Project Aims

Over the course of the next few hours you will be learning how to make a line following robot. The aims of this project are:

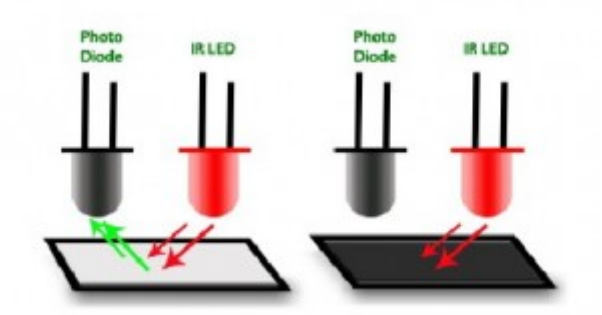
- Learn how a line following robot works
- Learn basic programming skills
- Learn basic Arduino skills
- Construct a line following robot

Pack Contents

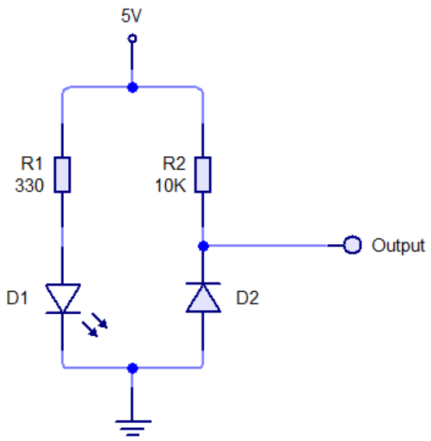
1x Arduino
1x Motor shield
1x 9v battery clip
1x 6xAA battery holder
6x AA Batteries
1x Base plate
2x Motor mounts
2x Rear axle mounts
2x Sensor holders
2x Geared motors
1x Axle shaft
3x Wheels
1x Long zip tie
8x Short zip ties
10x 20mm M3 bolts
4x 30mm M3 bolts
17x M3 nuts
1x Senor PCB
5x IR LEDs
5x IR Photodiodes
5x 330 Ohm resistors (Orange Orange Brown)
5x 10k Ohm resistors (Brown Black Red)

What is a line following robot and how does it work?

As you might have already guessed, a line follower is a robot that follows a marked line. In this project the goal will be to design a robot that can navigate a track with multiple turns made from black electrical tape. The way the robot will be able to detect its position relative to the track will be via an array of 3-5 Infrared (IR) reflectance sensors. These sensors consist of two parts, an IR LED and an IR photodiode. The LED emits light, which is then reflected by the surface into the photodiode:

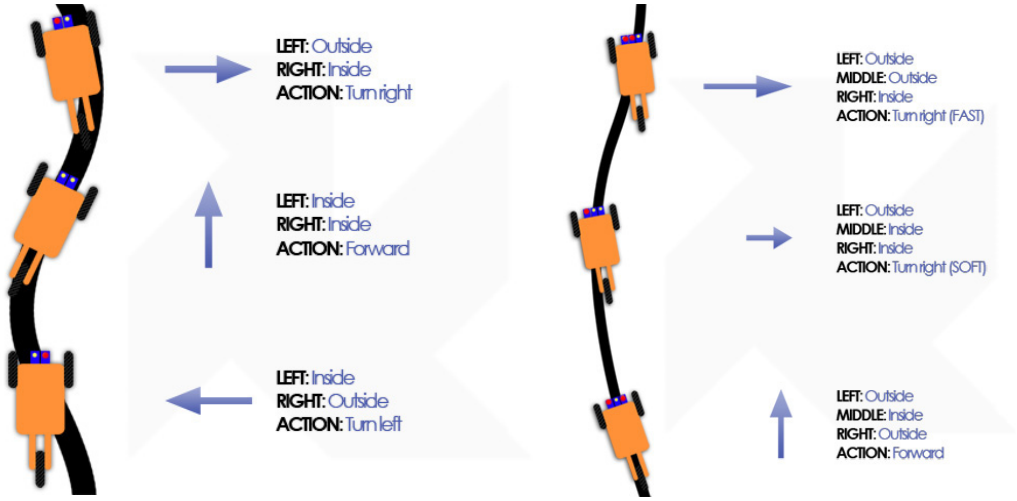


Depending on the colour of the surface the amount of light reflected will vary. In our case we will be using a white surface with a black line. As can be seen in the previous diagram, the black surface absorbs all the IR light while the white reflects it all. The photodiode will produce a current proportional to the amount of IR light it receives. Using this we can build the following circuit to detect whether the sensor is above a white or black surface:



