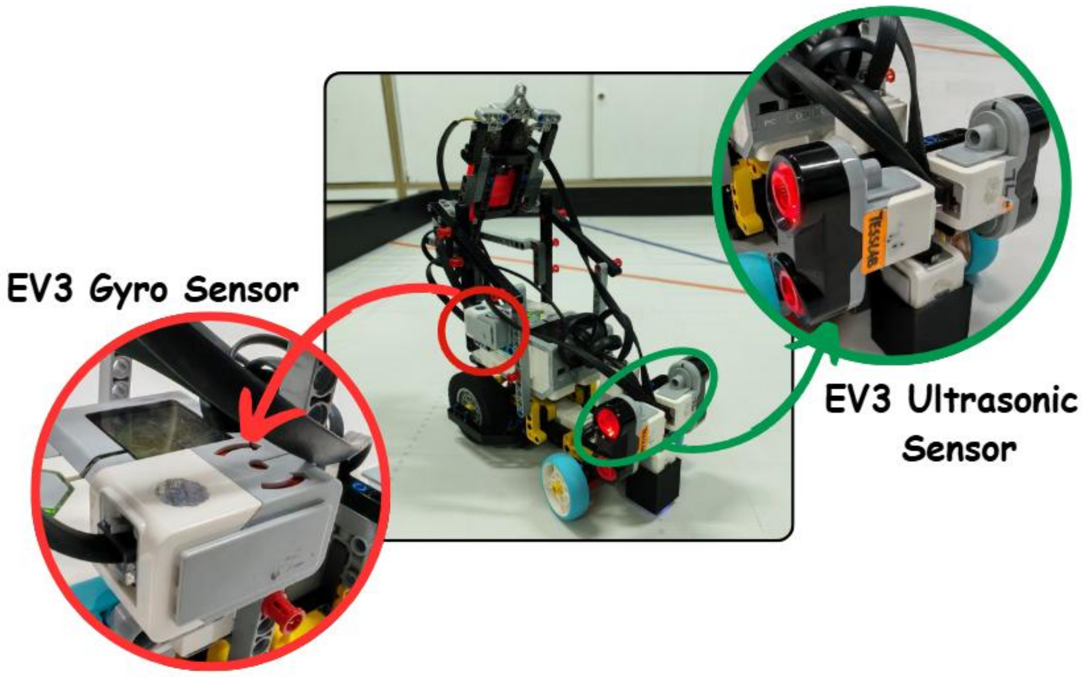
3.1 Open Challenge

To complete the open challenge, our robot uses two main sensors which is **an EV3 Gyro Sensor** and **two EV3 Ultrasonic Sensors** on both the left and right sides. Since both **EV3 Ultrasonic Sensors** share the same port, we use a **multiplexer** to allow the robot to read data from each one separately. The left ultrasonic sensor is connected to Channel 1 of the multiplexer, while the right ultrasonic sensor is connected to Channel 3.



The **EV3 Ultrasonic Sensors** helps the car to measure the distance between the robot and the walls. By comparing readings from the left and right sensors, the robot can determine whether it should move **clockwise** or **counter clockwise**. These readings also help the robot adjust its steering to avoid collisions and stay at a safe distance from inner and the outer walls. Additionally, the **EV3 Gyro Sensors** monitor the robot's heading, ensuring it maintains a straight path and performs accurate turns at the correct angle when turning.

