VEX Virtual Robotics Unit Summative Assignment

Mark Chen

CLICK HERE FOR CODE

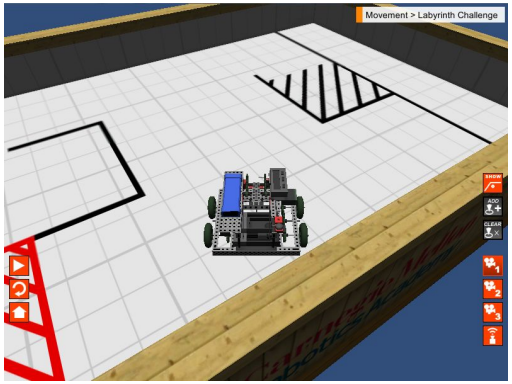
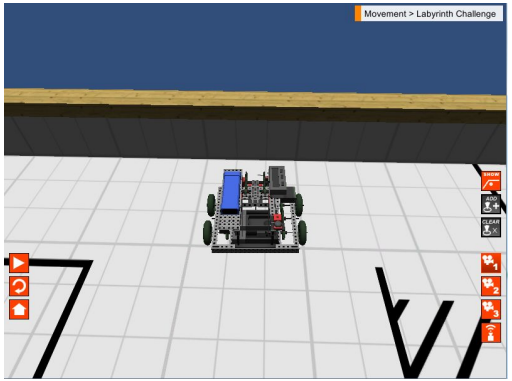
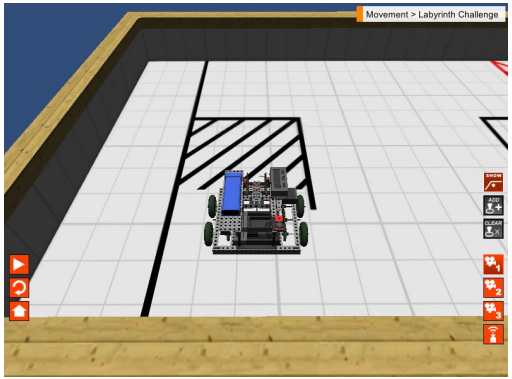
For some reason, the formatting when viewed in various text editors becomes messed up (especially the tab-spacing for documentation).

General Preface of Commonly Used Functions

Here are 4 functions that move the robot forwards a number of “*int blocks*” blocks, make a 90 degree turn to the left or right, and lastly a function that moves the arm above the robot’s frame so the arm is not in the way.

```
//This function will make the robot move straight forwards "int bl
void forwards(int blocks){
  motor[lMotor]= 127;
  motor[rMotor]= 127;
  wait1Msec(blocks*200/5.3*10);
}

//This function will make the robot turn 90 degrees to the left
void left(){
  motor[lMotor]= -127;
  motor[rMotor]= 127;
  wait1Msec(630);
}
//This function will make the robot turn 90 degrees to the right
void right(){
  motor[lMotor]= 127;
  motor[rMotor]= -127;
  wait1Msec(630);
}

//This function will move the arm up and out of the way
void armMove(){
  motor[armMotor] = -127;
  wait1Msec(100);
}
```

Screenshots of Labyrinth Challenge

		
The robot is making its first left turn after a specific amount of distance from the start position.	The robot moves forwards closer to the finishing area and is about to turn to the right.	The robot is approaching the end of the exercise and moves forwards towards the end of the maze.