While ideally the black line will absorb all IR emitted by the LED, in reality this isn't the case. The reflected IR and ambient light will mean that the sensor will output a small amount of current even on the black line. It will be your task to figure out how to solve this in your code later on.

The minimum number of sensors needed to successfully follow a line is two. One on either side of the track (you can get away with one sensor but its unreliable). Using two sensors tends to lead to a lot of wobbling so it is recommended to use three, one on the line and one either side.

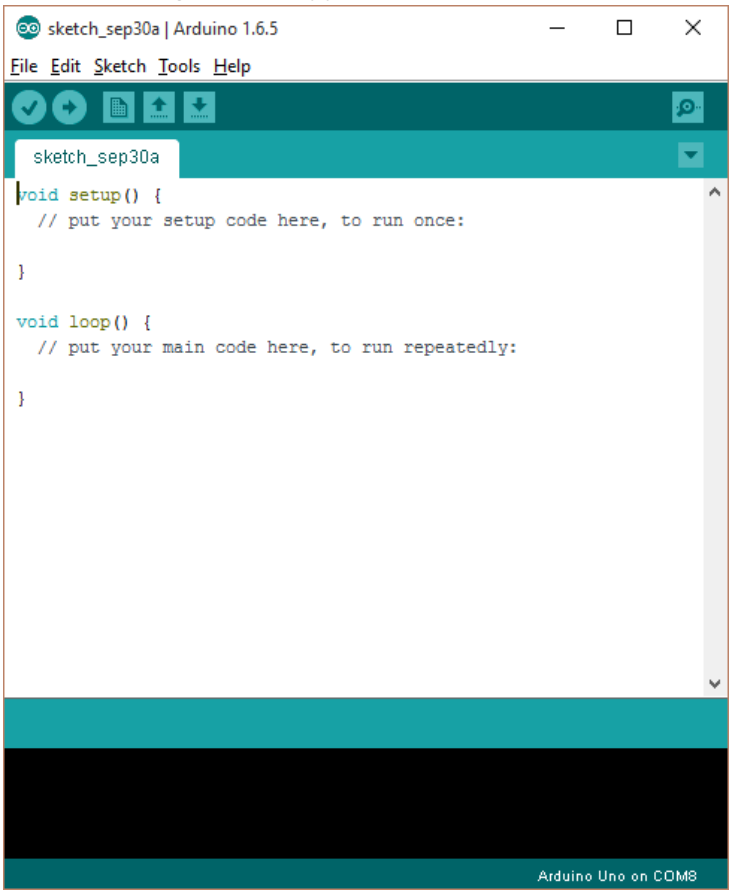


Once you have an array of sensors you can then steer the robot based on how much light each one receives.

Writing your first Arduino program

1. On the desktop you should find the following icon, double click it to open the Arduino software:

2. You should then have following window appear:



Please take note of the two functions called “setup()” and “loop()”. The setup function is run once when the Arduino starts, after this the loop function runs indefinitely. Also Arduino programs are called sketches.

3. If you look at the Arduino pinout diagram provided, you will notice it has 14 digital pins numbered 0 to 13. These pins can be set as either inputs or outputs. In this example we will set pin 13 to an output. Pin 13 is connected to an on-board LED. To do this simply add the following line of code into the setup function:

```
pinMode(13,OUTPUT);
```

