
Arduino Swipe



Header Picture Here - To Be Shot!

Welcome!

This tutorial will teach you how to build a simple swipe and gesture sensor using an Arduino. It will also show you how to use it to control the brightness of an LED, and then to use it to control other more complicated things.

Time To Complete: ~ 1 - 1.5 Hours

Age Range: 12+

Difficulty: Beginner / Intermediate

Getting Started

This is about making sure you can connect to the Arduino properly, and that you have all of the tools and components you will need.

Step 1 - The Sensor Circuit

Here will will build the sensor circuit, and write a little bit of code to make sure it works.

Step 2 - Detecting Movement

Now, lets try and make a simple dimmer switch and actually do something useful with it!

Step 3 - Swiping Left & Right

Detecting a swipe to the left or the right.

Arduino Swipe

Arduino Swipe

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build a simple swipe gesture sensor, and use it to play ping-pong.

Arduino Swipe



Getting Started

This page will tell you how to get the Arduino software installed on your computer, and run a basic program to check everything is working.

1. Downloading The Software

First, head over to [the Arduino website](#) and download the latest version of their software for your computer. They do versions for Windows, Mac OS X and Linux.

Once you have downloaded and installed the Arduino software, launch it and you should see a window like the one shown in the image below.

2. Connecting The Board

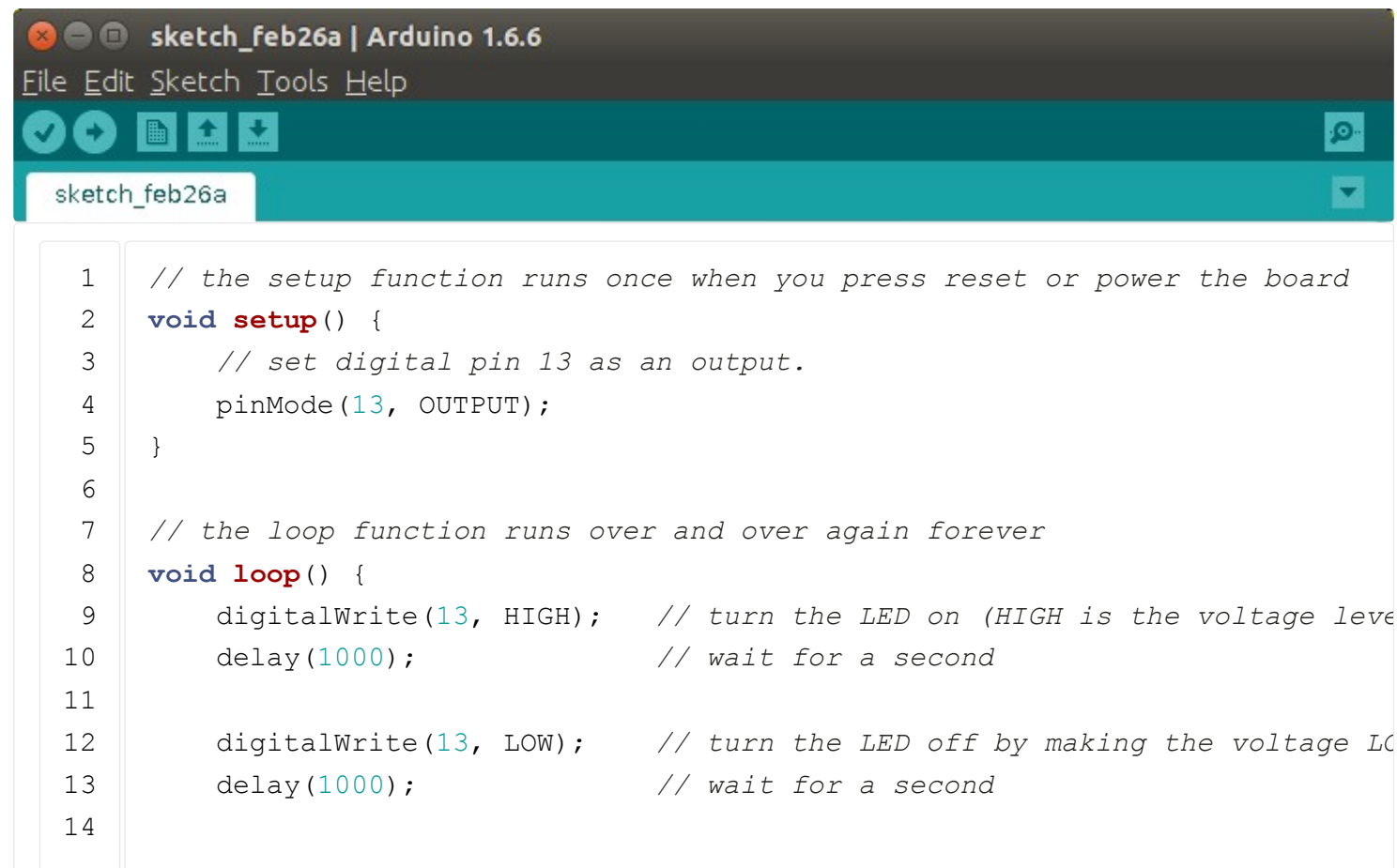
Next, we need to actually connect our Arduino Board to the computer. Use the USB cable provided, and plug it into the mini-USB port on the Arduino, and a USB port on your PC. You should see the little green LED on the board light up.

