robot goes to the mode loop.

Here it calls awander function until it finds a wall or fire, and then it will return to the control mode. In mode, if the robot has found a wall so it will call Adjust1: which backs the robot up and turns left.

Otherwise if the color sense read blues, it has found a fire and will signal fire found and stop. If the wall flag is 0 ( robot has not found a wall ) the robot Wanders. Otherwise, the robot will begin wall following.

**Section 1: wander**

To start the robot calls a wander function within the wander loop that does the following.

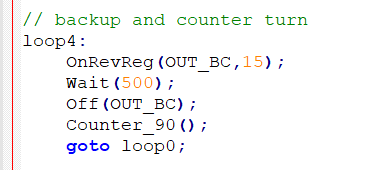


This code is designed to try to find a wall to latch onto and become parallel, or if it hits a wall return to the wander loop and begin wall folowing by backing up and turning left..

If the touch sensor is not triggered, or it doesn’t detect a fire, the robot will move forward. If the color sensor reads the number 2 it has found the fire and stops. When the robot finds a wall it will call adjust 1 ( back up and turn left ).



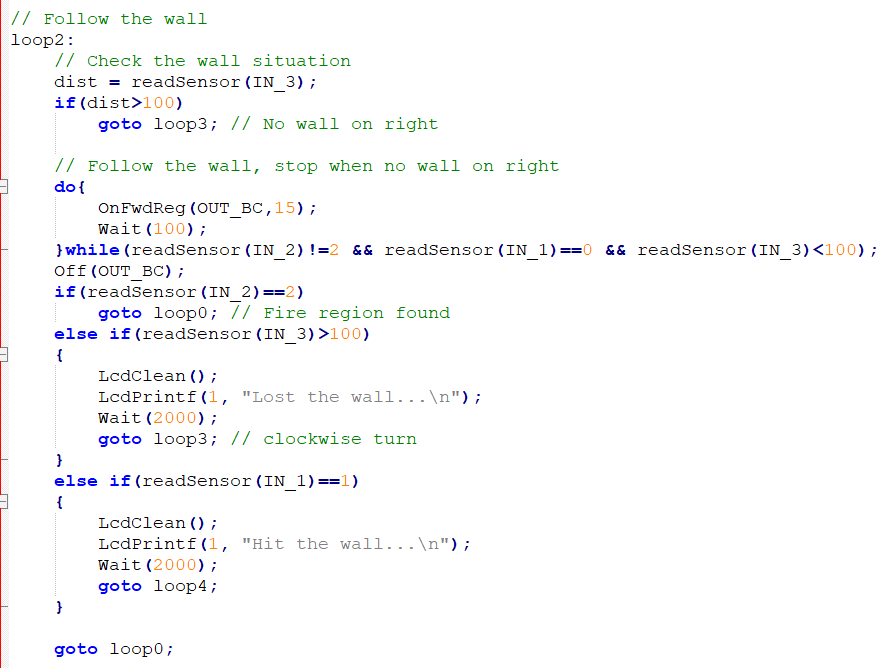
In wander mode the robot is looking for a wall to follow, it does so by moving turning towards a wall and moving forward until the robot hits a wall via its touch sensors or it detects the fire. When the robot hits a wall, it sets the mode flag to 1 to signal to the main control system it found a wall, calling the adjust method.



It then calls loop4 which backs the robot up and turns left. It then goes back to the main control loop where it will begin counter-clockwise wall-following until it finds the candle or another wall.

It sets the wall flag to 0 signalling to the control mode to wall follow.

**Section 2: Wall following**



Once the robot has found a wall, the robot will begin to follow the wall in a counter-clockwise motion. It knows to do so if the wander flag is not 0. The beginning of the wall following algorithm assumes the robot has hit a wall and is now parallel to it or is already following a wall. The robot moves forward to follow the wall until either the wall disappears, it hits a wall, or finds the fire.

If the robot hits a wall, it calls adjust1, which backs the robot up and turns left.

If the robot finds a fire it stops and signals fire has been found.