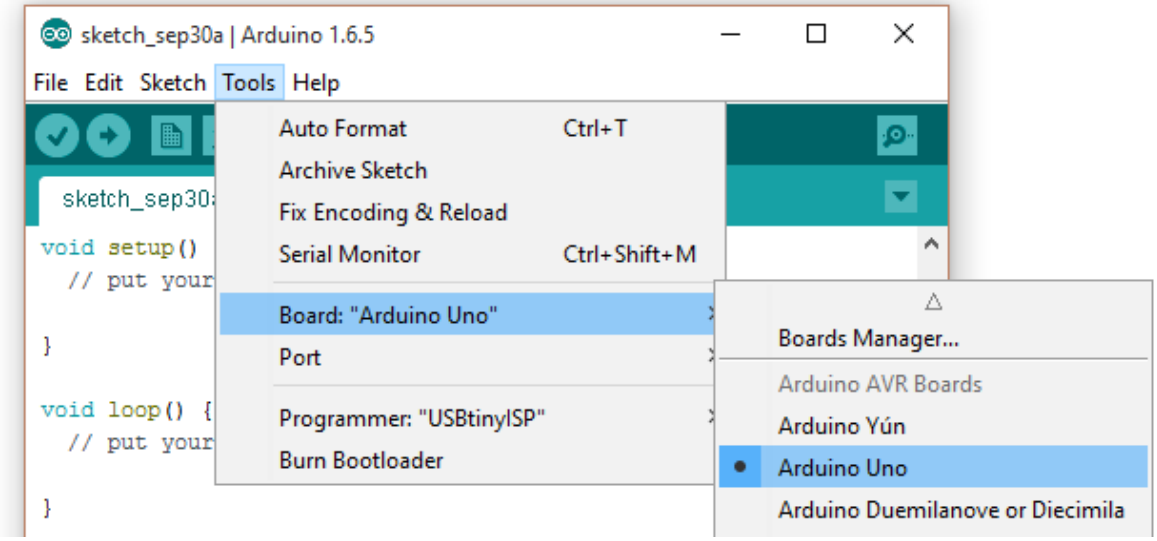
Code is case sensitive so make sure you copy it exactly as shown.

4. Now that pin 13 is set as an output, we can either output “HIGH” or “LOW”. Setting a pin HIGH will make it output 5 volts, while setting a pin LOW will make it output 0 volts.

To change the state of a pin we use the “digitalWrite” function. In this example we will blink the LED connected to pin 13 once a second (1000 milliseconds). Type the following code into the loop function:

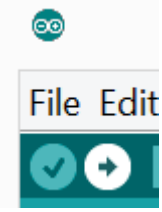
```
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
```

5. Now that we have written our code it’s time to upload it to the Arduino. The first thing we need to do is to make sure the Arduino IDE is set to the correct board type. To do this go to Tools>Board and ensure Arduino Uno is selected:



Next we need to ensure the Arduino IDE is using the correct COM port. Depending on your PC configuration there may be more than one. The COM port can be changed by going to Tools>Port. In the latest version of the Arduino IDE it should show you which COM port is the Arduino (If you have any problems just ask a demonstrator for help).

6. The last step is to actually upload the code to the Arduino. This can be done by pressing the arrow button at the top of the Arduino IDE:



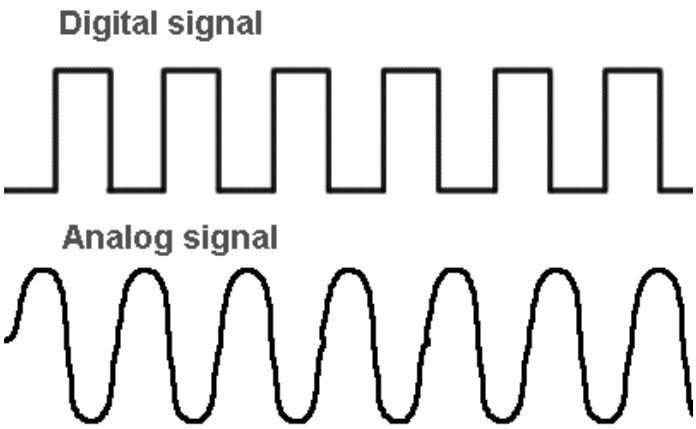
If everything has been successful, the bottom part of the window should say “Done uploading.” (If you get an error call over a demonstrator for assistance).

