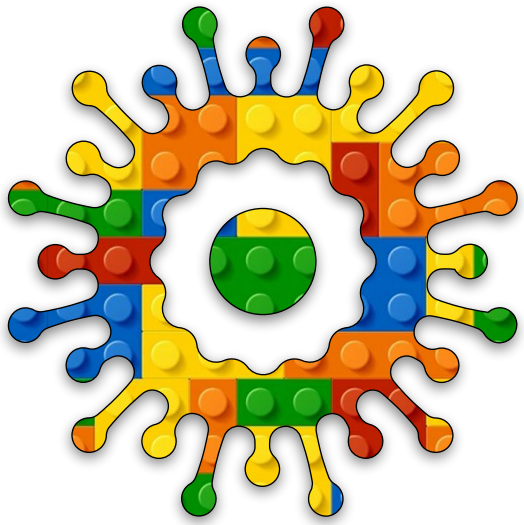
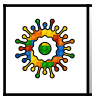
Internet of Things
Module A
Unit #3
blinking





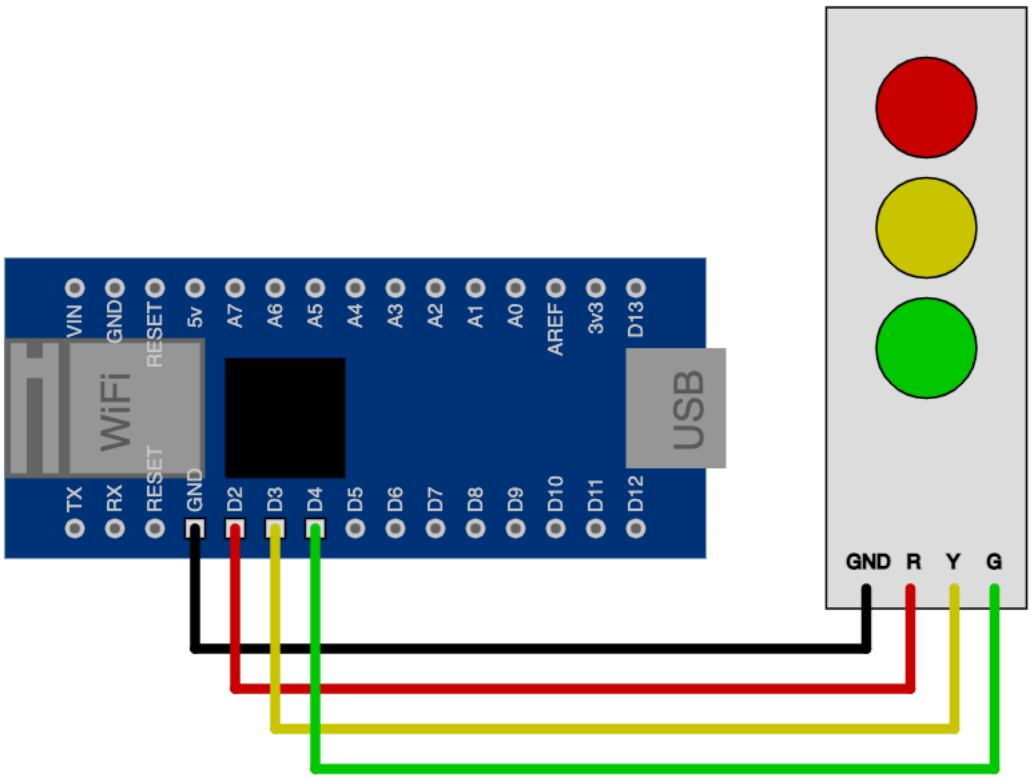
Contents

Module A Unit #3 blinking

Circuit diagram for traffic light LEDs

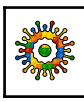
The circuit diagram below shows which pins on the Arduino Nano 33 IoT are connected to which pins on the traffic lights. In the real world, you will put the traffic light on the breadboard and connect them directly to the pins on the Arduino.

Figure 1: circuit diagram



The use of circuit diagrams is to simplify the connections. This table just illustrates the connections.

Arduino Pins	Traffic Lights
GND	GND
D2	R (red)
D3	Y (yellow)
D4	G (green)



