Glow Your Own

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/Q9Z4QH93S57B and enter your nickname. If you need a new nickname, please email visitral@stfc.ac.uk

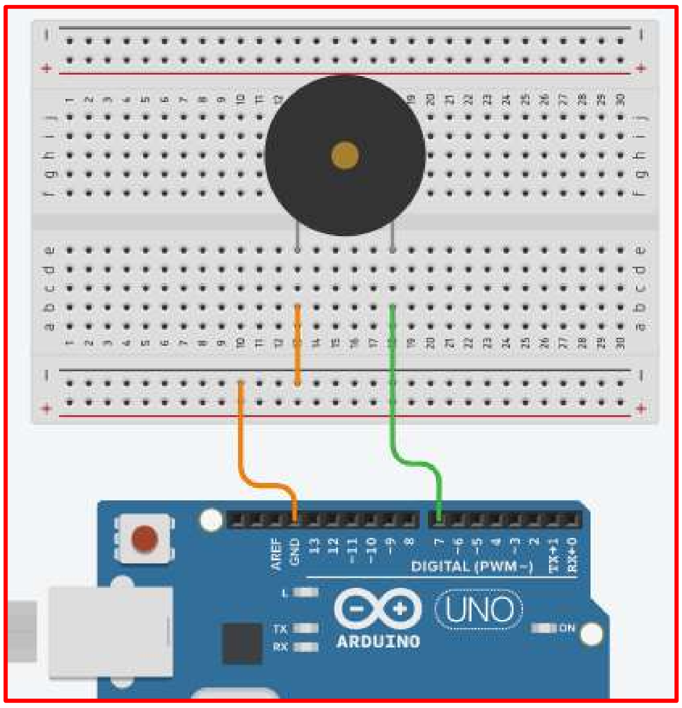
You can remind yourself how to get started, and upload code to your Arduino, using the "GlowYourOwn-HowToSheet".

Building a theremin

A theremin is an electronic musical instrument that's controlled without touching it. It was invented in 1928 by Leon Theremin. To build a simple theremin, you need a **piezo buzzer**, a resistor, some wires and a **photoresistor**. The buzzer will make a noise – and the frequency of the noise depends on how much light the photoresistor detects. You, as the musician, can control the sound by changing how much light the photoresistor sees.

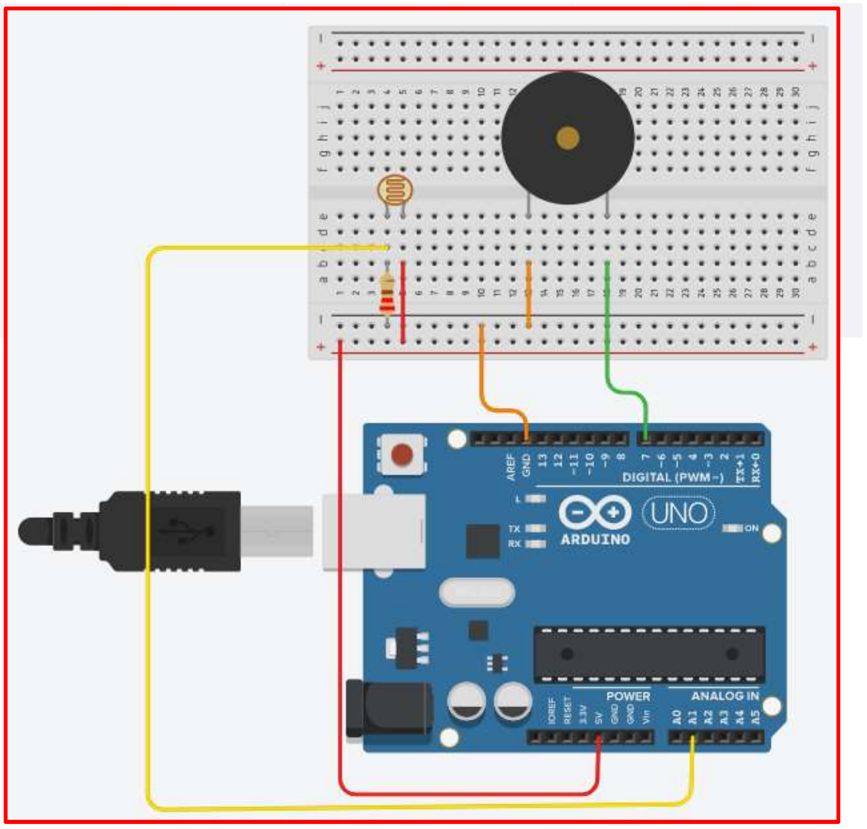
WARNING! This can be noisy, so don't practice when people are asleep around you!