Now to set up the connection to the board. Using your Arduino software, click on the **Tools** menu icon, and under the *Board:* sub-menu, select the board you are using. For the Digimakers workshop, we are

using the "*Arduino Leonardo*".

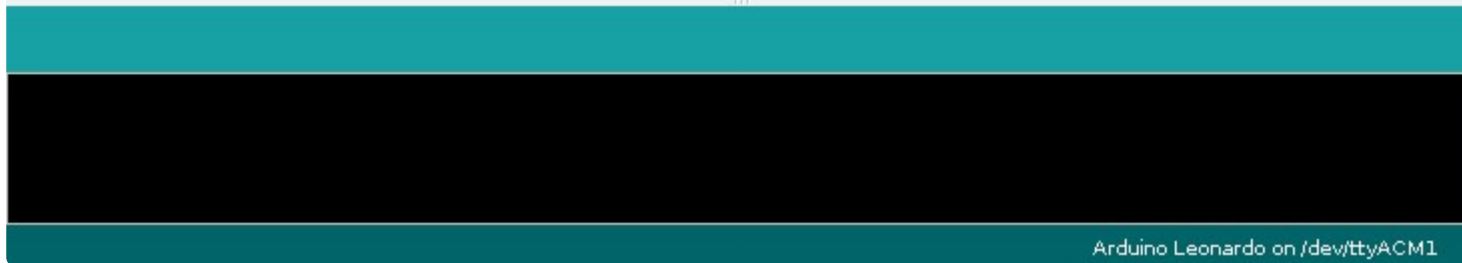
With the correct board selected, click again on the **Tools** menu, and this time click on the "*Port:*" sub-menu. What you will see will depend on your operating system. Windows users will see "COMX" where X is a number, while Linux users will usually see "/dev/ttyACMX". Select the first one in the list. You can try the others if the first one doesn't work in the next step.

Back on your Arduino IDE, goto the **File** menu and select *Examples -> 01. Basics -> Blink*. A new window will appear with some code already inserted. You can now click on the *Upload* button in the **Sketch** menu, or, for wizz-kids, press *Ctrl-U*.

The image shows a screenshot of the Arduino IDE 1.6.6 interface. The title bar reads "sketch_feb26a | Arduino 1.6.6". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening a file, saving, and uploading. A tab labeled "sketch_feb26a" is active. The main text area contains the following C++ code:

```
1 // the setup function runs once when you press reset or power the board
2 void setup() {
3     // set digital pin 13 as an output.
4     pinMode(13, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
10    delay(1000);             // wait for a second
11
12    digitalWrite(13, LOW);   // turn the LED off by making the voltage LOW
13    delay(1000);             // wait for a second
14
```

```
15      // Automatically loop back to the beginning of the loop() function.  
16  }
```



Some text should fly past a the bottom of your Arduino IDE window, the last line of which will be "Upload Complete!". The yellow LED on your Arduino should now be flashing!

3. Next Steps

Now we know we can upload code to our Arduino, can you work out what our "Blink" program does, and how? We'll go over the code properly in the next section, but the more you can figure out now, the better.

In the next section, we will start building some circuits!

[< Home](#) | [Step 1 - The Sensor Circuit](#) >

Arduino Swipe

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build a simple swipe gesture sensor, and use it to play ping-pong.

Arduino Swipe



Step 1 - Setting Up The Sensor

First things first, we need to build the circuit for our gesture sensor. This section will show you the circuit diagram for you to copy, and also explain how it works.

About The Components

We will be using just two different components to build our gesture sensor! The first is the Arduino, obviously. The second is called a *Light Dependent Resistor* or *LDR*. It's the one that looks like a little circle with legs in the picture below.