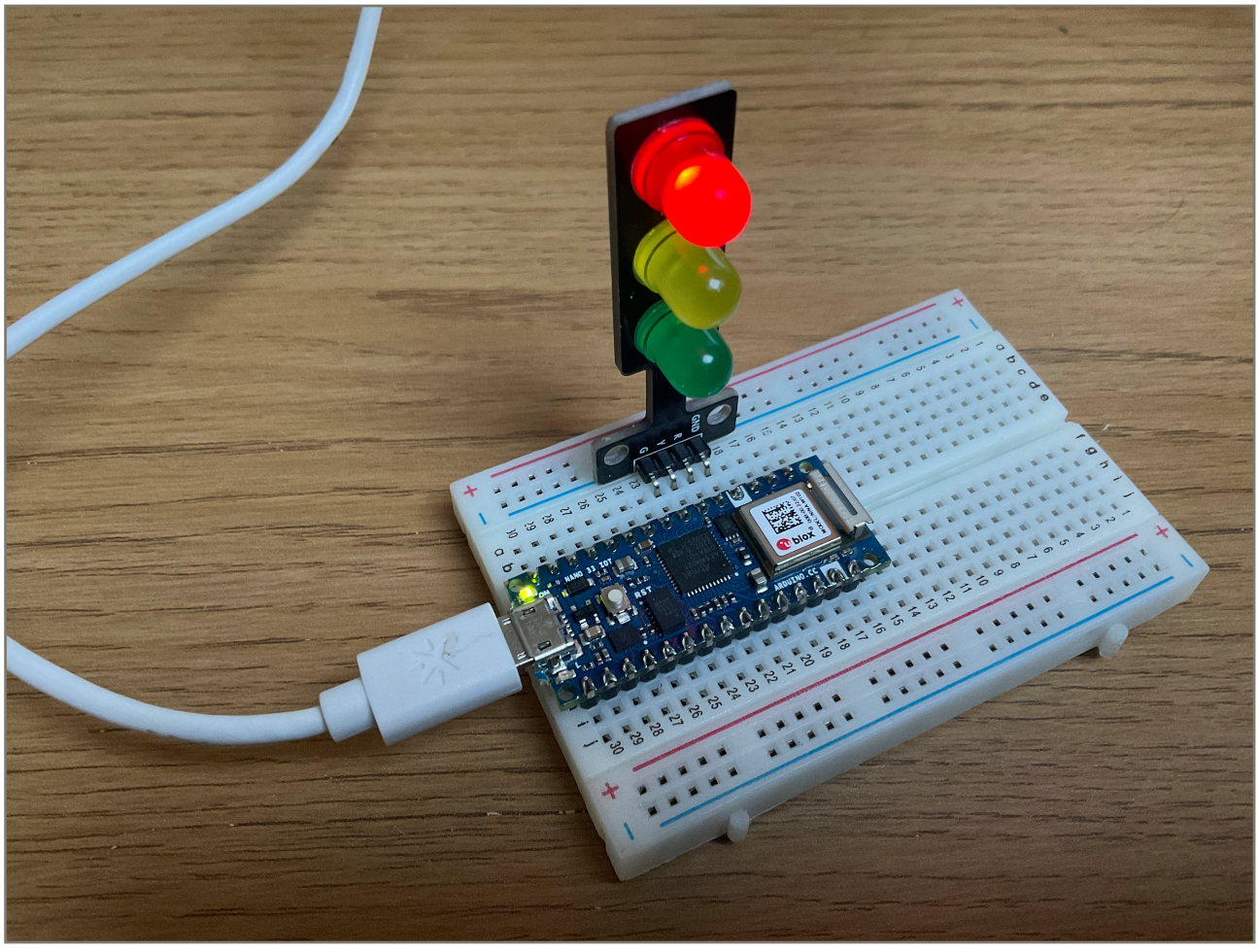
Connecting your Arduino Nano 33 IoT and LEDs

The photo above shows one end of the USB cable connected to the Arduino Nano 33 IoT board. The other end will connect to your computer. The LED Traffic Lights module sits on the breadboard and is connected directly.

Make sure that the GND on the LED module lines up with the ground on the board. The ground has a white square box round the pin to differentiate it. There is also one on the other side, but don't use that one.

If you get it right, then R lines up with D2, the Y lines up with D3, and the G lines up with D4. See fig.2 below.

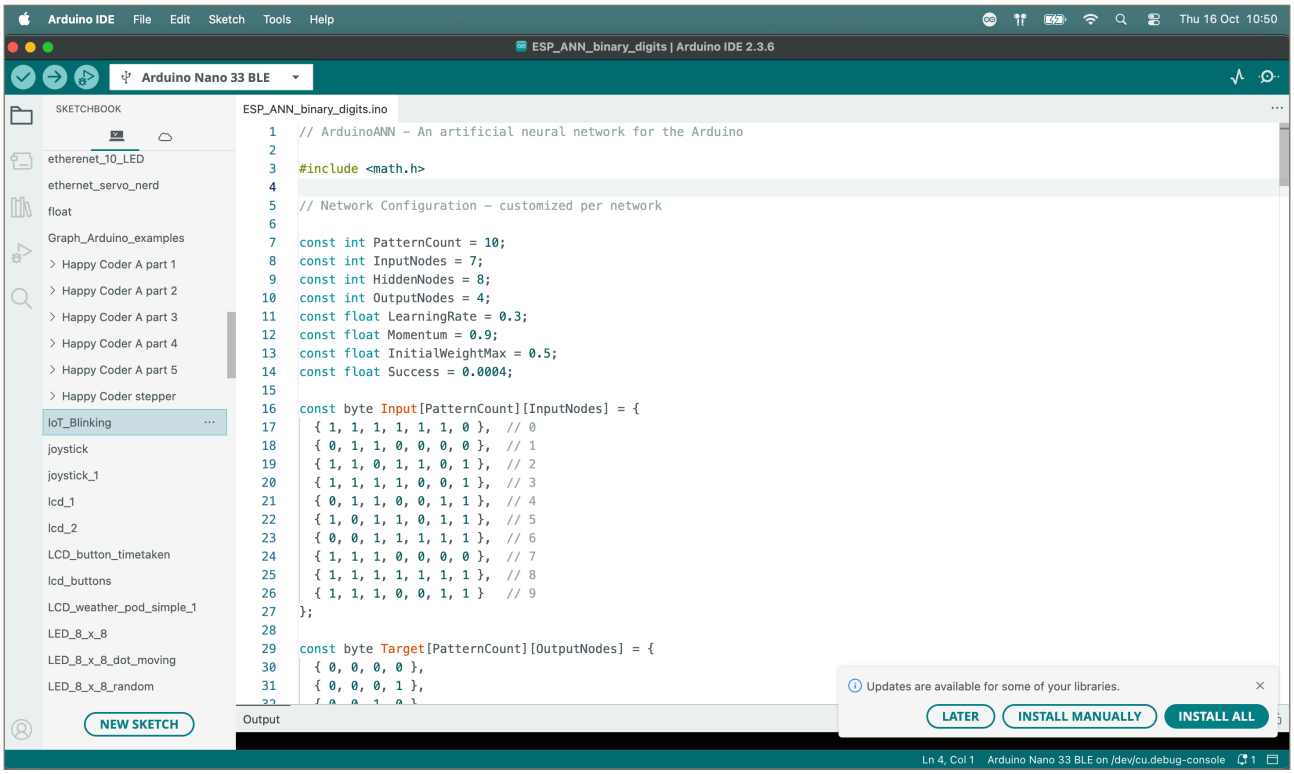
Figure 2: Arduino Nano 33 IoT and Traffic Lights LED



Opening the IDE

When you open the Arduino IDE, you will want to start a new sketch. If, like me, you have other sketches, you may get a page like this. I'm not sure what you will get because I have used the IDE for a variety of purposes over a number of years.