Firstly, we'll build the circuit that connects the piezo buzzer to the Arduino. This is a simple circuit, going from one of the Arduino pins to the piezo buzzer and then back to ground.



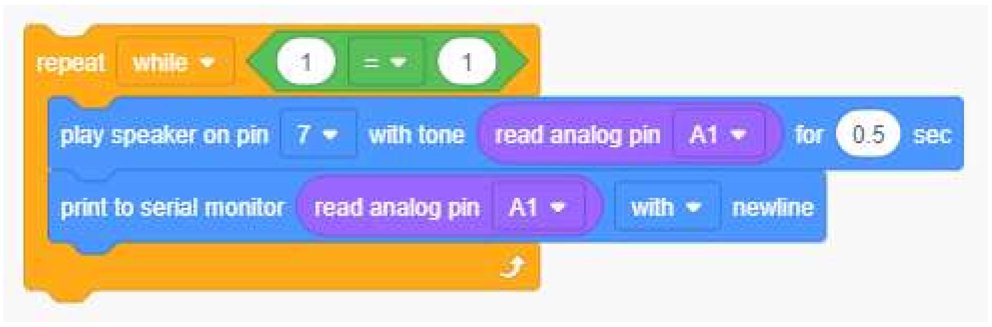
When you're making the physical circuit, make sure the wires are connected to the same breadboard columns as the buzzer legs – it can be hard to see under the buzzer!

Next, we need to add another circuit for the photoresistor – which can send information to the Arduino about how bright it is. We've used this circuit before, when we turned an LED on when it got dark. Add the photoresistor circuit below to your breadboard.



(The best value to use for your resistor is 1 kΩ.)