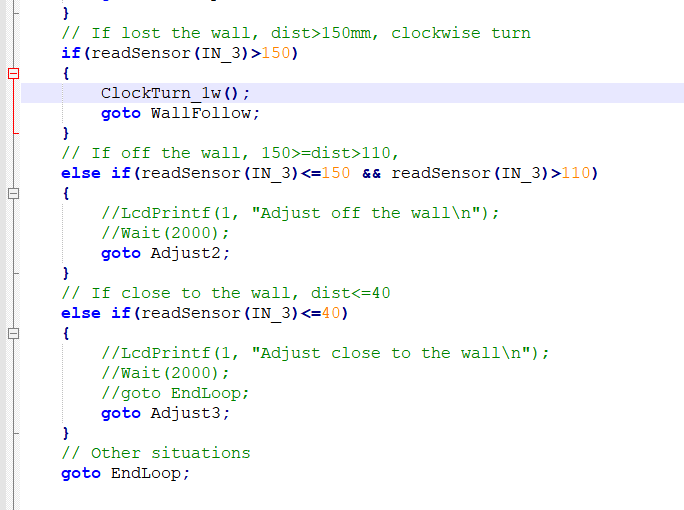
Otherwise the robot moves forward until either its been a while, it finds a fire, hits a wall, and the sonar isn’t displaying an error.

Here is what happens after the above cases in wall following:

If the robot hits a wall, it calls adjust1, which backs the robot up and turns left.

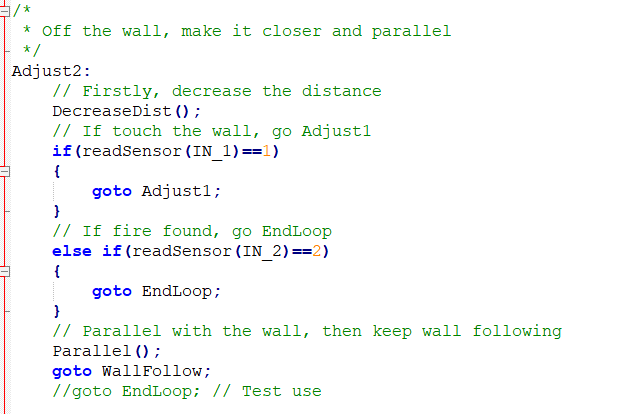
If the robot finds a fire it stops and signals fire has been found.

If the sonar begins to display an error, it is no longer parallel to the wall, so it needs to re find the wall.

