Aberystwyth Robotics Club - Elegoo Robot
Line Following

Introduction

In this worksheet we'll learn how to adjust how fast our robot moves and how to change behaviour based on conditions in order to follow a line.

Changing the speed

Previously we just used `digitalWrite()` to change the signals going to pins. Some of the pins on the arduino allow you to send Analog signals for more control. In these cases, instead of sending `HIGH` or `LOW`, we can send a number between 255 (HIGH) and 0 (LOW). We can use this on the pins that enable the motors to set how fast the motors turn.

1. Just above the function `void setup()` we are going to specify the initial speed to use

```
int carSpeed=150;
```

2. Inside the function `void setup()` we are going to change the `digitalWrites` to `analogWrites` and set the analog value to our speed.

```
analogWrite(LeftEnable, carSpeed); //Enable left motor
analogWrite(RightEnable, carSpeed); //Enable right motor
```

3. Run the updated code on your robot, trying out different values for carSpeed between 0-255.

Reading from the line sensor

1. As with the motors, we need to tell the Ardunio which pins are connected to the line tracking sensor on the underside of the robot. The line tracking sensor uses 3 pins for left, middle and right. These should be defined just below the motor pin defines at the top of our program:

```
#define LineSensorRight 10
#define LineSensorMiddle 4
#define LineSensorLeft 2
```

2. In the setup function, we initialised the motor pins as output pins. For the line sensors, we want to define these as input pins instead. We also want to set a serial rate for reporting the values read. Add the following to the setup method:

```
Serial.begin(9600);

pinMode(LineSensorRight, INPUT);
pinMode(LineSensorMiddle, INPUT);
pinMode(LineSensorLeft, INPUT);
```

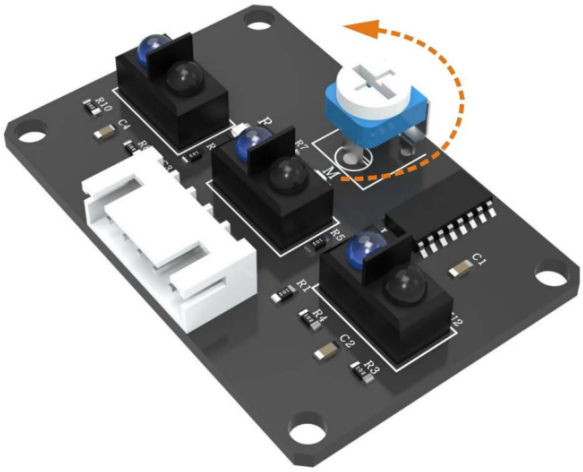
3. Below is an example function that will read the value from the left line tracking sensor and print the value in the Serial Monitor. Add this function at the end of your program, and then add two more for the middle and right line sensors.

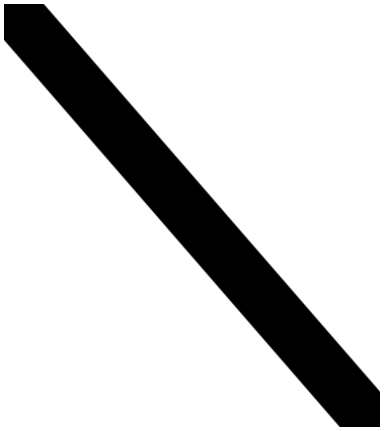
```
bool readLeft() { // define a function which returns a value
  bool reading;   // temporary variable
  Serial.print("Left ");  // print a message to the serial monitor
     so we know which sensor it is
  reading=digitalRead(LineSensorLeft);  // obtain a reading from the
      sensor, either 0 ir 1
  Serial.println(reading);  // print the reading alongside the
     message above, followed by a line break (println)
  return reading; // return the reading for use elsewhere
}
```

4. In the loop, we now need to call our new methods. You may want to comment out or remove the previous motor commands for this.

```
readRight();
readMiddle();
readLeft();
  Serial.println(); // add a blank line between sensor readings
delay(500); // pause before taking the next reading
```

5. Upload the program onto the robot and whilst still connected to computer, and the serial monitor open, move the line tracking sensor over the black line below to see the sensor values change. If it is not sensitive enough, you may need to adjust the potentiometer highlighted below.





Following a line

To follow the line, we will need to read the values from the line tracking sensor and make a decision about what action to perform. This can be done with an *if-then-else* statement. The typical structure of this is as follows:

```
if(CONDITION) {
  //THEN perform actions here
}
else if(CONDITION 2) {
  //THEN perform actions here
}
else {
  //If none of the above conditions are true, do this
}
```

Where the condition should evaluate to true (1) or false (0). In our case, the values from the line tracking sensors can be used directly as conditions.

With your partner, discuss a strategy for following a line using the line tracking sensor and the if-then-else statements. Some hints are given below, but try out different configurations first.

Have a race to see you can navigate the line the fastest. Penalties should be added for leaving the line.

Hint 1 Each time we go through the loop, we want to check the sensor readings and change direction if necessary.

Hint 2 Adjusting the speed of the robot may make it more accurate.

Hint 3 The order in which the checks are done can make a big difference in the if-then-else statement.

Hint 4 If a side sensor is returning `true`, we probably want to turn in that direction.

Hint 5 Changing direction too fast could result in the robot not moving at all. You may need to add a short pause in the loop after setting the direction.

Hint 6 When setting the direction as left or right, the settings for the other side are not changed. You may wish to completely stop the robot at the start of each loop, or stop the other side when changing direction.

Hint 7 An example structure might look like:

```
setStop();
if(readLeft()){
  ????
}
else if(readRight()){
  ????
}
else if(readMiddle()){
  ????
}
delay(20);
```