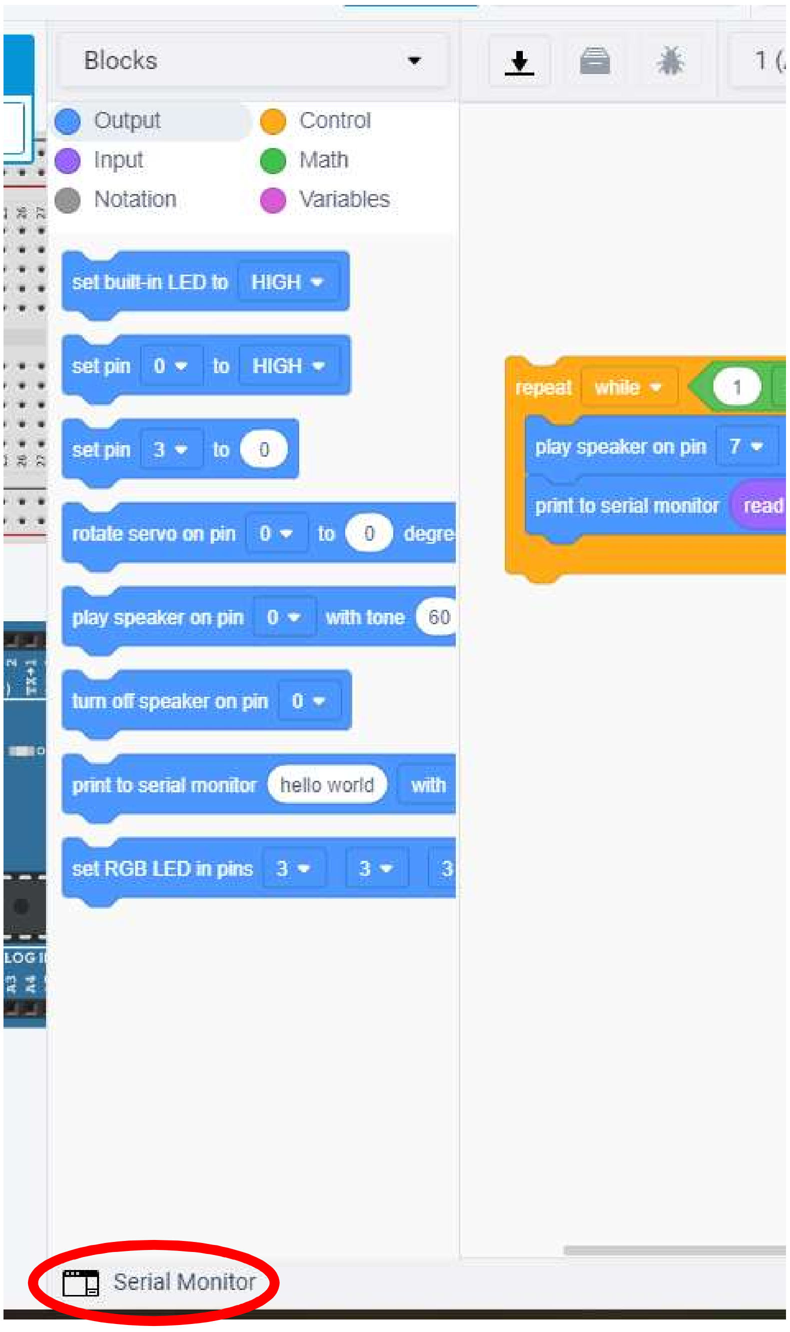
To code the piezo buzzer to make a noise that depends on the light seen by the photoresistor, use the following code:



```
repeat while 1 = 1
    play speaker on pin 7 with tone read analog pin A1 for 0.5 sec
    print to serial monitor read analog pin A1 with newline
```

This tells the Arduino to play the speaker connected to pin 7 with a sound that's controlled by the value the photoresistor is sending to pin A1. You can then test your code by clicking "Start Simulation" on Tinkercad.

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. If you’re using Tinkercad, you can see the Serial Monitor at the bottom of the screen:


Blocks
Output
Control
Input
Math
Notation
Variables
set built-in LED to HIGH
set pin 0 to HIGH
set pin 3 to 0
rotate servo on pin 0 to 0 degre
play speaker on pin 0 with tone 60
turn off speaker on pin 0
print to serial monitor hello world with
set RGB LED in pins 3 3 3
repeat while 1
play speaker on pin 7
print to serial monitor read
Serial Monitor

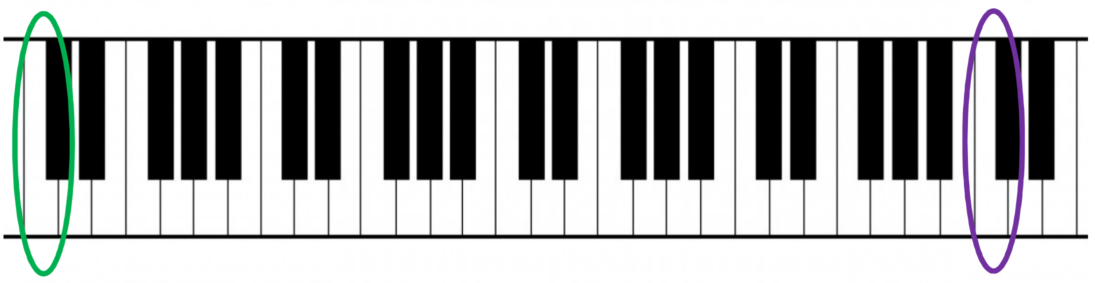
If you’re using a physical Arduino, you can see the Serial Monitor by clicking on the “**Tools**” menu and choosing “**Serial Monitor**”.