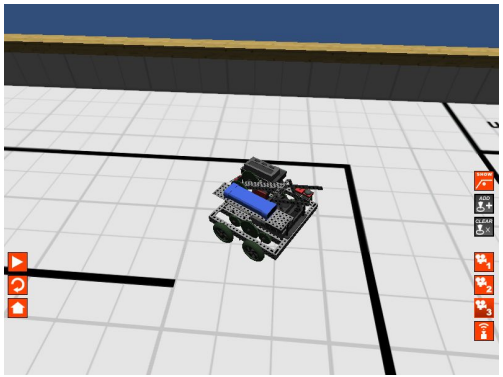
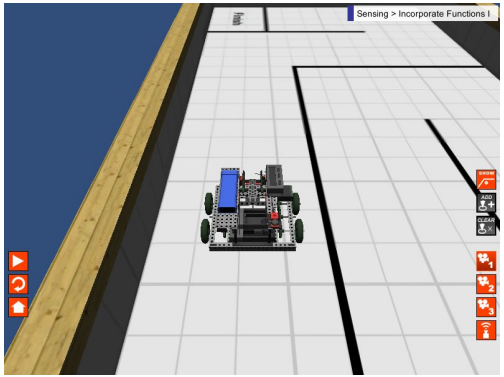
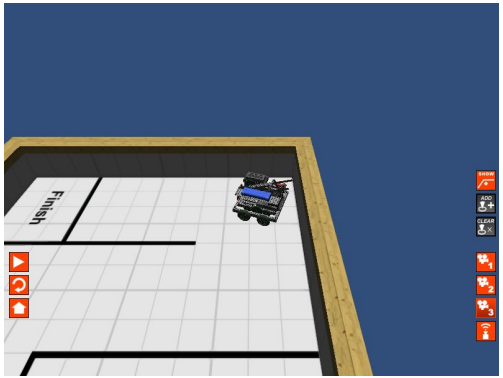
Code of Labyrinth Challenge

The first thing the robot does is moving the arm upwards and away from its front so the robot is more flexible to move and turn.

Next, the robot follows a series of instructions telling it to make a left or right turn, or to advance forwards for a given interval of time. Those functions will allow the robot to complete the maze.

```
task main()
{
  //Get rid of arm
  armMove();

  //Navigate through the maze
  forwards(2);     //Forwards for 2
  left();          //Turn left
  forwards(2);     //Forwards again
  right();         //Turn right
  forwards(1);     //Etc.
  right();
  forwards(1);

}
```

Screenshots of Incorporate Functions I

		
The robot has completes its first set of turns, pretty much making a U-turn, and is about a quarter of the way done the maze.	The robot is speeding straight forwards on this long straight stretch of the maze.	The robot is almost done the maze, and makes its last turn before moving to the finish zone.

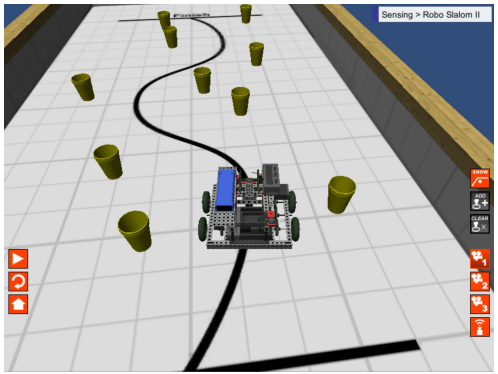
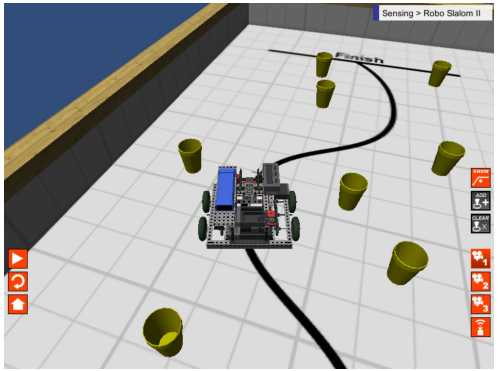
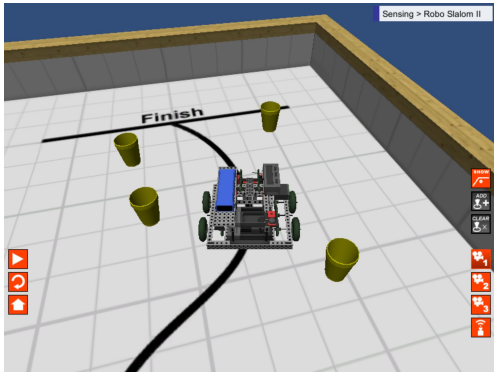
Code of Incorporate Functions I

Again, the robot moves the arm upwards for more mobility.

Again, the robot follows left, right, or forwards instructions that make the robot turn 90 degrees to the left or right, or advance a given number of blocks forwards. These instructions will allow the robot to go through the maze-like challenge.

```
task main()
{
  //Rid of arm
  armMove();

  //Advance, turn, and keep moving in the maze

  forwards(11);   //Forwards 11 blocks
  left();         //Turn left
  forwards(4.5);  //Forwards 4.5 blocks
  left();         //Turn left
  forwards(11);   //Etc.
  right();
  forwards(4);
  right();
  forwards(16);
  right();
  forwards(8);
  left();
  forwards(4);
  left();
  forwards(9);
}
```

Screenshots of Robo Slalom II

		
The robot has lifted its arm up and is following the black line, passing in between the first two obstacles.	The robot is closely following the black line as a bend in the line is approaching.	Nearing the end, the robot is still following the line and weaves through the obstacles.