7. Once the uploading is complete you should notice an LED near the USB plug blinking roughly once a second.

8. See if you can change the speed at which it blinks, or make the Arduino blink a pattern.

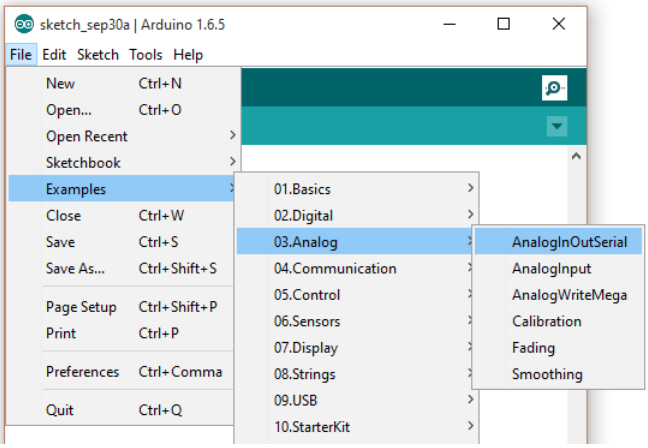
Reading analog inputs (You will need to have soldered your PCB to complete this section)

In the previous section we used the digital I/O (Input/Output) pins as outputs to blink an LED. In your robot these outputs will be used to drive motors. The next thing we need to learn is how use the analog inputs. In electronics you will find two types of signals, digital and analog:



Digital signals have only two states: HIGH/1 or LOW/0, information is sent as a series of 0s and 1s in digital signals. Analog signals on the other hand can have any number of states, therefore they cannot be expressed using only two values. The Arduino has special inputs for reading analog signals. These inputs have an analog to digital convert attached to them. The converter will turn a signal in the range of 0 volts to 5 volts into a value between 0 and 1024.

1. Once you have built the circuit open the Arduino IDE and go to: File>Examples>03.Analog>AnalogInOutSerial



This file contains some ready made code to allow you to send data to the PC.

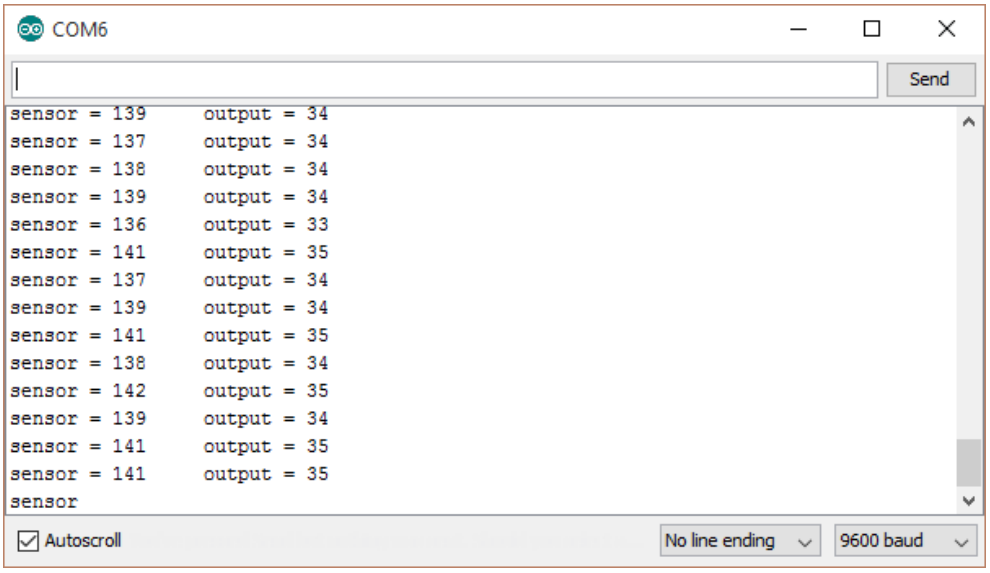
2. To read analog inputs we need to use the analogRead() function. This function takes an analog pin as a parameter. If you look on the Arduino, you will notice the analog pins are labelled A0 to A5. You can see in the file we just open this is done on the following line:

```
sensorValue = analogRead(analogInPin);
```

Notice there are two things special about this, firstly we did not have to set the pin mode, this is because all analog pins are input only. Secondly we assigned the value to a variable called "sensorValue". Variables are used in code when you wish to save a value to later manipulate or recall it. In this case we declared sensorValue as an "int" at the top of the code, this means we expect sensorValue to be an integer (whole number).