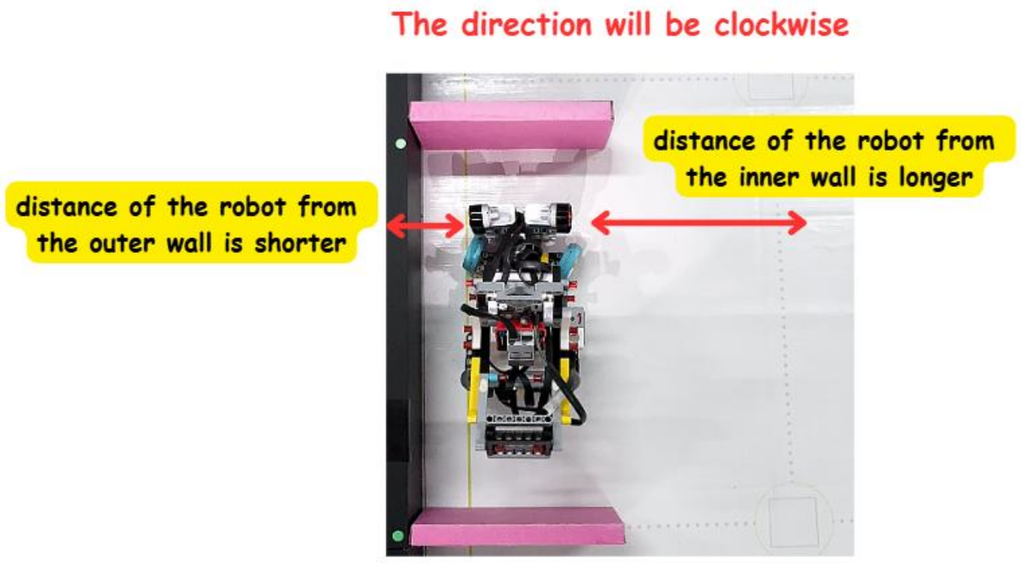
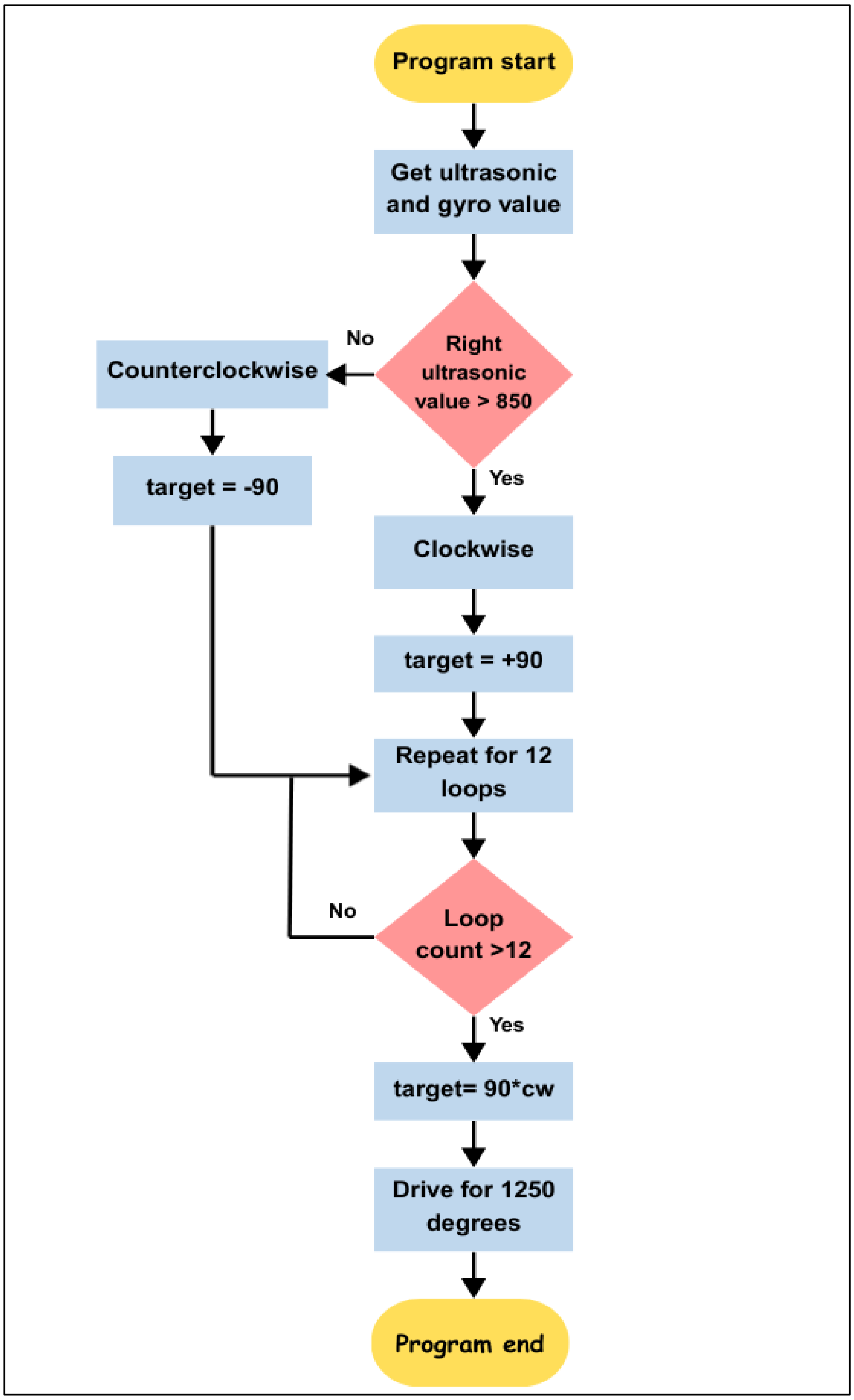