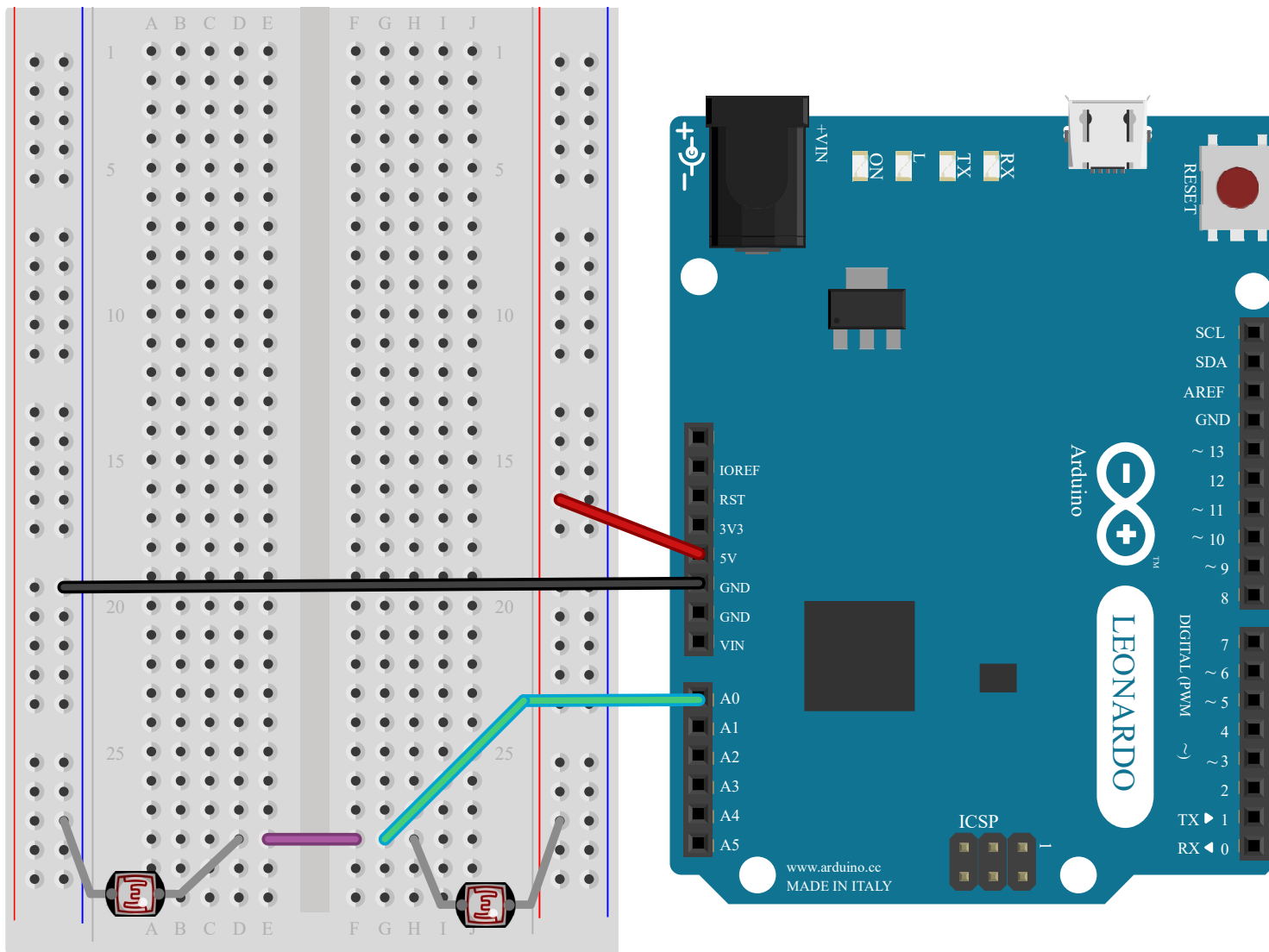
Photo of light dependent resistor

You've probably guessed, that it is a resistor, and that it's *resistance* changes depending on how much light we shine on it's face. We can build a circuit that uses two of these, and detects that one of them has been covered up.

Building the circuit.

The diagram below shows how to connect two LDRs to the Arduino. Believe it or not, this is all the circuitry we need to make our sensor! Copy it carefully using your components, and then move on to the next section. Can you guess how the circuit is going to work? Remember, the **A0** pin we have connected