3. Next we will need to connect one of the IR sensors to the Arduino. Connect the power and ground connections from the sensor board to the corresponding pins on the Arduino. Along with one of the sensor cables to an analog input A0 (a pinout of the Arduino can be found at the end of this guide).
4. Now upload your code to the Arduino.
5. Once the uploading has successfully completed, you will need to open the serial monitor. This is a window that will allow you to see data the Arduino is sending back to the PC. It can be accessed by clicking the following icon in the top right of the Arduino IDE:

When this has opened you should see something like this (ignore the output value):



6. Now try passing the line test sheet you were given over the sensor, you should notice the values change
7. Take note of the sensor values you get when you hold the sheet about 1cm from the sensor

White: _____

Black: _____

Conditional branches

Now that we are able to get an input, you may be wondering how we can react to the inputs. This can be done using by conditional branching, the simplest of which is an if statement. An if statement is formatted as follows:

```
if(condition)
{
  //Code that will run if condition is true
}
else if(otherCondition)
{
  //Code that will run if condition was false but otherCondition is true
}
else
{
  //Code that will run if both condition and otherCondition are false
}
```

The condition can be comprised of many different logical operators below is a table of all the ones you might need:

Operator	Name	Description
>	Greater than	True when the left is greater than the right
<	Less that	True when the left is less than the right
==	Equals	True when the numbers are equal
||	OR	True when either condition is true
&&	AND	True when both conditions are true

E.g. `if(sensorValue>300)`

We will now modify the code from the previous section, to light the on-board LED depending on if the sensor is over the track or not.

1. Add the following code below the sensorValue = analogRead(analogInPin); line

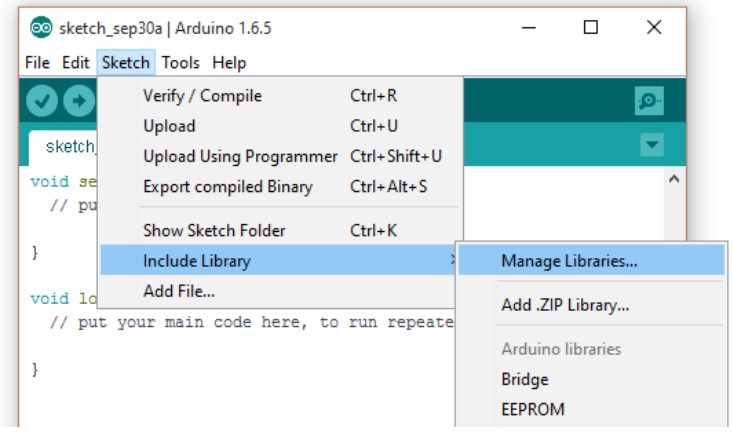
```
if(sensorValue>VALUE)
{

}
else
{

}
```

   Change VALUE to be a number somewhere in between the two values you wrote down earlier.
2. Now adding the code you wrote in part one into the if statement, make the on board LED light up when the track is over the sensor (Don't forget to set the pinMode in the setup function!).