Code of Robo Slalom II

We have two global variables for the sensors readings

Again, the robot's arm is moved for more mobility.

There is an infinite loop that detects what the left and right light sensors' readings are. Based off those readings, the robot will move towards the left or right, becoming more centred on the black line. This process is repeated over and over again, and the end result is a line following algorithm.

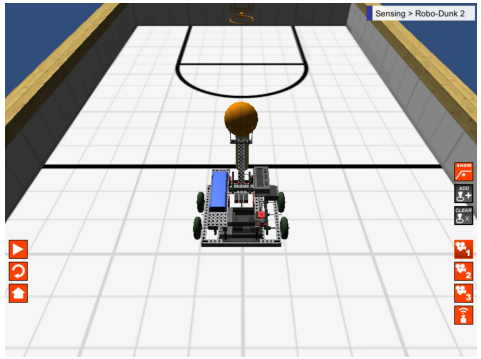
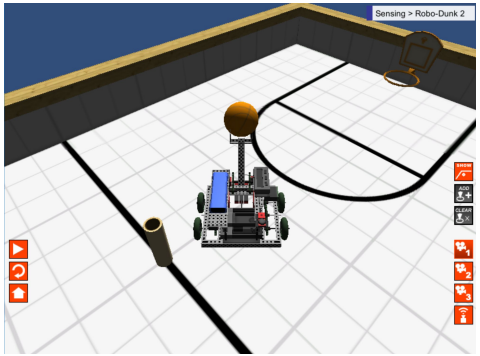
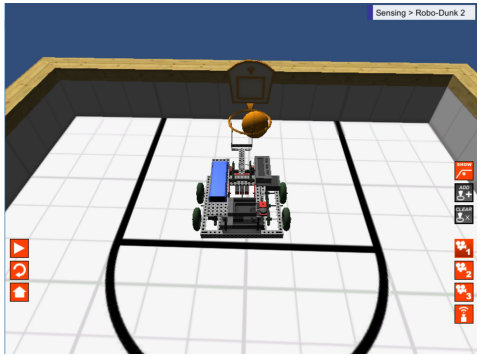
```
//Global variables for the return values of
int senseL = 0;
int senseR = 0;

task main()
{
  //Get rid of the arm to avoid potential ob
  armMove();

  //Keep following the line forever
  while(true){
    //Update the values of the sensors
    senseL = SensorValue[lSensor];
    senseR = SensorValue[rSensor];

    //Check to see if the left side is darke
    if(senseL > senseR){
      //If the left side is darker than the
      motor[rMotor] = 100;
      motor[lMotor] = 10;

    } else {
      //Otherwise, move towards the right
      motor[rMotor] = 10;
      motor[lMotor] = 100;
    }
  }
}
```

Other important functions

Brief descriptions of functions are in green
Function naming can also describe use and role

```
//This function will make the robot travel backwards for a set amount of time by the parameter
void backwards(int time){
  motor[lMotor]= -100;
  motor[rMotor]= -100;
  wait1Msec(time);
}

//This function will make the robot "scoot" forwards a little bit because the forwards function is not flexible enough
void scoot(){
  motor[lMotor]= 60;
  motor[rMotor]= 60;
  wait1Msec(500);
}

//This function will move the arm upwards and the stop the arm movement after some time to pick up an object
void pickUpBall(){
  //Slightly move forwards
  motor[lMotor]= 40;
  motor[rMotor]= 40;

  //Move arm up, and stop arm
  motor[armMotor] = -80;
  wait1Msec(500);
  motor[armMotor] = 0;
}

//This function is the opposite of the "pickUpBall" function, and will revert the arm back to its normal position downwards
void armDown(){
  //Slightly move forwards
  motor[lMotor]= 40;
  motor[rMotor]= 40;

  //Move arm down, and stop arm
  motor[armMotor] = 80;
  wait1Msec(650);
  motor[armMotor] = 0;
}
```

Screenshots of Robo Dunk II

		
The robot senses for the location of the ball and approaches the pole. When the robot is close enough, its arm lifts upwards, picking up the ball.	The robot turns and maneuvers around the pole where the ball used to be. It is turning to face the hoop.	The robot advances forwards and when it reaches the right distance from the hoop, the arm is lowered. The ball goes into the hoop, and the robot has completed its “dunk”.