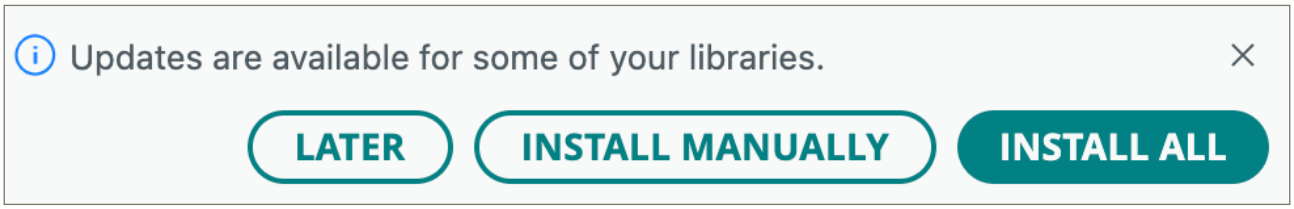
Figure 3: opening of IDE



Installing libraries

If you get a message at the bottom of your screen such as this, I would suggest clicking on INSTALL ALL. It will take a few minutes, but it is worth it in the long run, even though we aren't using any libraries in this unit. Later on, you might.

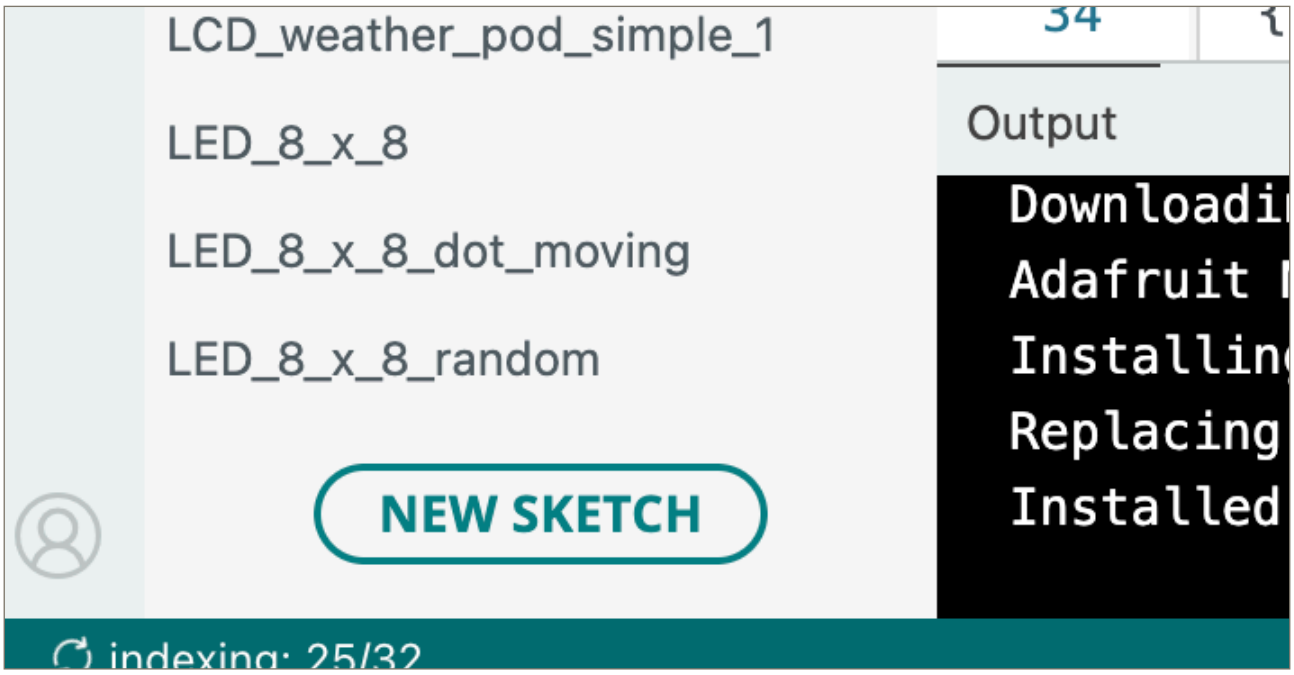
Figure 4: installing libraries



New sketch

Click on NEW SKETCH, especially if you have lots of sketches. It will open in a new tab.