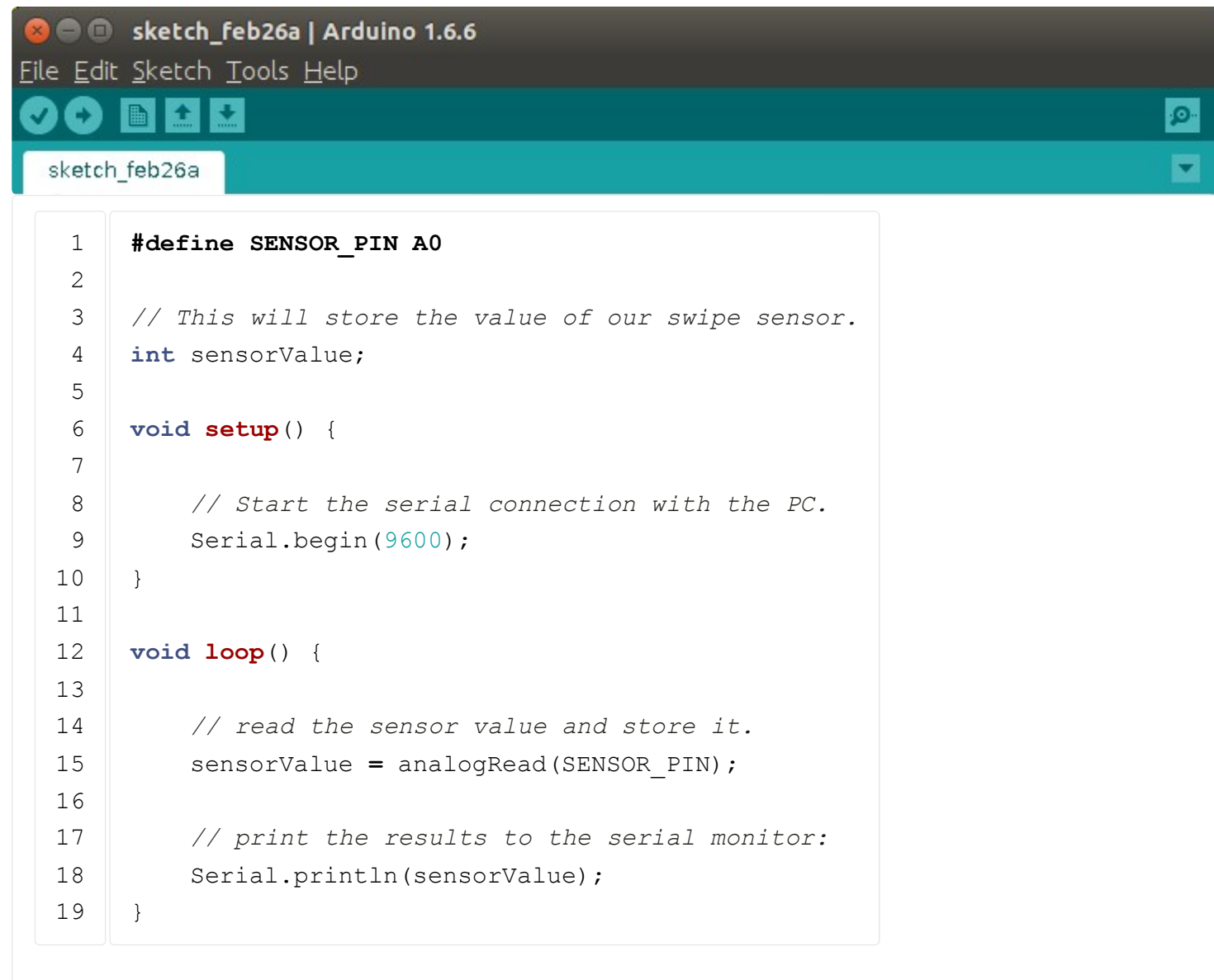
to the Arduino is an *input*, and the LDRs change depending on how much light they can see.



Testing It Out.

Now, lets write some code, and work out how our circuit behaves when we wave at it. See if you can

work out what each line of the code does, and how your code relates to the circuit you just made.



```
1  #define SENSOR_PIN A0
2
3  // This will store the value of our swipe sensor.
4  int sensorValue;
5
6  void setup() {
7
8      // Start the serial connection with the PC.
9      Serial.begin(9600);
10 }
11
12 void loop() {
13
14     // read the sensor value and store it.
15     sensorValue = analogRead(SENSOR_PIN);
16
17     // print the results to the serial monitor:
18     Serial.println(sensorValue);
19 }
```



Upload this code to the Arduino, like you did in the [getting started](#) section. Once that has finished, you can use a tool called the *Serial Plotter* by going to **Tools -> Serial Plotter** in the Arduino IDE Menu Bar. All this does is make a graph of the sensor value we are sending back to the computer with the `Serial.println` command. You should see something like what appears in the GIF below:

Finished circuit and graph GIF.

How it works.

Figured it out yet? Don't worry if not. Here's how:

This circuit is called a *voltage divider*. It is a way of taking a big voltage, and making it smaller. It can also be used to do simple maths, make a signal more manageable, and, in our case, detect swipes!

It works by *dividing* the voltage across the two LDRs. Remember, we start with 5V at the top of one LDR, and finish with 0V at the bottom of the other one. Somewhere, we need to lose or "drop" 5V. If the two LDRs have the *same resistance* then each one will drop half of the voltage, or 2.5V. Now, if the top LDR has a much bigger resistance than the bottom one, then *more* volts are dropped by it than the bottom one. This means that the sensor value will be smaller, because by the time we get to the second LDR, there are fewer volts left!