Code of Robo Dunk II

There is a global variable for sonar input values.

First, we approach the pole with the ball and then we stop approaching it when the arm is the right distance away to pick up the ball. The ball is then picked up.

Next, the robot moves backwards/forwards, and turns left/right to evade the pole that was holding the ball. This code positions the robot to face the basketball net without any obstacles in between.

```
//Global variables for sonar's readings
int sonarVal = 100;

task main()
{
  //Get the ball from the pole

  //While the robot is not the right dista
  while(sonarVal>13){
    scoot();                          //Scoot
    sonarVal=SensorValue[sonar];  //Check
  }

  //Pick up the ball
  pickUpBall();


  //Go around the pole holding the ball

  backwards(500); //Go backwards
  right();        //Turn right
  forwards(2);    //Go forwards to evade t
  left();         //Turn left to face the
  forwards(5);    //Advance down the court
  left();         //Turn left, towards the
  forwards(2);    //Advance back to the ce
  right();        //Turn right, to face th
```

More Code of Robo Dunk II

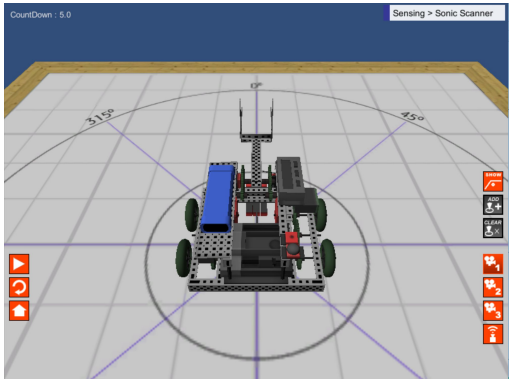
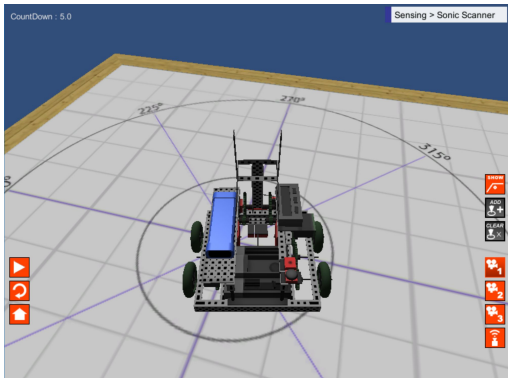
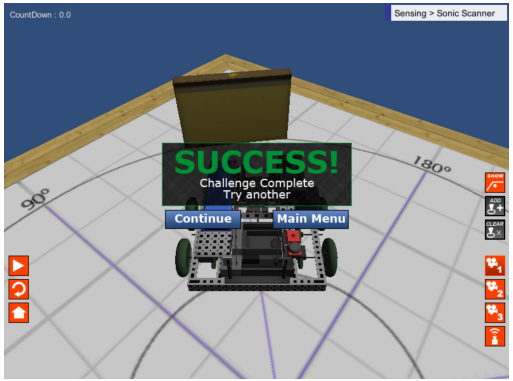
Now, we reset the global variable to something great, so we can reuse the variable to detect how far the robot is from the wall under the hoop.

The robot moves towards the net until it is the right distance away from the hoop.

When the robot is the right distance away, the robot's arm is lowered, and the ball can roll into the hoop, completing the challenge.

```
//Travel towards the hoop

//Reset sonar sensing variable
sonarVal=100;

//While the robot is the not the right d
while(sonarVal>37){
  scoot();                          //Scoot
  sonarVal=SensorValue[sonar];      //Check
}

//Move arm down to dunk
armDown();
```

Screenshots of Sonic Scanner

		
The robot lifts its arm and begins to turn towards the left.	The robot scans the area as it spins, covering every angle while searching for an obstacle to approach.	The robot finds a wall and stop spinning. The robot approaches the wall and stops.