The direction will be clockwise
distance of the robot from the inner wall is longer
distance of the robot from the outer wall is shorter

Diagram below shows the flowchart for Open Challenge:

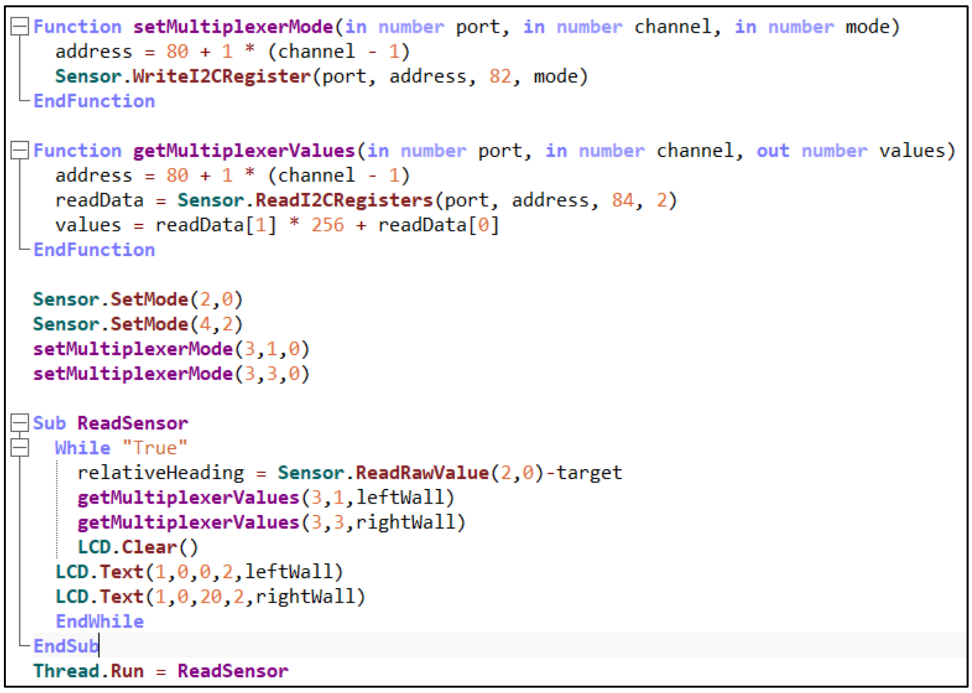


Appendix 2 contains the complete programming code for the **Open Challenge.**

Below is a brief explanation of the key parts of the code used to control the robot during the challenge.

