Glow Your Own – Session 4

Building and coding your circuit

You can build, test and code your circuits virtually on Tinkercad: www.tinkercad.com

If you would like to join the Glow Your Own class on Tinkercad, please log on to https://www.tinkercad.com/joinclass/DF717MS2Y **(class code D F 7 1 7 M S 2 Y)** and enter your nickname – your nickname has the form **gyo###**, and everyone who signed up received a code with three unique numbers. If you need a new nickname, please email visitral@stfc.ac.uk

For a reminder of how to access Tinkercad and get started, see the 'How To' sheet for Session 1. There are instructions for how to upload your code in the 'Duino' How To sheet.

Remember, if you're building physical circuits with your real Arduino:

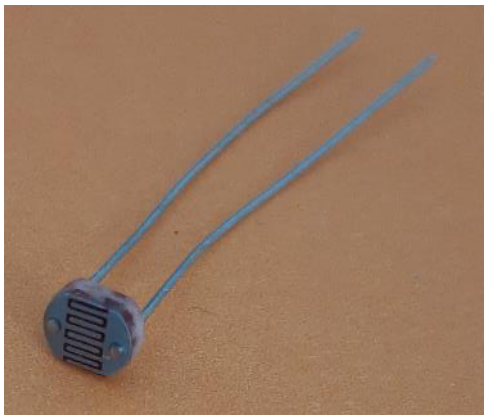
- Always remember to unplug your Arduino when you're changing components.
- Make sure all of the small components are kept out of reach of young children and pets.
- Make sure that all components are tidied up at the end of the session, and none are left on the floor or table.

Circuits to build and code

Turning an LED on when it gets dark

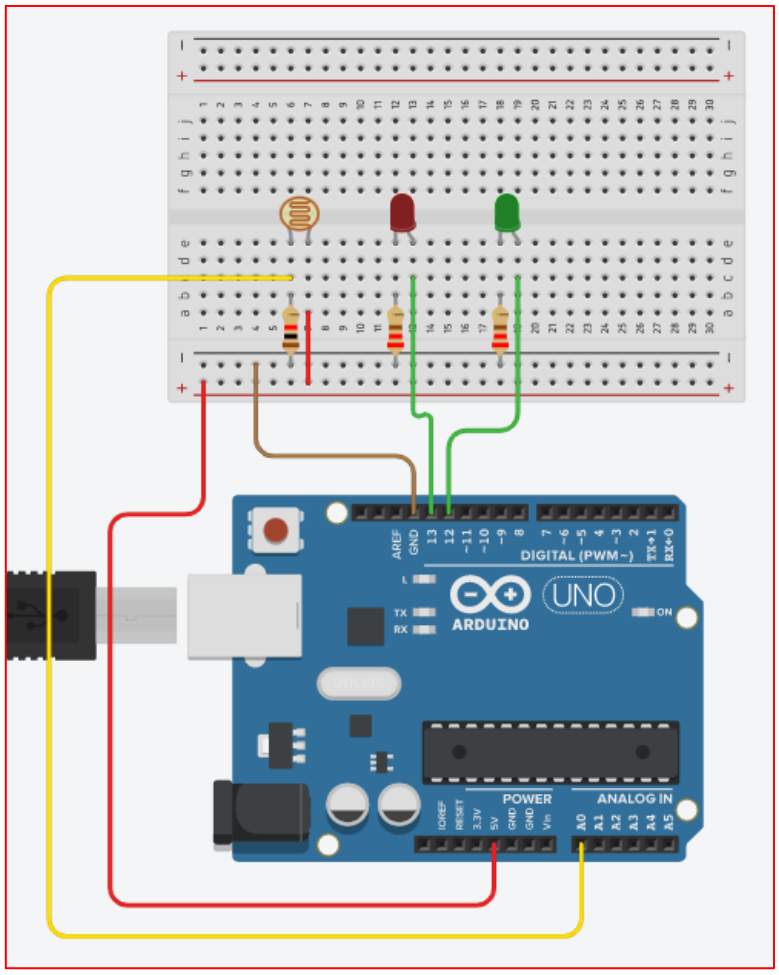
We've used a button to turn our LED off – but we can also use other sensors to control our circuit. You can control your LED by measuring the light around your circuit using a **photoresistor** – so that the LED turns on when it gets dark.

The photoresistor changes its value depending on the light around it. In Tinkercad and in your Arduino kits, the photoresistor looks like this:



To start with, let's build a circuit that makes our LEDs turn on when it gets dark. There are two circuits: one output circuit (with the LED), and one input circuit (with the photoresistor). As you can

see, the photoresistor needs another resistor to work properly – use a 1 kΩ resistor for this. Use 220 Ω resistors for the LEDs, as before.