Code of Sonic Scanner

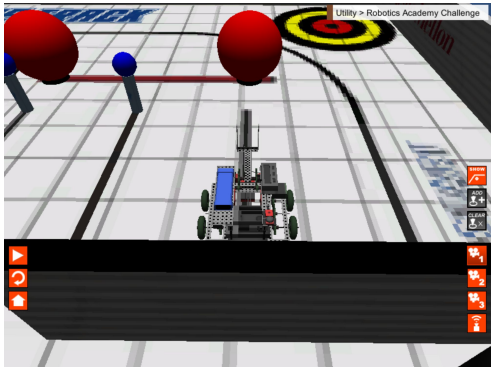
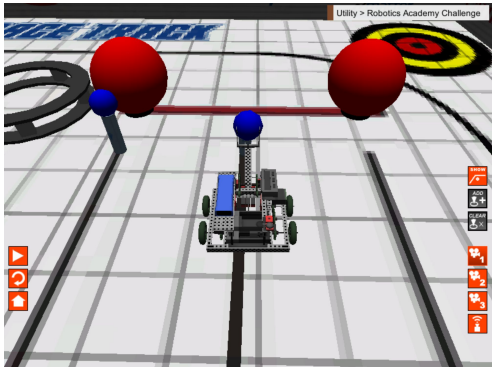
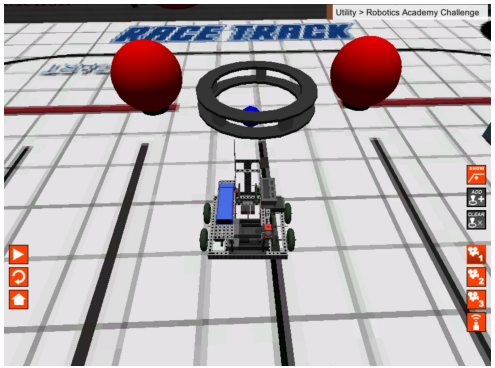
First, the robot will move its arm out of the way.

Until the sonar detects something, keep rotating towards the left and continually taking in the most up to date sonar sensing input value. When something is detected, the loop terminates. This will leave the robot directly facing the obstacle.

When the robot detects something in front of it, it stops spinning, and moved straight towards what the sonar sensed. Then, it stops.

```
task main()
{
  //Get rid of arm
  armMove();

  //Initialize a boolean declaring that the obstac
  bool found = false;

  //While the object is not found, keep searching
  while(!found){
    //Check the sonar for an object in front of it
    senseVal = SensorValue[sonar];

    //If an obstacle is picked up by the sonar, th
    if(senseVal!=-1){
      found = true;
    } else {
    //Otherwise, keep rotating to the left, scanni
      left();
    }
  }

  //Move forwards 3 units towards the obstacle
  forwards(3);
}
```

Screenshots of Robotics Academy Challenge

		
The robot finds the first black line and checks to see if there is a pole. The robot's sonar does not detect a pole, so it moves on.	Finding the second line, the robot sees a pole on the second line so it follows the line. It slowly picks up the second ball.	The robot locates the target area on the 4th line and then drops the ball into the target area.

Code of Robotics Academy Challenge

First, the robot moves forwards until it sees a black line. It turns on the black line and sees if there is a pole at the end of the line. If there is, then the robot will follow the line and then pick up the ball. If no pole is sensed, then the robot retracks backwards and continues to search for other places where a ball could exist.

This only works for getting a ball from any of the first 3 lines of the arena.

```
while(!gotBall){
  //While we are not at a good search
  while(!foundSearchablePlace){
    senseM = SensorValue[mSensor];  //

    //If the robot does not see a dark
    if(senseM<2000){
      //Keep moving forwards
      scoot(50);
      } else {
      //Otherwise you see a black line
      linesPassed+=1;    //Add the numb
      scoot(400);        //Adjust to tu
      right();           //Turn right t

      //A position where a ball might
      foundSearchablePlace = true;
    }

  }
  //Measure the distance of the first
  dist = SensorValue[sonar];

  //If the distance is bigger than whe
  if(dist>44){
    left();              //Turn back to
    forwards(1);         //Move forward

    //The robot is not at the place of
    foundSearchablePlace = false;

    } else {
    //There exists an obstacle, which
    backwards(500);      //Move backwar
    trackLine(2);        //Track and fo
    shake();             //Friction bet
    pickUpBall();        //Pick up the
    gotBall = true;      //Finally, we
```

More Code of R.A.C.