1. **Initialization and Setup**

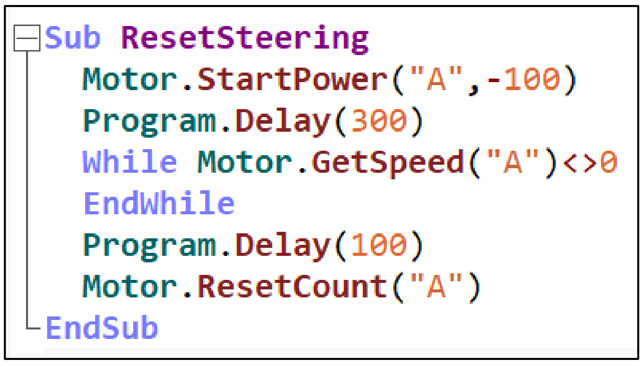
   This part is aim to set up the sensors and multiplexer before running. The **setMultiplexerMode function** allows the EV3 to connect with both ultrasonic sensors in the same port while the **getMultiplexerValues function** allows the robot to read the distance values from the left and right ultrasonic sensors. Additionally, the **ReadSensor** continuously updates the robot's heading and wall distances in real time.



```
Function setMultiplexerMode(in number port, in number channel, in number mode)
  address = 80 + 1 * (channel - 1)
  Sensor.WriteI2CRegister(port, address, 82, mode)
EndFunction

Function getMultiplexerValues(in number port, in number channel, out number values)
  address = 80 + 1 * (channel - 1)
  readData = Sensor.ReadI2CRegisters(port, address, 84, 2)
  values = readData[1] * 256 + readData[0]
EndFunction

Sensor.SetMode(2,0)
Sensor.SetMode(4,2)
setMultiplexerMode(3,1,0)
setMultiplexerMode(3,3,0)

Sub ReadSensor
  While "True"
    relativeHeading = Sensor.ReadRawValue(2,0)-target
    getMultiplexerValues(3,1,leftWall)
    getMultiplexerValues(3,3,rightWall)
    LCD.Clear()
  LCD.Text(1,0,0,2,leftWall)
  LCD.Text(1,0,20,2,rightWall)
  EndWhile
EndSub
Thread.Run = ReadSensor
```

2. **Reset Steering**

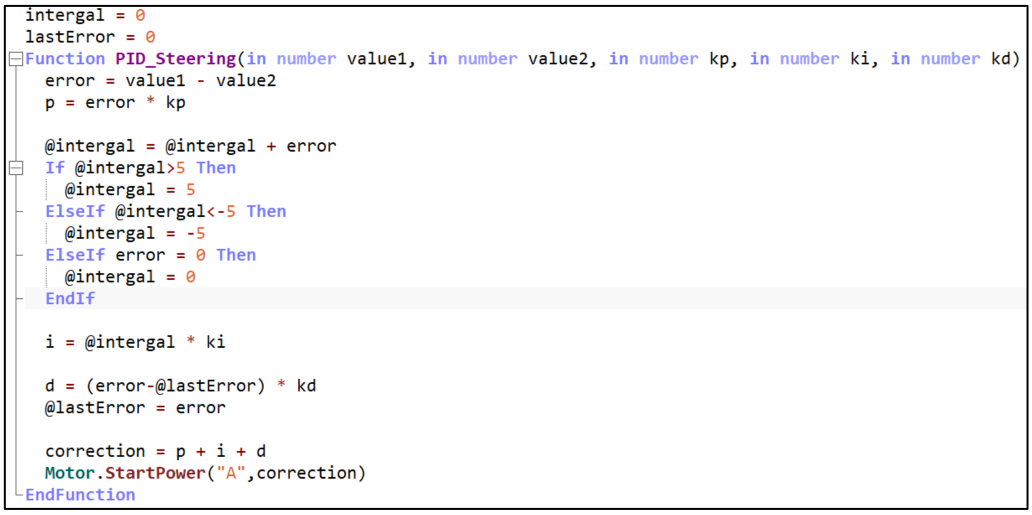
   Reset Steering make sure the steering motor starts at a fixed desired position before driving. The robot turns the steering motor all the way to the left for **300 milliseconds** by using the **medium motor in the port A**. Hence, it waits until the motor completely stops moving, which means it has hit the limit. A short pause is added to let the medium motor. After that, the motor's rotation counter is reset to zero. This step is important because it tells the robot exactly where the "straight ahead" position is.



```
Sub ResetSteering
   Motor.StartPower("A",-100)
   Program.Delay(300)
   While Motor.GetSpeed("A")<>0
   EndWhile
   Program.Delay(100)
   Motor.ResetCount("A")
EndSub
```