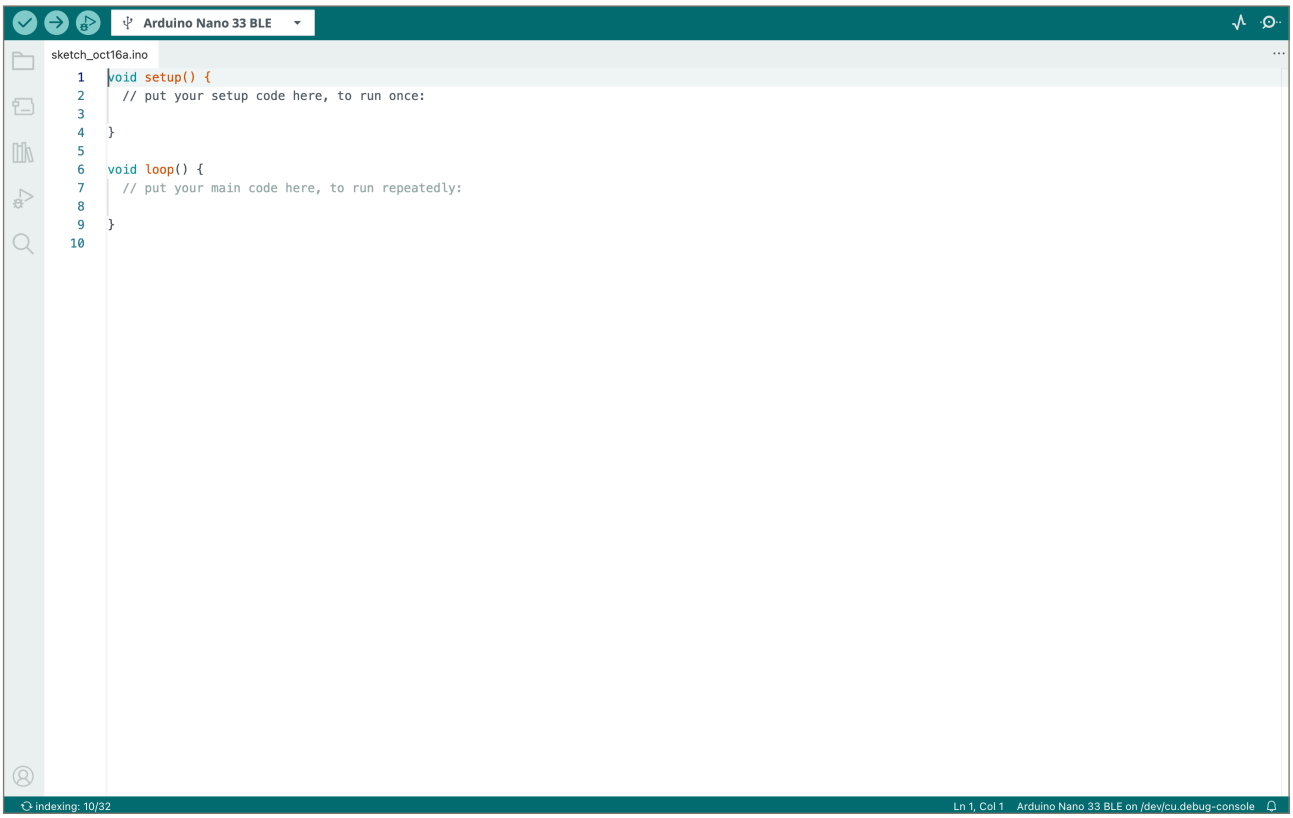
Figure 5: new sketch



Checking you have the right board and port

When you start your new sketch, you get something like this; yours will most likely be slightly different. It gives you the name of the board, in this case, the Arduino Nano 33 BLE (which is not what we want), and if you look at the bottom left, it says which port you are on; in this case, it is /dev/cu.debug-console, which is not what I want either. So we can alter them. This takes a bit of practice and some trial and error. It is important, especially if you are using different boards, like the Uno or something else.

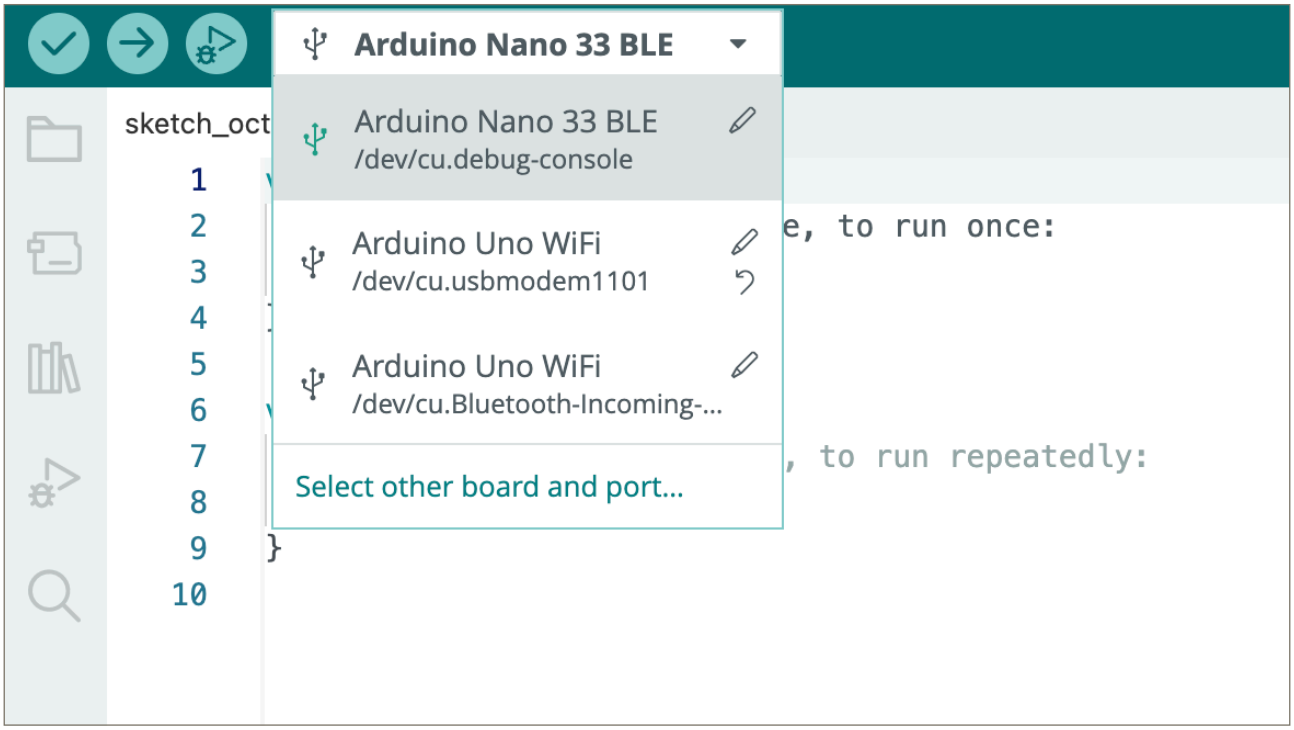
Figure 6: wrong board (and port)



Installing libraries

Click on the arrow next to the name of the board and you will get a drop-down menu. Go to select other board and port…