Try thinking of it the other way round, if the top LDR has a very small resistance, and the bottom one a very big one, what do you think the sensor value will be? Big or Small?

Big of course! Because all the volts are "dropped" across the bottom LDR. Now remember that we can change the resistance of the LDR by changing how much light they can see? This is how our gesture sensor will work. By changing the light on one sensor, we affect the voltage between the LDRs, and can measure it with the Arduino!

Don't worry if all of that was confusing (looking at you too parents!), hopefully it will become clear as you carry on. Next up, let's write a little more code so we can easily tell if we are swiping left or right.

[< Previous](#) | [Next >](#)

Arduino Swipe

Arduino Swipe

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build a simple swipe gesture sensor, and use it to play ping-pong.

Arduino Swipe



Finished Circuit Picture - to be shot!

Step 2 - Detecting Movement

Now we have the circuit set up, lets try and do something useful with it. We can start really simple, and just use it like a dimmer switch. It will be up to you later on to make it do something more exciting!

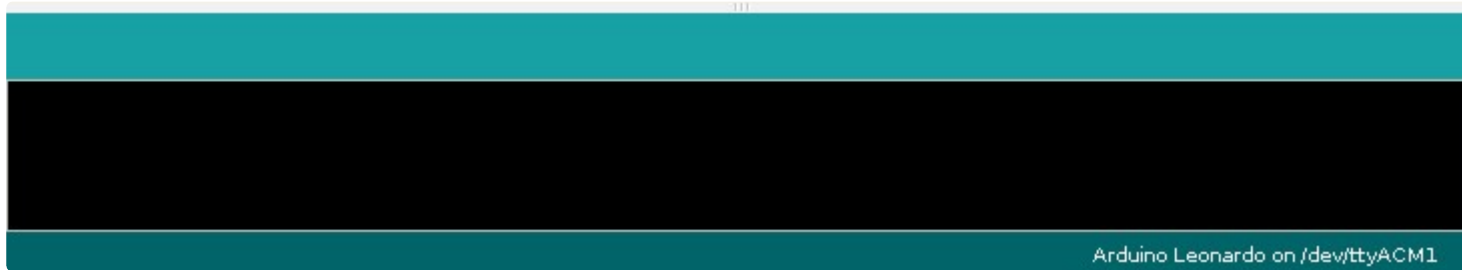
Show Me The Code

Right, lets write the code! The example code is shown below, can you work out which lines do what? See if you can explain to yourself where we work out how fast the sensor value is changing.

```
sketch_feb26a | Arduino 1.6.6
File Edit Sketch Tools Help
[Icons: Check, Run, Upload, Download, Serial Monitor]
sketch_feb26a
1  #define SENSOR_PIN A0
2  #define LED_PIN    11
3
```

```
4  // This will store the most recent value of our swipe sensor.
5  int currentSensorValue;
6
7  // This stores the value we will compare with.
8  int previousSensorValue;
9
10 // This stores how fast the sensor value is changing.
11 int sensorSteepness;
12
13 void setup() {
14     // Start the serial connection with the PC.
15     Serial.begin(115200);
16 }
17
18 void loop() {
19
20     // read the sensor value and store it.
21     currentSensorValue = analogRead(SENSOR_PIN);
22
23     // Update the new steepness value.
24     sensorSteepness = currentSensorValue - previousSensorValue;
25
26     // Move the most recent sensor measurement to the previous one.
27     previousSensorValue = currentSensorValue;
28
29     // Make the LED brighter or dimmer.
30     int brightness = map(currentSensorValue, 0,1023,0,25);
31     analogWrite(LED_PIN, brightness);
32
33     // print the sensor steepness value for the plotter.
```

```
34     Serial.print(currentSensorValue);  
35     Serial.print(" ");  
36     Serial.println(sensorSteepness);  
37     delay(1);  
38 }
```