3. **PID Steering Control**

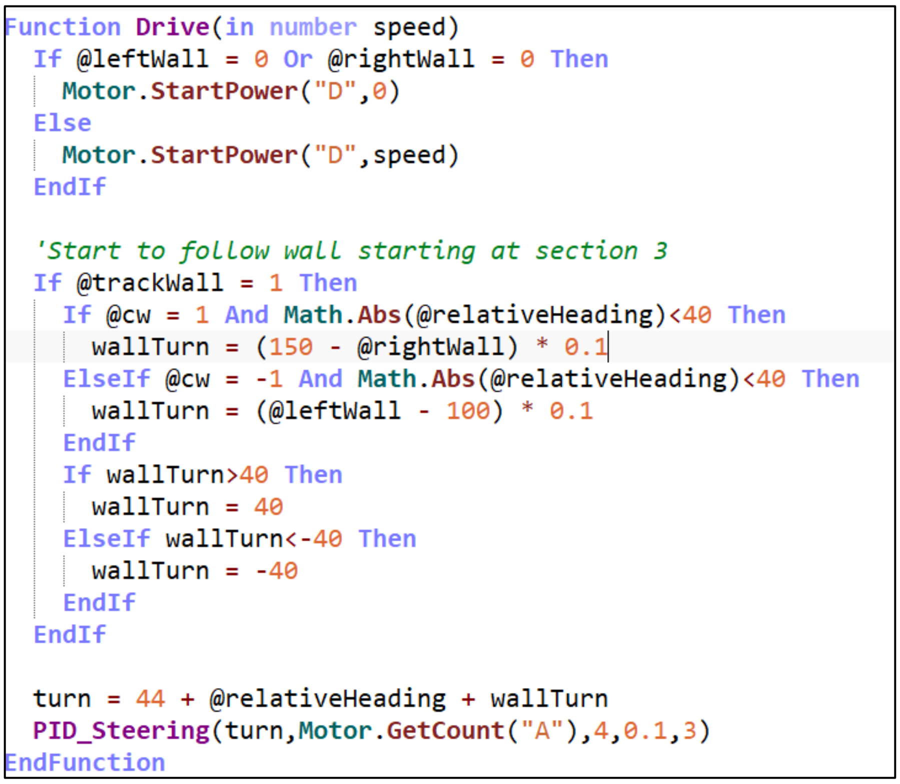
   The **PID Steering function** helps the robot to adjust the steering motor whenever it starts to drift off from the target. The **P** part fixes most of the error right away, the **I** part makes small adjustments over time and the **D** part make sure that the correction is done smoothly. These three adjustments are added together to get one correction value, which is sent to motor A to turn the steering. This keeps the robot moving smoothly, avoiding zig-zagging, and makes its turns more accurate.



```
intergal = 0
lastError = 0
Function PID_Steering(in number value1, in number value2, in number kp, in number ki, in number kd)
  error = value1 - value2
  p = error * kp

  @intergal = @intergal + error
  If @intergal>5 Then
    @intergal = 5
  ElseIf @intergal<-5 Then
    @intergal = -5
  ElseIf error = 0 Then
    @intergal = 0
  EndIf

  i = @intergal * ki

  d = (error-@lastError) * kd
  @lastError = error

  correction = p + i + d
  Motor.StartPower("A",correction)
EndFunction
```