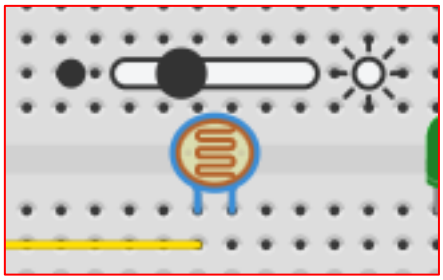


Don't forget the yellow and red wires that are on the left hand side of the Arduino – the red wire provides power to the photoresistor, and the yellow wire lets the Arduino read in how dark it is – the value of the photoresistor.

To code our new circuit, we need to use an "**if then else"** block: as with the button, this block is used when we want to ask the computer a question. We ask what value the photoresistor is saying (which tells us how bright it is). If it's very dark, we instruct the Arduino to turn both LEDs on, if it's quite dark we instruct the Arduino to turn just one LED on, and if it's bright we instruct the Arduino to turn the LEDs off, using the code below.

The block **"print to serial monitor"** means we can tell what value the photoresistor is telling the Arduino, as it will print that value to the screen – there are some more details about this at the bottom of this worksheet. You'll also notice we've put in some **comments**. Comments are really useful in your code – the Arduino ignores them, but they tell us (and other people!) what you are trying to do in your code.

When you're simulating your circuit in Tinkercad, click on the photoresistor and using the slider you can change the light level the photoresistor sees.



Challenge with photoresistors

The values that we've used in the code above, to decide whether it's dark or light, can be found by looking at the serial monitor, and experimenting to see what values the photoresistor returns when it's very dark or very bright. These are then 'hard coded' into the Arduino (we type the numbers in: 550 and 350 above). We can code the Arduino to work out those numbers for itself by calibrating the photoresistor: recording the readings when it's bright and dark.

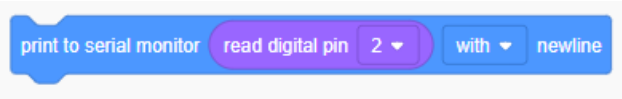
Can you work out how to calibrate your photoresistor? You could use a button to measure the maximum and minimum light reading over a period of 5 seconds, and use these values in the code above. You'll need to use **variables** to do this, which is when you instruct the Arduino to remember a certain value! Good luck!

Using the serial monitor

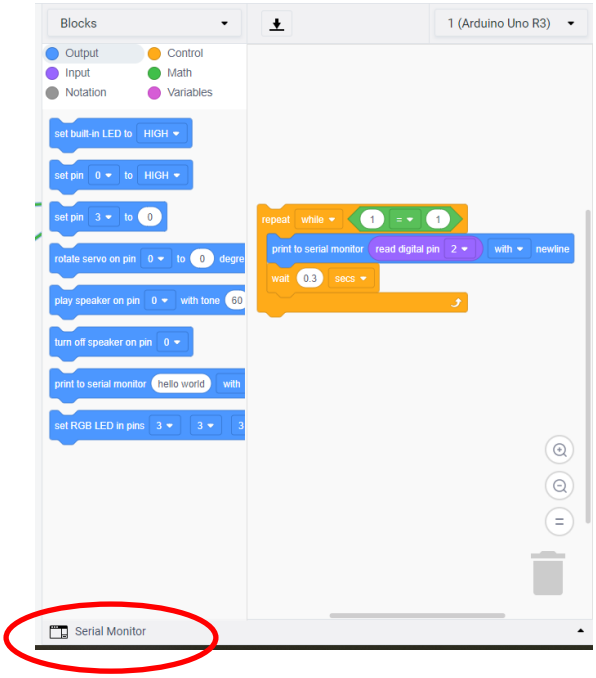
You might have noticed by now, but coding often doesn't work the first time you write something! In this case, it's helpful to be able to **debug** (fix) our code by getting the Arduino to let us know what values it's using. We can do that using the **serial monitor**, which is an area you can make appear on your computer screen, where the Arduino can talk to you.

The serial monitor in Tinkercad

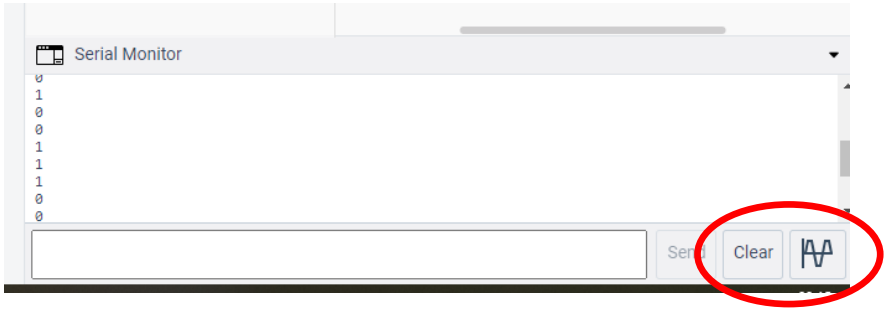
In Tinkercad, you can ask the Arduino to print a value to the serial monitor using the "**print to serial monitor"** block, which is in "Outputs"