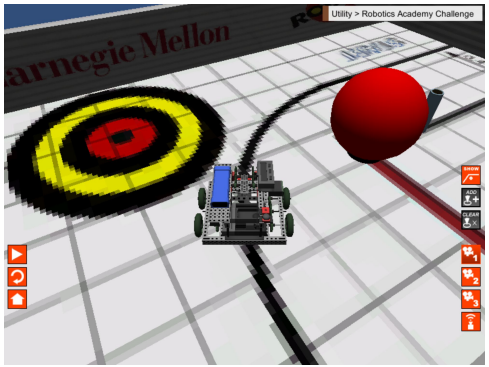
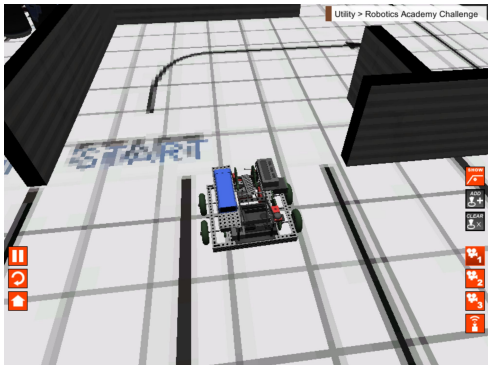
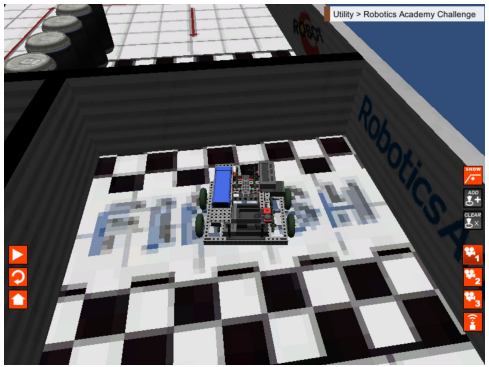
Next, the robot takes the ball away from the pole and moves back on course to find the dropping area. The robot will approach the target area until it senses that 4 black lines have passed. Once the target area is found, then the robot moves to the target area, and drops the ball into the hole.

Once again, the robot reroutes itself to face the end of the map, and it moves forwards until it is on top of the final black line, which is calculated using the distance from the end wall. Then, the robot follows the line back to the starting point of the program.

```
//This function will search for the t
void approachTarget(){
  //A variable to indicate if the targ
  bool checkpoint = false;

  //Continue forwards when the target
  while(!checkpoint){
    //If the target is reached, termin
    if(linesPassed==4){
      checkpoint = true;
      } else {
      //Otherwise, keep checking for r
      senseM = SensorValue[mSensor];
      //If the line is dark, add to th
      if(senseM>1100){
        linesPassed+=1;
        forwards(1);
      }
    }
    //Inch forwards slowly
    scoot(100);
  }
}
```

```
backwards(1000);          //Hav
left();                   //Fac
forwards(2);              //Mov


//Find the target area
approachTarget();

//After finding the target ar

right();                  //Tur
forwards(2);              //Mov
armDown();                //Dro
freeze();                 //Buf
motor[armMotor] = -100;   //Mov


//Part 2 - After dropping the

backwards(1500);  //Back up a
left();           //Face the
dist = 100;       //Reset the

//While the distance between
while(dist>40){
  //Continue to update the di
  dist = SensorValue[sonar];
  //Move forwards 1 block
  forwards(1);
}

backwards(300); //Back up ont
right();        //Turn right
trackLine(1);   //Follow the
```

More Screenshots of Robotics Academy Challenge

		
The robot is following the black line around the map, turning in the direction that is darker as indicated by the left and right sensors.	The robot is at the entrance of the final part of the challenge. The robot turns into the area, and will execute a series of turns and movements to reach the end point.	The robot completes the challenge and stops in the finish zone.

More Code of R.A.C.

Lastly, after the robot finishes the line tracking step to the beginning of the map again, the robot turns to the end point and then moves to the entrance of the final part.

At the entrance of the last part, the robot makes a series of left/right turns and forwards movements to get to the finishing point.

The End!!!

```
//Part 3 - Going to the end point
backwards(500); //Move away from t
right();        //Turn right to fa
forwards(21);   //Advance to the e
left();         //Turn towards the
forwards(10);   //Enter the ending
right();
forwards(8);
left();
forwards(6);
left();
forwards(10);
```