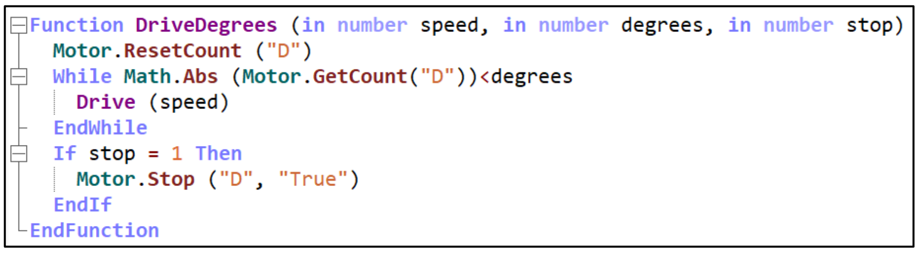
4. Drive

The **Drive** function moves the robot forward and keeps it aligned when following a wall. If no wall is detected on either side, the drive motor stops; otherwise, it runs at the set speed. When wall-following mode is on, the robot adjusts its steering based on the distance to the wall. It then combines this adjustment with its current heading to get a target steering angle, which is sent to the **PID Steering function** for smooth and accurate control.



```
Function Drive(in number speed)
  If @leftWall = 0 Or @rightWall = 0 Then
    Motor.StartPower("D",0)
  Else
    Motor.StartPower("D",speed)
  EndIf

  'Start to follow wall starting at section 3
  If @trackWall = 1 Then
    If @cw = 1 And Math.Abs(@relativeHeading)<40 Then
      wallTurn = (150 - @rightWall) * 0.1
    ElseIf @cw = -1 And Math.Abs(@relativeHeading)<40 Then
      wallTurn = (@leftWall - 100) * 0.1
    EndIf
    If wallTurn>40 Then
      wallTurn = 40
    ElseIf wallTurn<-40 Then
      wallTurn = -40
    EndIf
  EndIf

  turn = 44 + @relativeHeading + wallTurn
  PID_Steering(turn,Motor.GetCount("A"),4,0.1,3)
EndFunction
```

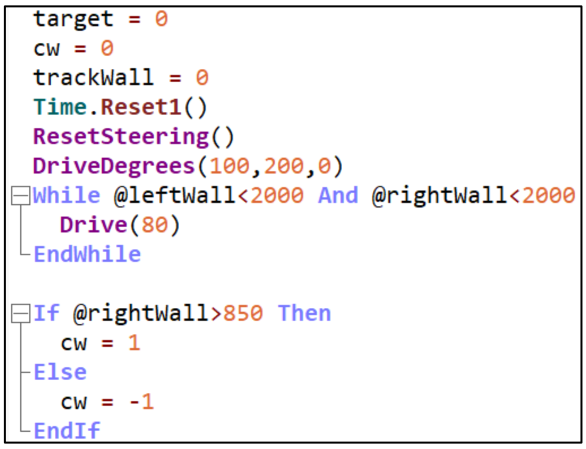
The **DriveDegrees function** moves the robot forward in rotation degrees. He robot will keep driving at the given speed until the target degrees are reached, and stops the motor if stop is set to 1.