Here, we're asking the Arduino to print the value it is reading from whatever is connected to digital pin 2. You can see the serial monitor in Tinkercad at the bottom of the Code section:



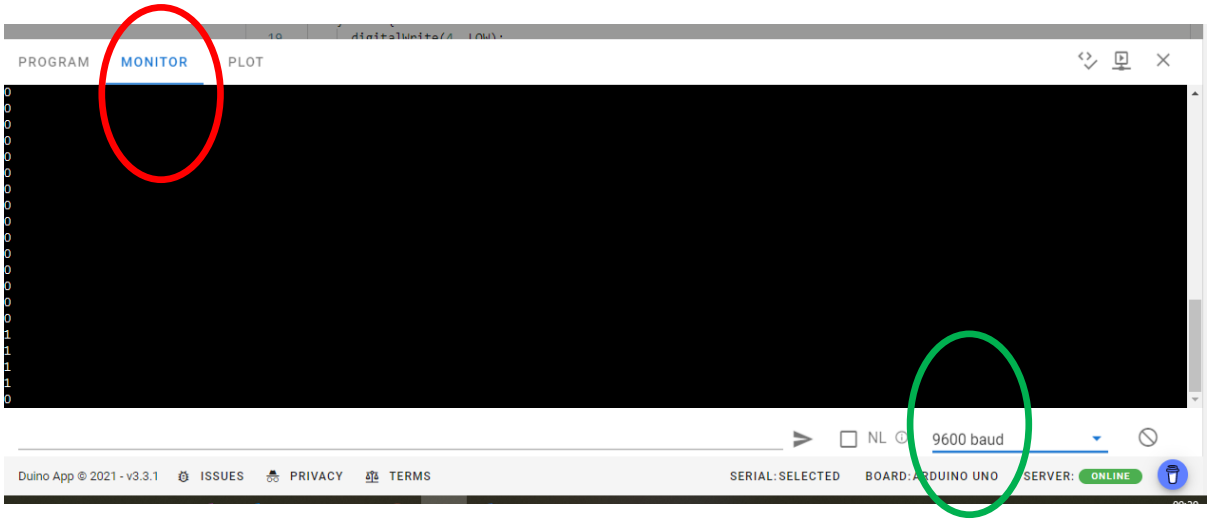
When you click on the Serial Monitor button, you will see values appear when you run a simulation.



You can use the “Clear” button to clear all previous readings from the serial monitor, or the Graph button to see a graph of your readings.

The serial monitor in the Duino App

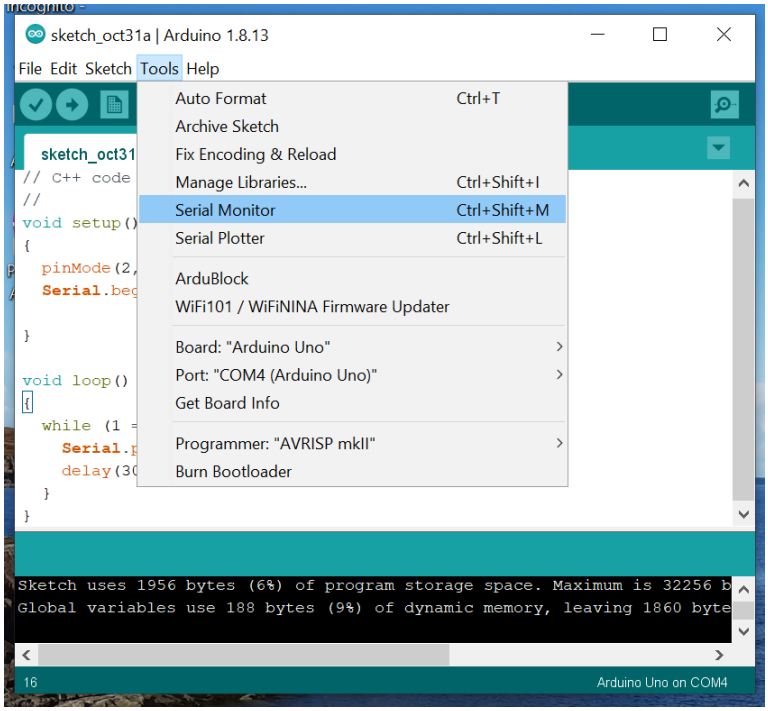
In the Duino App, click on ‘MONITOR’ to see the Serial Monitor output:



Make sure it is set to 9600 baud rate (circled in green) – this sets the speed for serial communication with the Arduino.

The serial monitor in the Arduino IDE

You can also use the serial monitor with your physical Arduino, using the Arduino IDE. To see the Serial Monitor, click on the Tools menu, and then Serial Monitor (or press ctrl + shift + M).



This opens up a new window, with the serial monitor – you can clear the serial monitor using the "Clear output" button.