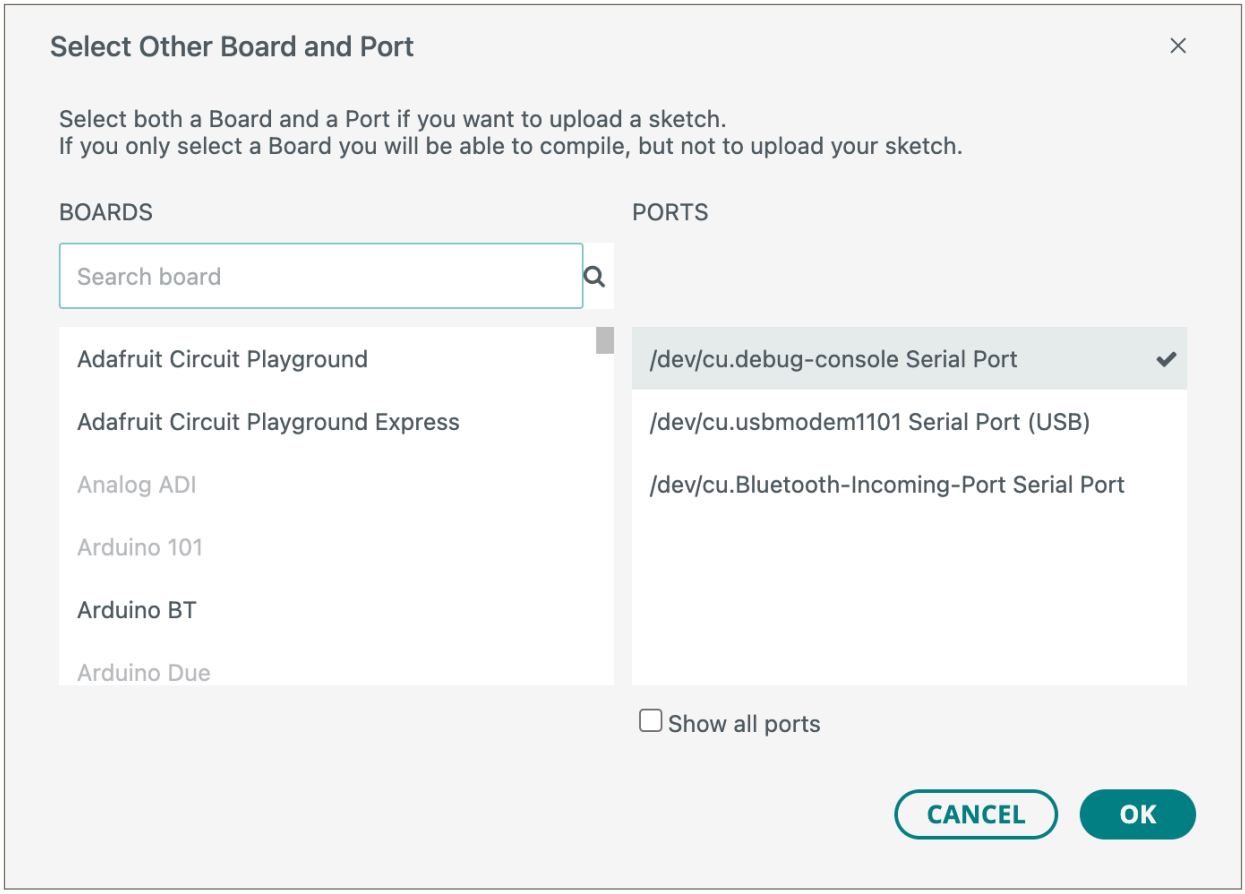
Figure 7: board and port



Selecting the board and port

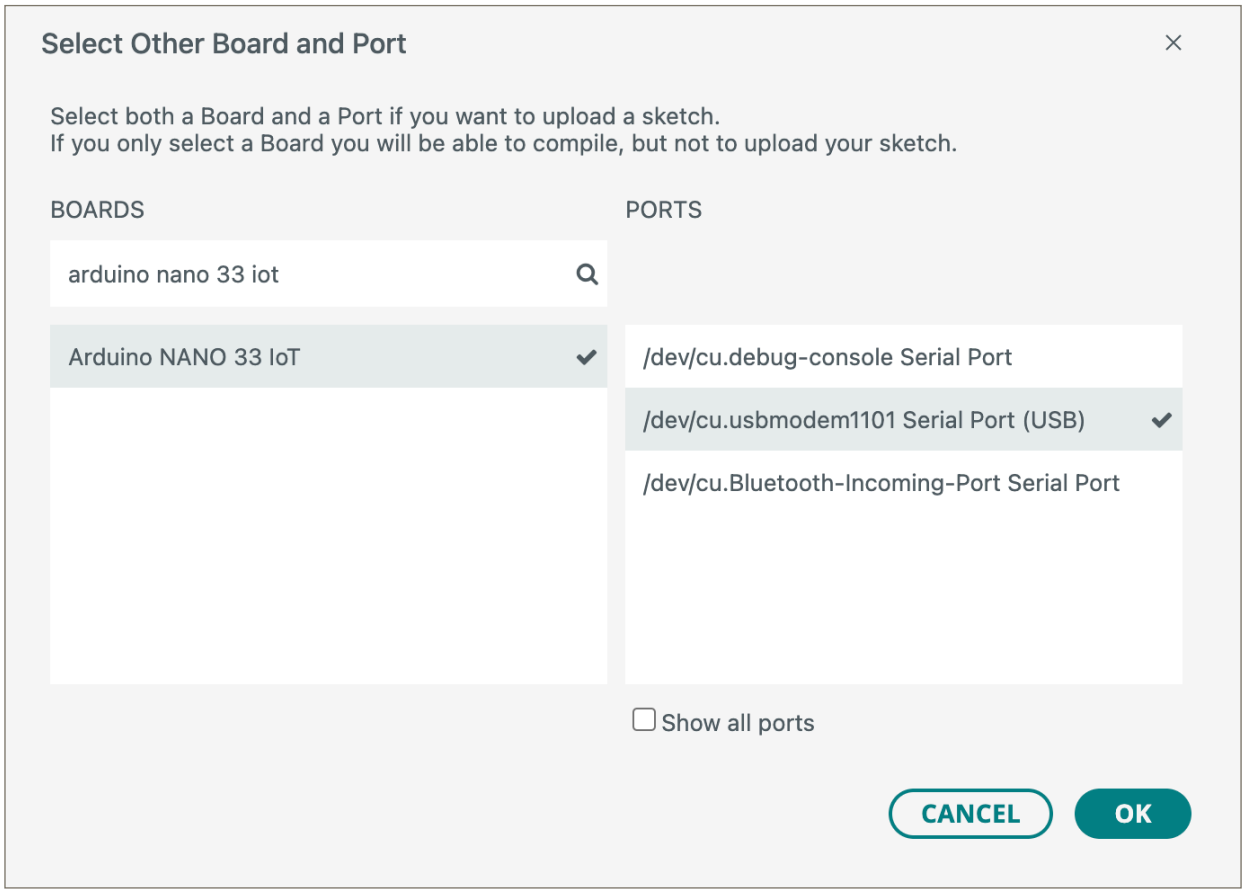
1 On the BOARDS side, there is a box called Search Board. Start typing in Arduino Nano 33 IoT, and when you see it appear, click on it to select that board. On the PORTS side, select the Serial Port (USB) or something similar (your's might be COM2); this will depend on what type of machine you are using. It may well look completely differently (hence a bit of trial and error).