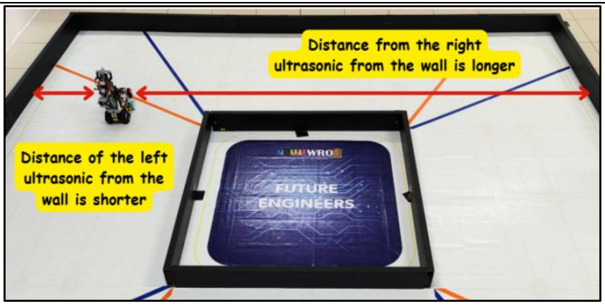
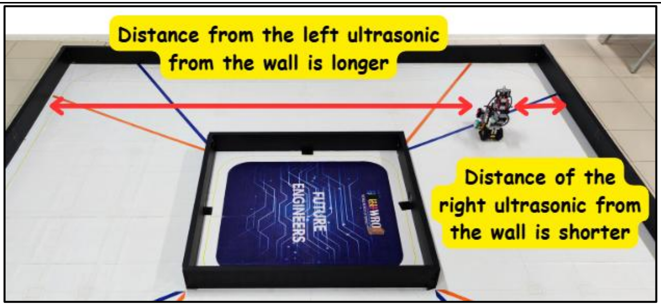
```
Function DriveDegrees (in number speed, in number degrees, in number stop)
  Motor.ResetCount ("D")
  While Math.Abs (Motor.GetCount("D"))<degrees
    Drive (speed)
  EndWhile
  If stop = 1 Then
    Motor.Stop ("D", "True")
  EndIf
EndFunction
```

5. Main program

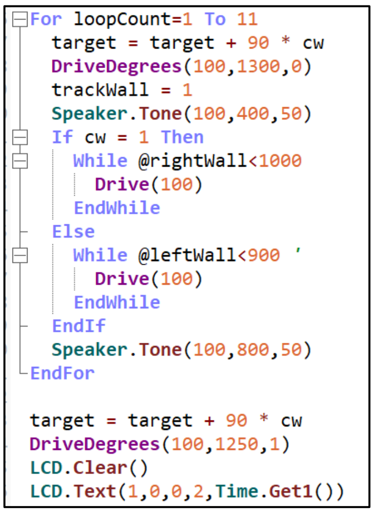
This program resets timers and steering, then drives forward until it detects a wall on either side. It determines whether to follow the right or left wall based on sensor readings.



```
target = 0
cw = 0
trackWall = 0
Time.Reset1()
ResetSteering()
DriveDegrees(100,200,0)
While @leftWall<2000 And @rightWall<2000
  Drive(80)
EndWhile

If @rightWall>850 Then
  cw = 1
Else
  cw = -1
EndIf
```

 Distance from the right ultrasonic from the wall is longer Distance of the left ultrasonic from the wall is shorter	 Distance from the left ultrasonic from the wall is longer Distance of the right ultrasonic from the wall is shorter
The robot will move in clockwise direction.	**The robot will move in counter clockwise direction.**

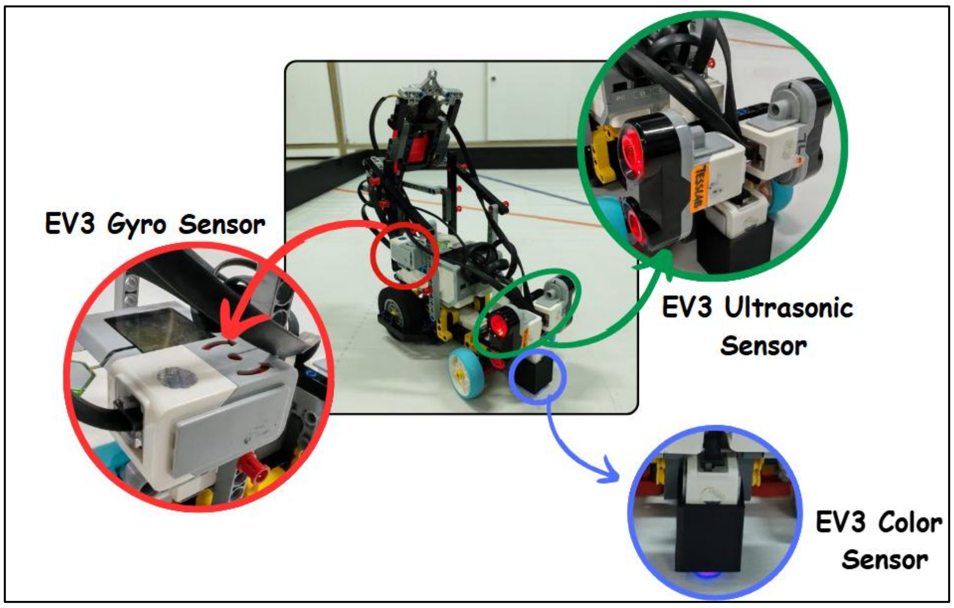
Then, it loops for 11 times, turning 90° in the chosen direction in every loops. At the end, it makes one final 90° turn, drives forward, and displays the elapsed time on the LCD.