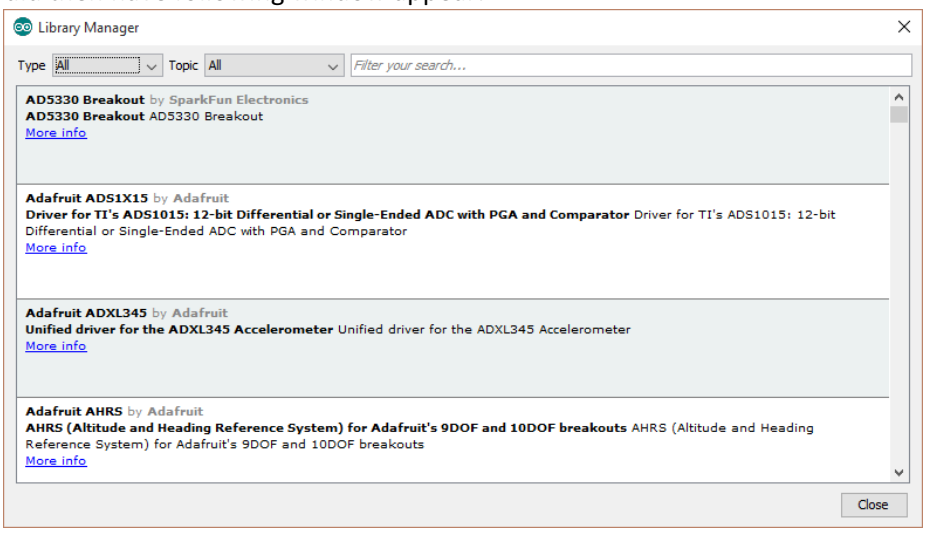
Motor Shield

Before we can start using the motor shields, we have to install an Arduino library for it. A Library is a set of ready made code that will make using the motor shield easier.

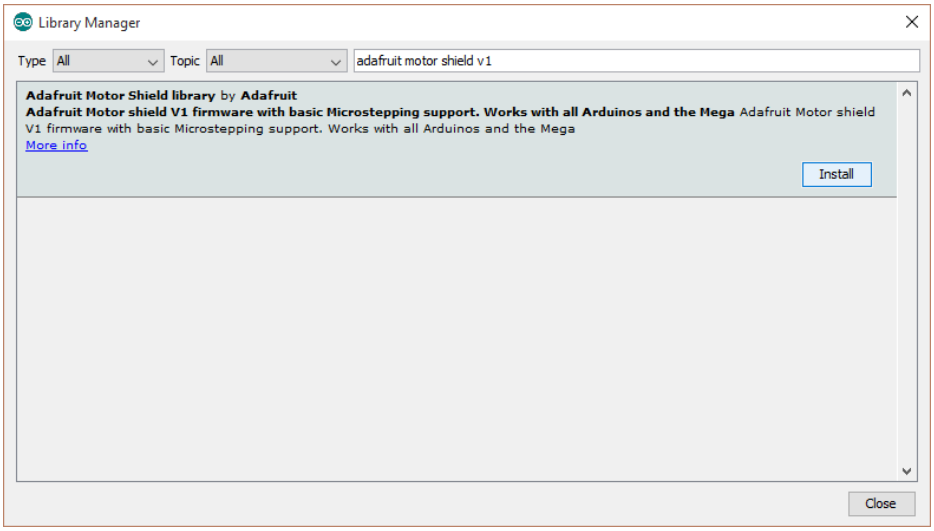
1. First we need to open the Library manager by going to Sketch>Include Library>Manage Libraries…



2. You should then have following window appear:



3. In the search box type in “Adafruit motor shield v1” and you should see:



4. Click the install button to install the library onto your computer

Once you have the motor shield library installed, please fully close and re-open the Arduino IDE to ensure it will work correctly. Once you have a fresh Arduino sketch open, add the following line as the first line in your sketch:

```
#include <AFMotor.h>
```

This line will allow us to use the functions provided in the motor shield library. After this line we will need to add the following code:

```
AF_DCMotor motor(2);
```

This creates an object called motor that will allows us to control motor 2. If you look on your motor shield you will see pairs of terminals along the edges labelled M1, M2, M3 and M4.

In this example the code will control M2. Now we need to set a speed for the motor by adding the following line into the setup function:

```
motor.setSpeed(255);
```

The motor speed can be varied in 255 increments with 0 being off and 255 being full speed. For this project there is no need to vary the speed of the motors unless you want to get fancy. Now in our loop function we simply need to tell the motor shield how we want our motor to move. There are

three options: FORWARD, BACKWARD or RELEASE. To set the direction of the motor simply use the following code:

```
motor.run(FORWARD);
```

See if you can write code to alternate the direction of the motor once every second. After that see if you can control two motors at once!

Note: When using a second motor remember to change the name of the second motor e.g.

```
AF_DCMotor left(1);
```

Line following robot

By using what you have learnt in the previous sections, you should now be able to program a simple line following robot. You will need to read in values from all 5 sensors and then decide how the robot should react. (If you have any queries or you code isn't compiling and you can't figure out why just ask for help from one you your demonstrators). Good luck and most importantly, have fun!

Pinouts