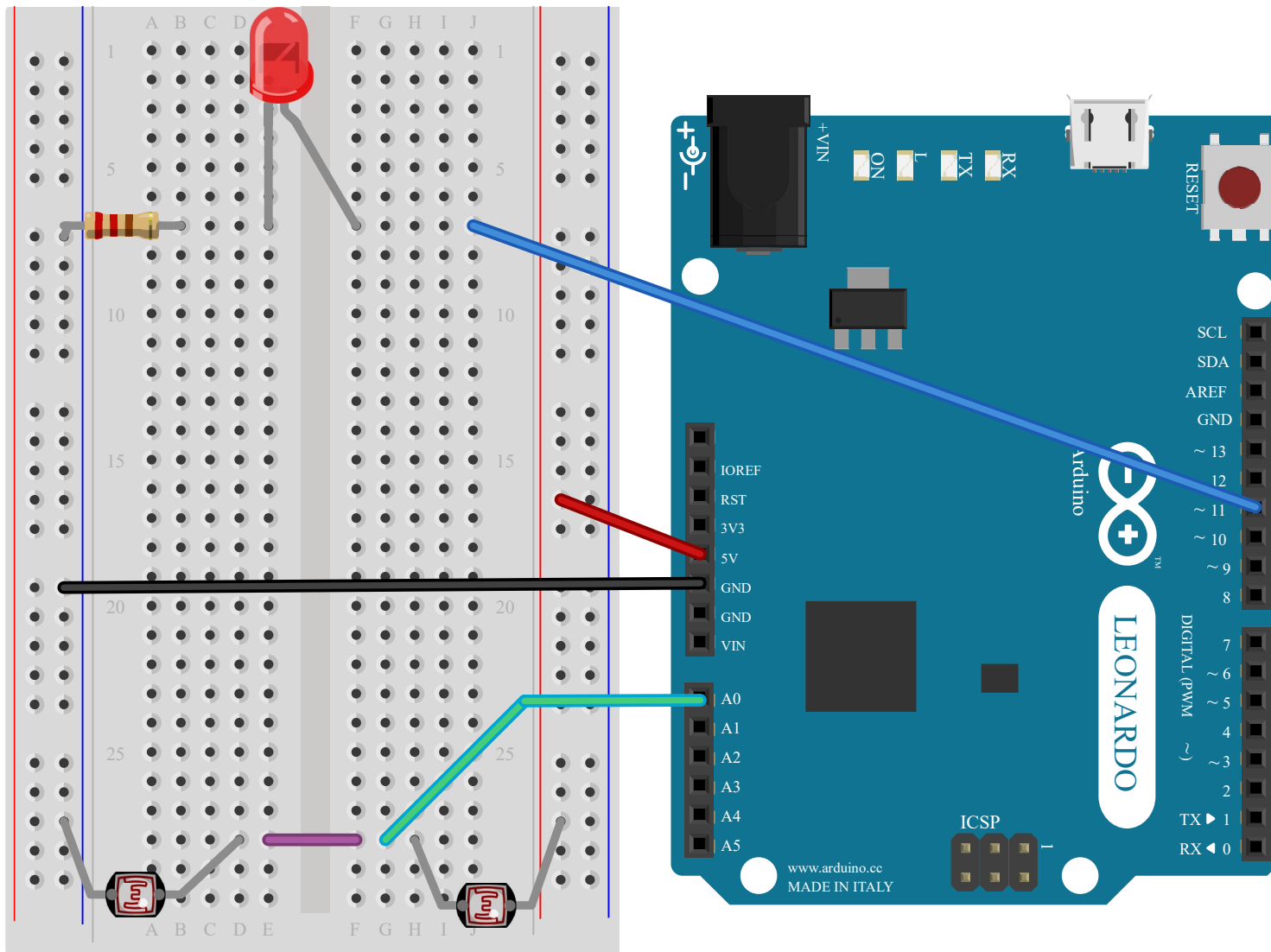


Again, upload the code, and have another look at the output of the serial plotter. What do you see?

Hopefully, you will see that if you move your finger across the two LDRs slowly that you can control the height of the blue plotter line? See if you can gradually move it up and down. If you move your finger slowly, the red line will not move. Now, if you quickly move your finger, you'll see the red line jump up and down very quickly. Can you remember why it jumps only if we move our finger quickly?

A Bright Idea

Now we can see how our code works, and that the circuit works too, let's extend it and make a little dimmer switch. You can copy the circuit below, and hopefully, when you finish it, the code will make the LED brighter or dimmer depending on how you move your finger over the sensor.



Results GIF

Now you've mastered that, let's move on to trying to do actual gestures!

[< Previous](#) | [Home](#) | [Next >](#)

Arduino Swipe

Arduino Swipe

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build a simple swipe gesture sensor, and use it to play ping-pong.

Arduino Swipe



Step 3 - Detecting Swipes

PICTURE HERE

A little bit of thinking...

Take a moment to play with your current circuit. While looking at the Serial Plotter tool, observe how the waveform changes when you swipe left or right over the sensors, or don't put your hand near it.

Notice that if you are careful, and get your hand or finger in just the right place, you can control quite accurately where the line on the plot window goes? We can use this as a simple *slider* and control the position of something on the screen!

We can also work out the *direction* of a swipe across the sensor, by looking at how the value we measure is changing. If it is increasing, we can call this a left swipe, and if it is getting smaller, it's a right swipe.