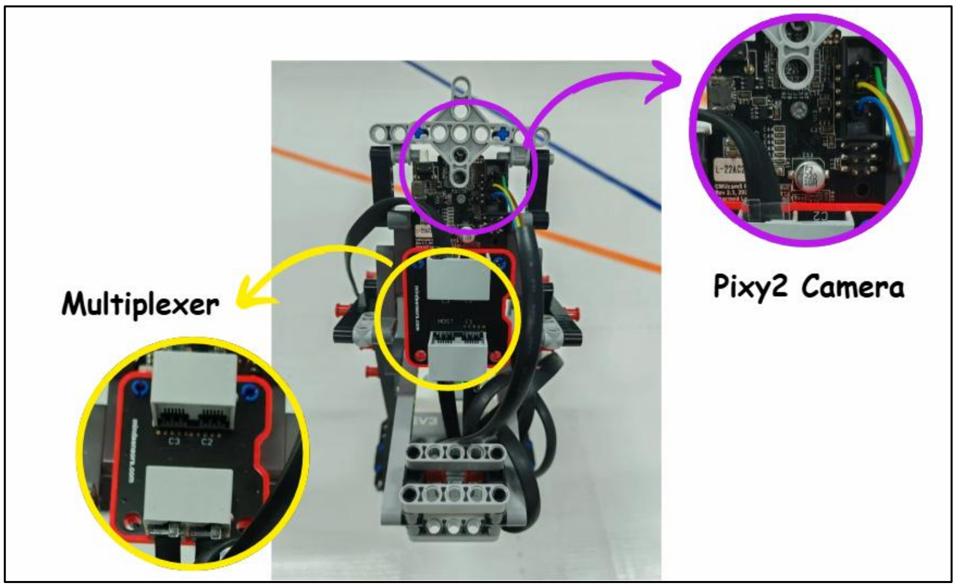


```
For loopCount=1 To 11
  target = target + 90 * cw
  DriveDegrees(100,1300,0)
  trackWall = 1
  Speaker.Tone(100,400,50)
  If cw = 1 Then
    While @rightWall<1000
      Drive(100)
    EndWhile
  Else
    While @leftWall<900 '
      Drive(100)
    EndWhile
  EndIf
  Speaker.Tone(100,800,50)
EndFor

target = target + 90 * cw
DriveDegrees(100,1250,1)
LCD.Clear()
LCD.Text(1,0,0,2,Time.Get1())
```

3.2 Obstacle Challenge

To complete the **Obstacle Challenge**, our robot utilizes four sensors which is **a Pixy2 Camera**, **a gyro sensor**, **two ultrasonic sensors on** both the left and right sides **and a colour sensor.** Since both ultrasonic sensors need to be connected to the same port, we use a **multiplexer** so that the robot can read data from both of the sensors.





The **Pixy2 Camera is connected to port 1 and it helps to detect the presence of pillars and magenta parking lot during the round. Next, the EV3 Gyro Sensor that connected to port 2 helps to ensures that the robot does the accurate turns and keep the robot's heading straight during the round.** In addition, both of the **EV3 Ultrasonic Sensor** is connected to EV3 Brick port 3 via the multiplexer. They help to detect the presence of the inner and outer wall and prevent the robot from hitting into the wall. Lastly, we utilize the

EV3 Colour Sensor to identify the coloured line on the map which is the orange and blue line. The identified colour allows the robot to know the direction accurately. For instance, if orange line is scanned first, the direction will be in clockwise direction.