Figure 8: selecting board and port menu



2️⃣ Select the correct board and port, and click OK.

Figure 9: selecting board & port

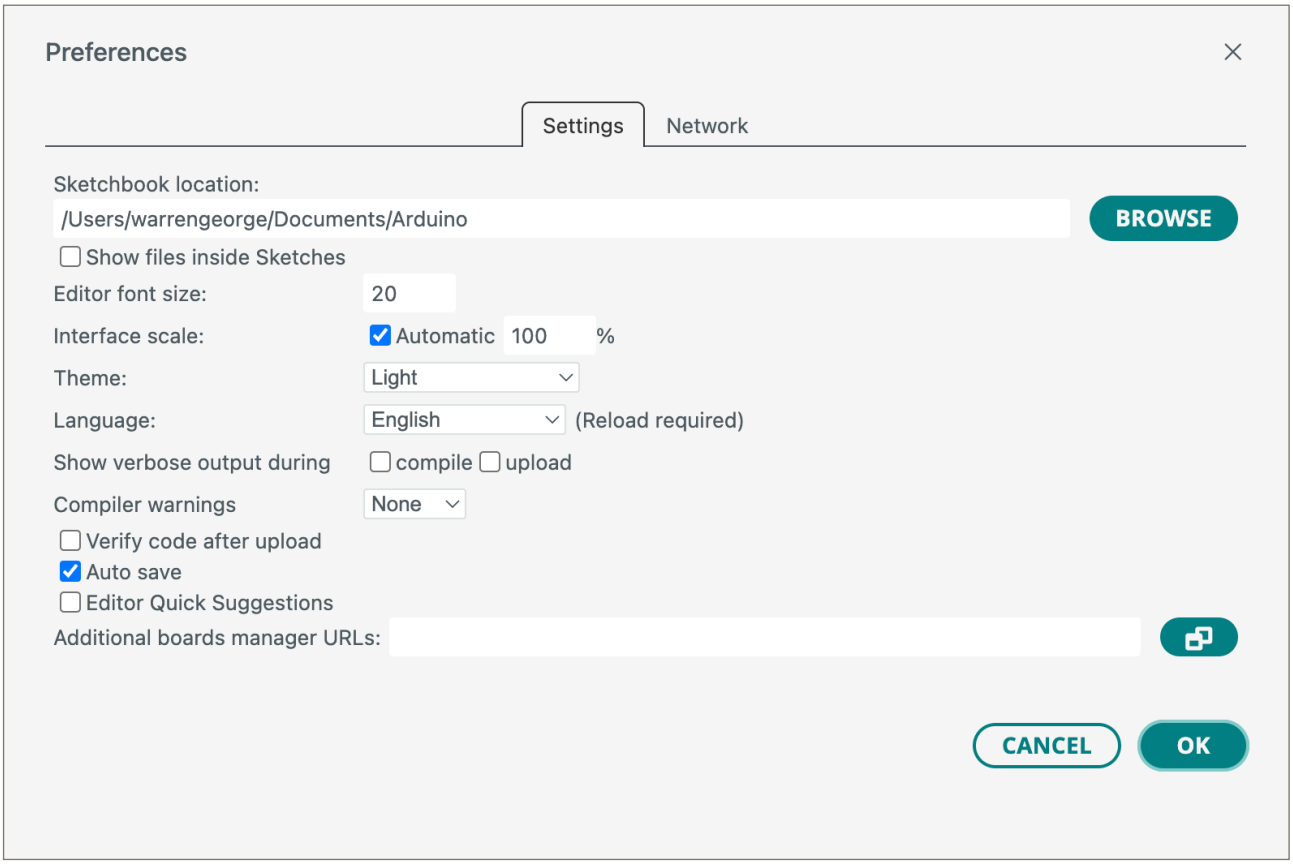


Settings

If you go up to the very top of the page and click on the words `Arduino IDE`, scroll down to `settings`, and here you can alter your editor to your own `preferences`. I have increased the font size to 20.

The `additional boards manager` is useful for when using non-Arduino boards, for instance, ESP32 boards; more on that at another time.

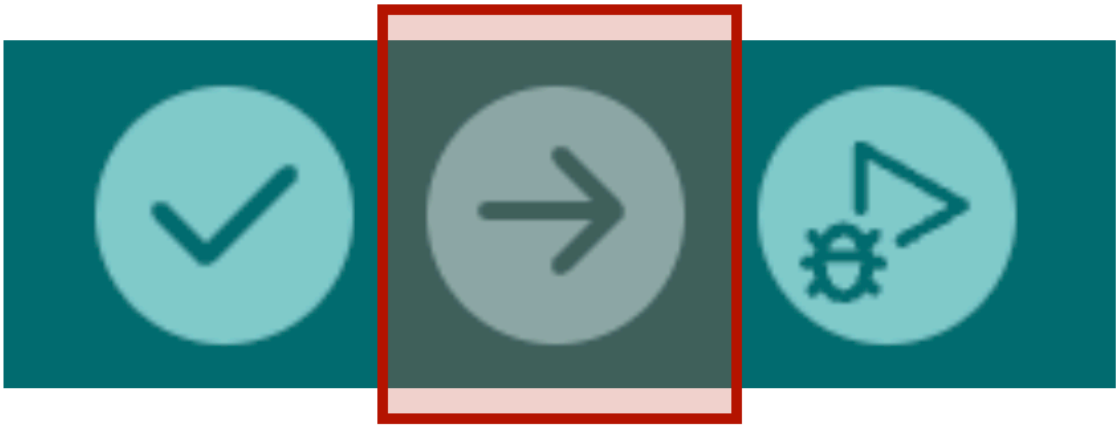
Figure 10: settings & preferences

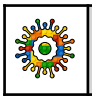


The buttons

Remember to click on the `upload` button (shown below, the middle one) once you have typed in the code for your first sketch. If you have missed something, even a semi-colon, it will tell you. Just make your corrections and try uploading again.