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

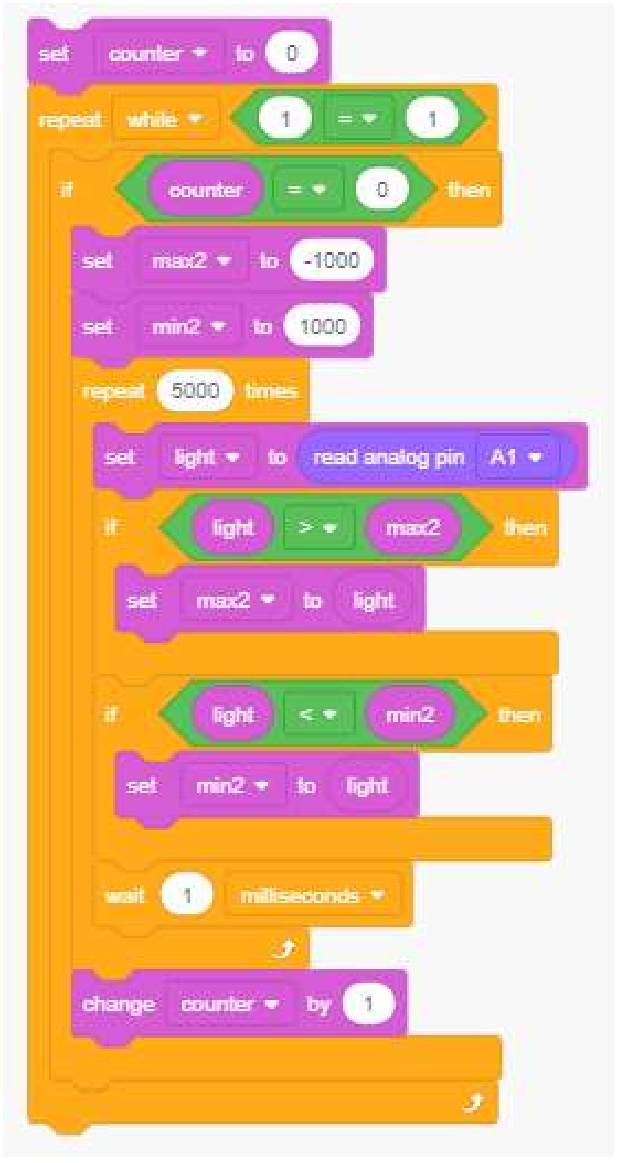
Does your buzzer make a noise? Try moving your hand up and down over the photoresistor – does the sound change?

Changing the frequency

You can change the frequency of the piezo buzzer's sound so that it's more like playing a 'piano'. Let's say the lowest note you want to play is two octaves below middle C (circled in green) – and the highest note you want to play is two octaves above middle C (circled in purple).