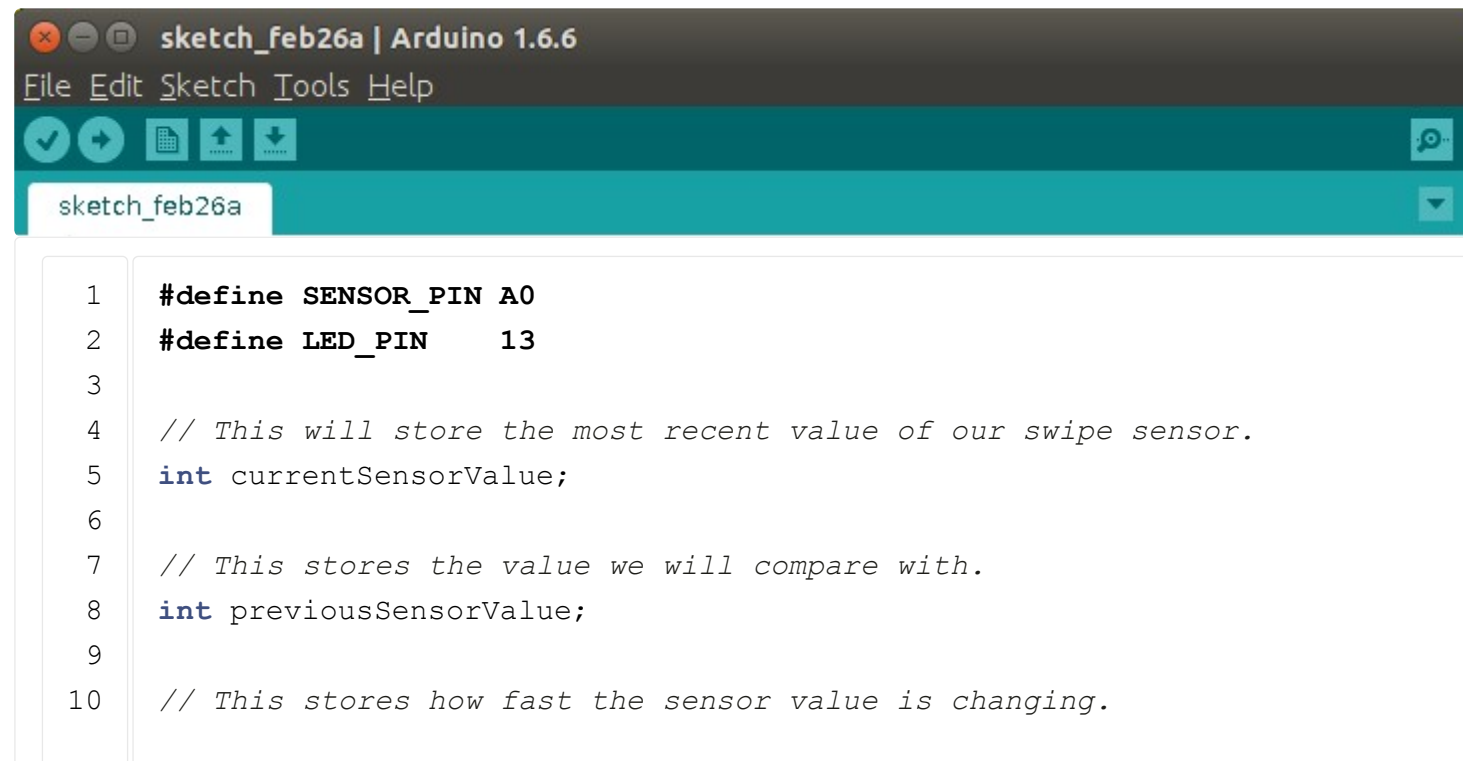
1. If the sensor value is not changing, let's assume there is no swipe. This is the easy bit!
2. Next, we need to work out *if* the value is changing, and then which *direction* it is going in. We can do this by storing two sensor values. The most recent value, and the *previous* value.
 - If we subtract one from the other, and the value is zero, we know it isn't changing.
 - If the result is big, then we are swiping one way, and if the result is negative, we are swiping the other way

Cool eh? This is way of finding how fast a signal is changing is called *taking the derivative*. The picture below shows what happens when we *take the derivative* of a signal. Notice that the graph on the right has a bigger value when the graph on the left is steepest?

Left and Right

So now we know what to look for, lets try and detect a left and right swipe properly. Try to look at the code below and work out what it is doing first. This will help you understand how to change it later on.