Figure 11: upload button

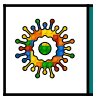




Introduction to blinking

Once you are set up. The very first sketch is very simple and will get you typing code. This may be very new to you, so don't worry; you can't break anything (at least not easily). The Arduino is very robust, and LEDs are very forgiving. Learn by doing, making mistakes, finding out what those mistakes are (debugging), and trying something new just to see what will happen. This is your playground; enjoy it and explore. I recommend that you try the challenges, but you don't have to; no one is watching.

1. Delete the default code.
2. Type in the new code.
3. Make sure your Nano is connected.
4. Check the board and port (under Tools).
5. Click on the upload button.
6. As you edit the code, you don't need to delete all the previous code every time.



Sketch A3.1 the LED is on

This will switch the red LED on. Type this code as is and upload (remember which button it is!). Once you click on the upload button, there will be a bit of activity, flashing of LEDs, and the red should turn on after a few seconds (or less). The word HIGH has to be in capitals and is another word meaning on (or 5 volts).

```
void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
}
```

Notes

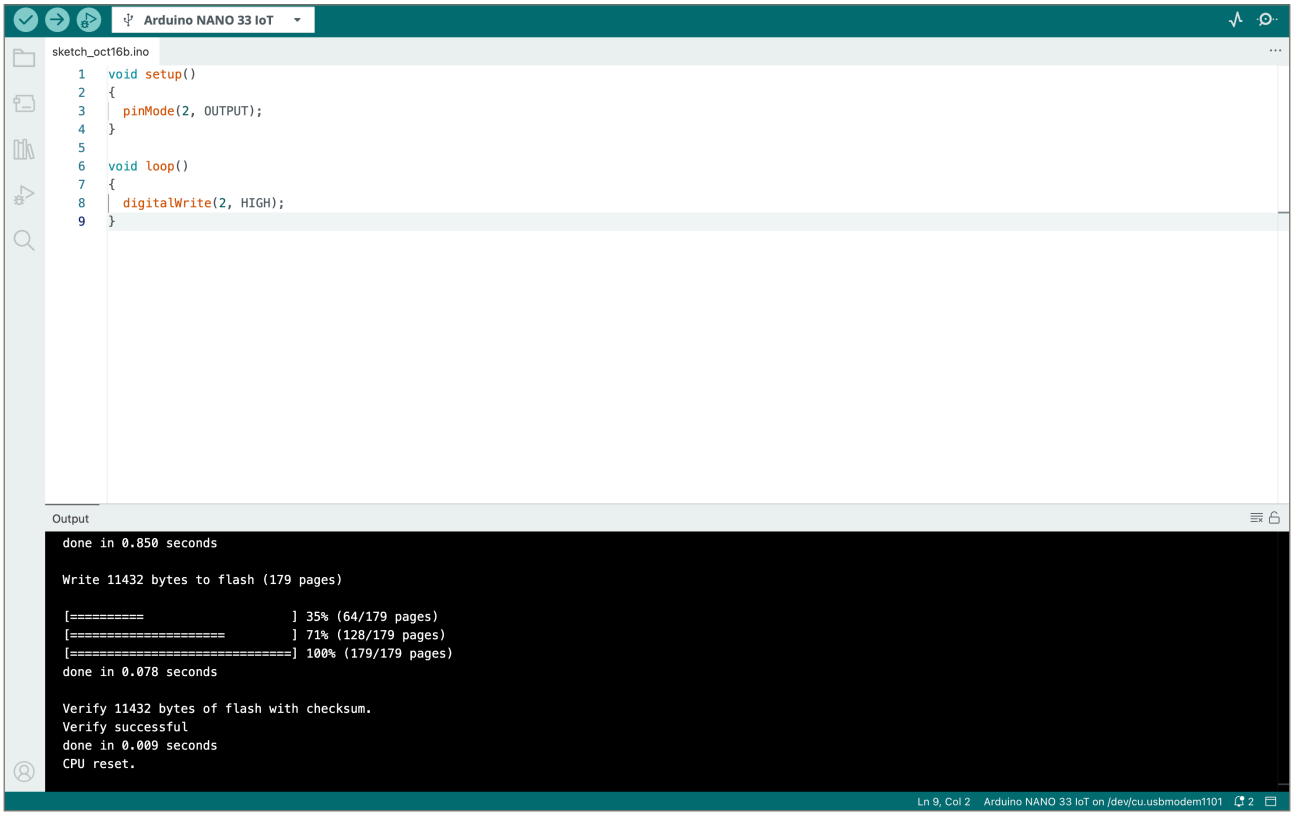
If the red LED isn't on, then you have the fun job of diagnosing the problem. Here are some suggestions.

1. Check that the jumper lead for the red LED is connected to pin D2.
2. Check that your ground is connected to the GND on the Nano.
3. If all else fails, just upload it again, checking your code for any minor mistakes, including semi-colons!

Code Explanation

`void setup()`	This happens once
`pinMode(2, OUTPUT);`	The pin on D2 is set to output as opposed to receiving an input from a device
`void loop()`	This is a continuous loop
`digitalWrite(2, HIGH);`	Then we tell pin D2 to be high which means 'on'

Figure A3.1


Arduino NANO 33 IoT
sketch_oct16b.ino
1 void setup()
2 {
3 pinMode(2, OUTPUT);
4 }
5
6 void loop()
7 {
8 digitalWrite(2, HIGH);
9 }
Output
done in 0.850 seconds
Write 11432 bytes to flash (179 pages)
[==========] 35% (64/179 pages)
[==========] 71% (128/179 pages)
[==============================] 100% (179/179 pages)
done in 0.078 seconds
Verify 11432 bytes of flash with checksum.
Verify successful
done in 0.009 seconds
CPU reset.
Ln 9, Col 2 Arduino NANO 33 IoT on /dev/cu.usbmodem1101

Sketch A3.2 the LED is off

This will switch the red LED off. The blue is the new bit of code, either to be added or changed (saves typing everything out again, but still good practice to retype everything as is below). Don't forget that keywords and functions are case-sensitive; that means if there are capital letters, then it is vital that you do the same. Hence, LOW is in capitals. The word LOW means 0 volts, or off.