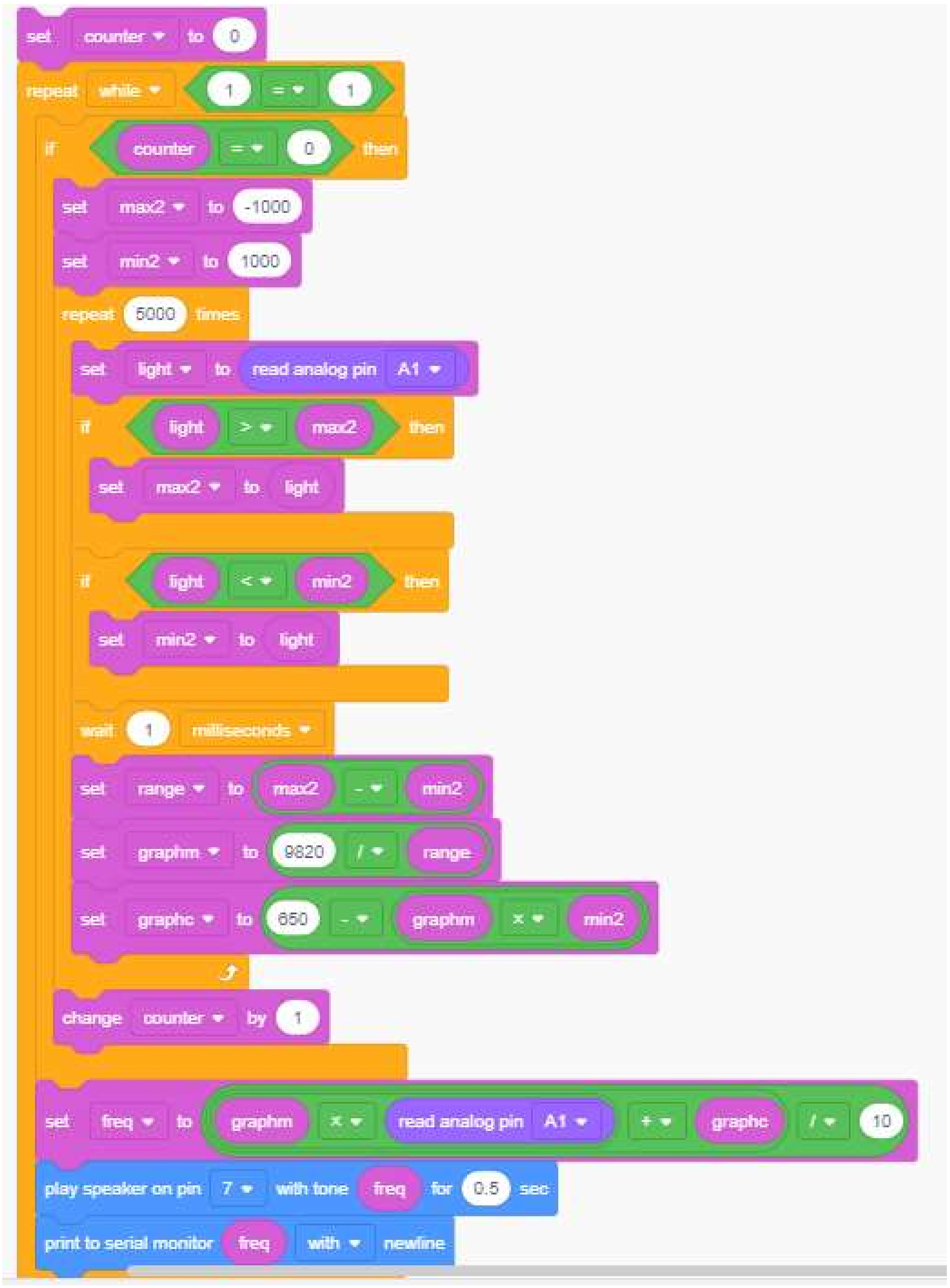


The frequency of the lowest note is 65 Hz, and the frequency of the highest note is 1047 Hz. You can calibrate the theremin by recording the lightest and the darkest light levels, using the code below



```
set counter to 0
repeat while 1 = 1
    if counter = 0 then
        set max2 to -1000
        set min2 to 1000
        repeat 5000 times
            set light to read analog pin A1
            if light > max2 then
                set max2 to light
            if light < min2 then
                set min2 to light
            wait 1 milliseconds
        change counter by 1
```

Once you know the maximum and minimum light levels, you can map that onto the maximum and minimum frequencies you want to play, as below:



```
set counter to 0
repeat while 1 = 1
    if counter = 0 then
        set max2 to -1000
        set min2 to 1000
        repeat 5000 times
            set light to read analog pin A1
            if light > max2 then
                set max2 to light
            if light < min2 then
                set min2 to light
            wait 1 milliseconds
            set range to max2 - min2
            set graphm to 9820 / range
            set graphc to 650 - graphm x min2
        change counter by 1
    set freq to graphm x read analog pin A1 + graphc / 10
    play speaker on pin 7 with tone freq for 0.5 sec
    print to serial monitor freq with newline
```

One thing to note is that when using the blocks to code, all the variables are integers – to make the mapping more accurate we have multiplied and then divided by 10. If you choose to use text to code, rather than blocks, you can change these integer variables to “DOUBLE”.