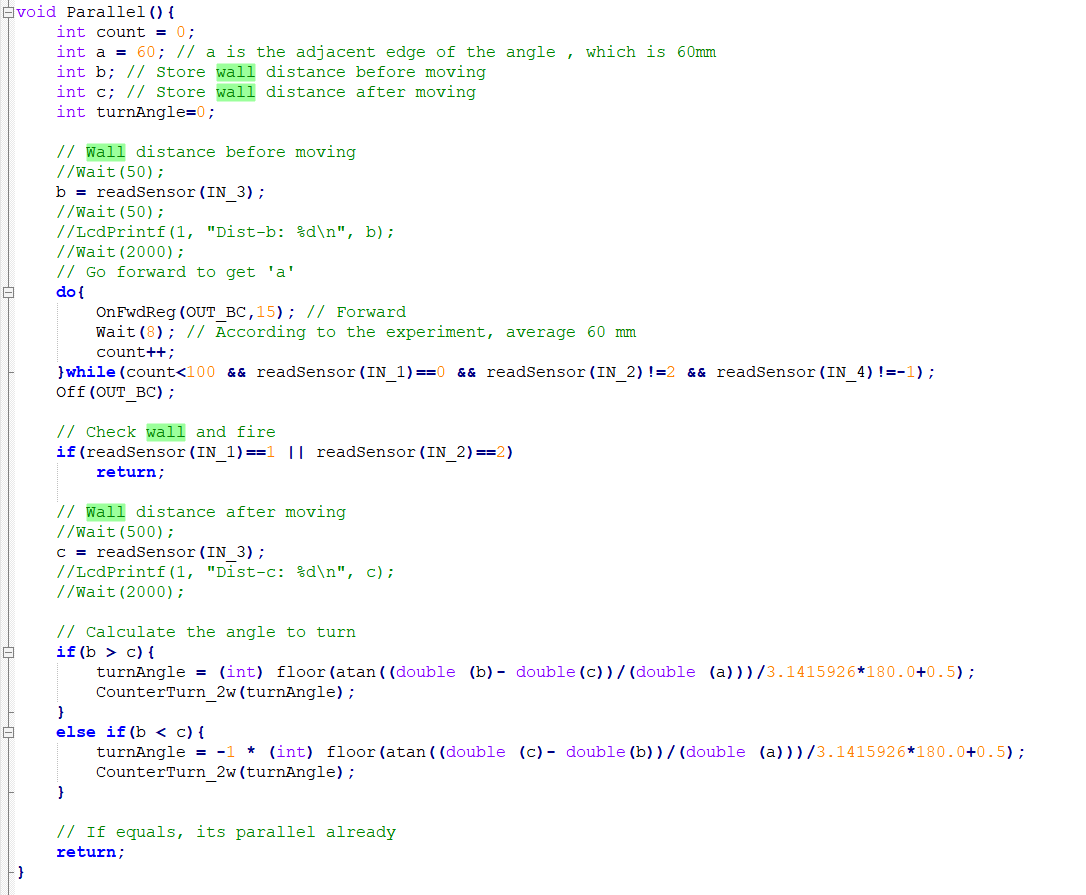


There are 3 cases.

1. The ClockTurn\_1w function is called if the Sonar detects a long distance on the right of the robot while following the wall. The robot has found a hallway, so it turns right moves forward and begins wall following on the new hallway.
2. The robot has wandered a little bit too far from but is still close enough, so it swivels and readjusts to be parallel again via the Adjust2 function.



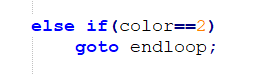
It does so via the Parallel function and a return to wall following. The parallel function assures the robot stays the same distance from the wall to ensure accurate sonar readings, via trigonometric pre calculated movement.

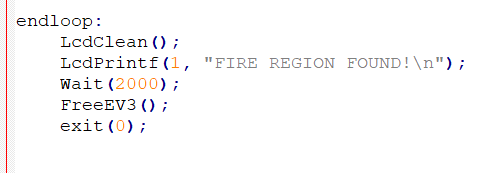


3. The robot is too close to the wall to get an accurate sonar reading it calls Adjust3, which swivels away from the wall to get parallel with it again, but further out. Once parallel, it returns to wall following.

**Section 3: Fire detection**

The robot handles fire detection via a color sensor facing down on the front of the robot. Since the fire is always on a blue sheet of paper, if the robot moves over that sheet of paper, it signals on the nxt it found the fire and stops. It checks whenever it moves, during the control sequence of the robot whether it has run over a fire or not. If so, it calls the exit function and displays Fire region found!





There is no handling of extinguishing the fire in our code.

**Part 4: Fire Distinguish**

Our solution to distinguish the fire is to use a USB Fan blow off the fire. The picture of the fan is shown as below:



However, we didn’t put this on our robot in the presentation. The problem is that, it doesn’t have a on-off switch, and it is USB driven. Which means, once it is plugged into the USB port of the robot, it will automatically start by itself. It runs out the battery fast, and might affect the functioning of the robot before it detects the fire region.

A solution we’ve came up with is that, we can attach the USB plug of the fan with a motor, and facing it right in front of the robot USB port. After the fire region is found, rotate the motor and plug the fan in. Then make the robot move around the edge of the fire region to blow off the fire.

**Part 5: Problems Met and Solutions**

1. **Motor never stops**

Since we’ve used a lot do-while loops and Wait() functions in the loop, there are situations that the motors don’t stop running even if the loop is finished.

Solution is to manually turn off the motors after each loop. By calling the function Off(OUT\_PORT) defined in the EV3API.

1. **Oversteer and Understeer**

Since we are using gyro to check the angles of the robot turns, it always cause the overturn when clockwisely turning, or understeer sometimes as well.

There are no really good solutions, the way we fixed it is by experimentally test the angles and minus or plus some integer values to compensate with the gyro readings.