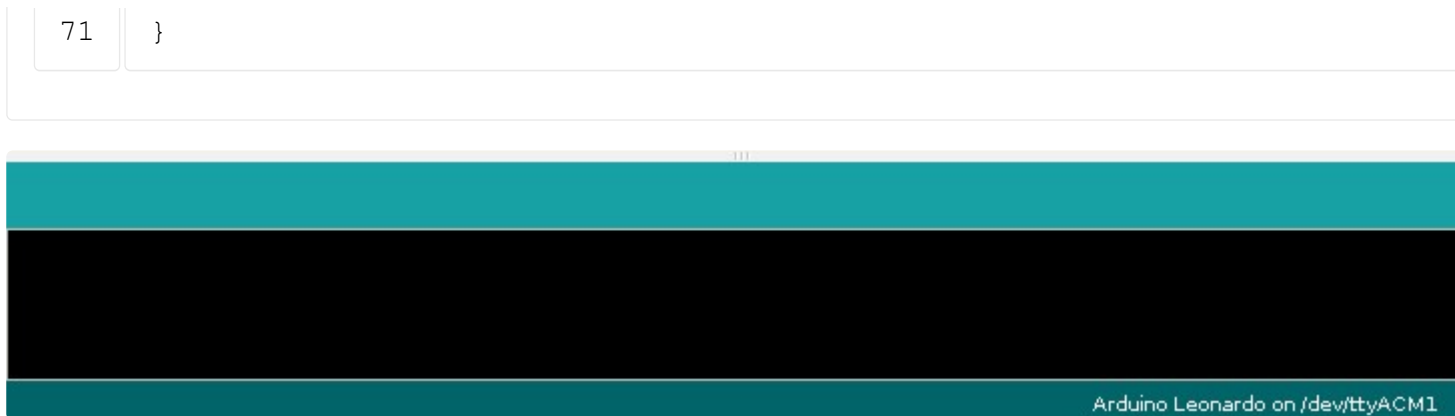
Remember, to find out the direction of a swipe, we look at the value of the *derivative* of the sensor value, which is how fast it is changing. By looking at whether the rate of change of the sensor value is above or below a certain point, we can say a left or right swipe has happened.

A screenshot of the Arduino IDE interface. The title bar reads 'sketch_feb26a | Arduino 1.6.6'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for opening, saving, and uploading. The main editor area shows a sketch named 'sketch_feb26a' with the following code:

```
1  #define SENSOR_PIN A0
2  #define LED_PIN    13
3
4  // This will store the most recent value of our swipe sensor.
5  int currentSensorValue;
6
7  // This stores the value we will compare with.
8  int previousSensorValue;
9
10 // This stores how fast the sensor value is changing.
```

```
11  int sensorSteepness;
12
13  // We use this to make sure we don't accidentally detect lots of swipes.
14  int swipe_damper = 1;
15
16  // Record which direction we are going in here.
17  int swipe_direction;
18
19  void setup() {
20      // Start the serial connection with the PC.
21      Serial.begin(115200);
22  }
23
24  void left_or_right(){
25      // This function will work out whether we are swiping in a particular
26      // direction or not.
27
28      // These will determine how fast we have to move to make a swipe.
29      const int threshold_1 = 10;
30      const int threshold_2 = 20;
31
32      if(sensorSteepness > threshold_2){
33          // We swiped left!
34          swipe_direction = 100;
35          swipe_damper    = 100;
36      }else if(sensorSteepness < -threshold_2){
37          // We swiped right!
38          swipe_direction = -100;
39          swipe_damper    = 100;
40      }else if(sensorSteepness < threshold_1 && sensorSteepness > - threshold
```

```
41         // No swipe
42         swipe_direction = 0;
43     }else{
44         // Not sure if there is a swipe, so assume no.
45         swipe_direction = 0;
46     }
47 }
48
49 void loop() {
50
51     // read the sensor value and store it.
52     currentSensorValue = analogRead(SENSOR_PIN);
53
54     // Update the new steepness value.
55     sensorSteepness = (currentSensorValue - previousSensorValue) / swipe_c
56     swipe_damper      = swipe_damper > 1 ? swipe_damper - 1 : 1;
57
58     // Lets work out the direction of a swipe.
59     left_or_right();
60
61     // Move the most recent sensor measurement to the previous one.
62     previousValues[valueIndexAdd] = currentSensorValue;
63
64     // print the sensor steepness value for the plotter.
65     Serial.print(currentSensorValue);
66     Serial.print(" ");
67     Serial.print(swipe_direction);
68     Serial.print(" ");
69     Serial.println(sensorSteepness);
70     delay(1);
```



When you run this code, you should see a new line in the plotter, which goes either up or down based on whether you swiped left or right!

PICTURE HERE

What Next?

So, now you know how to build a gesture sensor using an Arduino! But, we we haven't been very imaginative with it have we? This I leave to you, what possible uses can you think up for it?

- A motion sensitive burglar alarm?
- Using it to control a game? Perhaps [Ping-pong?](#)
- Can you make your own game, where you control the player position with the sensor?

Let us know what you build!

[< Previous](#) | [Home](#)

Arduino Swipe

Arduino Swipe

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build a simple swipe gesture sensor, and use it to play ping-pong.