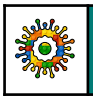
```
void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, LOW);
}
```

Notes

Your red LED should now switch off when you upload the code; it should be fairly instant.

Code Explanation

`digitalWrite(2, LOW);`	The output to pin 2 (D2) is set to LOW which means off



Sketch A3.3 blinking good

We will switch the LED on for `1` second, which is `1000` milliseconds. The `delay()` function is measured in milliseconds. Then switch off for `1` second and back on again, repeatedly. The `void loop()` function is just that; it is a loop that keeps going back to the start between the top curly bracket `{` and the bottom curly bracket `}`.

```
void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(1000);
  digitalWrite(2, LOW);
  delay(1000);
}
```

🗒️ Notes

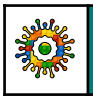
You will notice that it blinks on and off continuously. That is because it is in a loop and returns to the first line of the code inside the `void loop()` function, repeating indefinitely.

🌻 Challenge

Change the number of milliseconds.

🛠️ Code Explanation

`delay(1000);`	The delay function stops everything for the number of milliseconds



Sketch A3.4 adding a variable

❗ Please note nothing will change if you upload the sketch.

Introducing variables, there are many sorts, but the first one we will use is an integer. The word `int` is short for integer, which is a type of data. An integer is a whole number, e.g. 1, 2, 34, 87, etc. The word `delayPeriod` is a made-up word. We are going to give it an initial value of `500` milliseconds (half a second).

```
int delayPeriod = 500;


void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(1000);
  digitalWrite(2, LOW);
  delay(1000);
}
```

🛠️ Code Explanation

`int delayPeriod = 500;`	We have declared at the very beginning that delayPeriod is a variable, that it is an integer (int), and that it has a value of 500.

Sketch A3.5 using a variable

Instead of typing in the number every time we use delay, we can use the variable `delayPeriod`. This is useful if we want to change the `delay()` later on; we now only have to change it once.

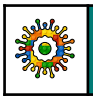
```
int delayPeriod = 250;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
}
```

Notes

Notice that it is blinking much faster now; each blink is a quarter of a second. Also notice that the way we named the variable means something. This is good practice rather than giving it a letter. It helps you later if you wonder what the variable is for, and others can come along later and look at your code. Also, if you combine two (or more) words, the first letter of the first word is lowercase, but the first letter of the other words is usually uppercase. This is the convention.

Code Explanation

`delay(delayPeriod);`	The variable replaces the fixed value



Sketch A3.6 adding a count

❗ Don't upload just yet.

Now, what if we wanted to stop after, say, 10 blinks? First, we need to count them. We introduce another variable called count (made-up word), we set it to 0, but in the loop, we add 1 each time it blinks.

```
int delayPeriod = 500;
int count = 0;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
  count = count + 1;
}
```

🛠️ Code Explanation

`int count = 0;`	Creating a variable called count and initialised to zero
`count = count + 1;`	One is added on each iteration of the loop

Sketch A3.7 the while() loop

Now this is the tricky bit, how does it know when there have been 10 blinks? We introduce a `while()` loop into the `void loop()`. In other words, it will loop through while the count is less than 10. That is what the < means. (You can cut and paste rather than deleting and rewriting the code inside the `while()` loop.) The counting starts with 0, not 1; hence, we stop before we get to ten, so it blinks the first time on 0.

```
int delayPeriod = 500;
int count = 0;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  while (count < 10)
  {
    digitalWrite(2, HIGH);
    delay(delayPeriod);
    digitalWrite(2, LOW);
    delay(delayPeriod);
    count = count + 1;
  }
}
```

🗒️ Notes

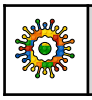
The red LED will blink `10` times and then switch off. The < is called a `comparison operator`. If you want to see it again without having to upload the sketch, then press the tiny grey button in the middle (ish) of the board just briefly.

🌻 Challenges

1. Change the number of times it blinks.
2. After a delay (pause), do another 10 blinks (repeated).

🛠️ Code Explanation

`while (count < 10)`	The while() loop happens for as long as the condition is true i.e. while count is less than 10.



Tidy up

If you cut and pasted the code, you might have seen that the code didn't line up nicely (shown below). You can tidy the code up if you are a bit OCD like me by clicking on the button on the right-hand side, or you can just do it manually with the space bar or the tab button on your keyboard.

I use two spaces for tabbing each block of code; that is my preference and seems to be the most common. Some use four spaces, and it will work even if you use none.

A quick way is to select the text, go to `Edit` (very top row), and click on `Increase Indent` for those lines of code. If you `Auto format`, the curly brackets will move also, which is fine as it is most people's preference.

Comparison Operators

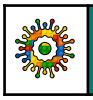
There are more of them. Below is a list of the `comparison operators`.

!=	not equal to
<	less than
<=	less than or equal to
==	equal to
>	greater than
>=	greater than or equal to

Arithmetic Operators

These are the `arithmetic operators`; some may be familiar and others not.

%	remainder
*	multiplication
+	addition
–	subtraction
/	division
=	assignment operator



Sketch A3.8 ten blinks

Another scenario is for there to be 10 blinks and then a break (of, say, 2 seconds) followed by another burst of 10 blinks, etc. For this, we will need an `if()` loop. So, remove the `while()` loop and start with the following code below. Don't upload just yet.

❗ decrease indent

```
int delayPeriod = 500;
int count = 0;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
  count = count + 1;
}
```

🗒️ Notes

Refactor the code so that it looks like above. All this will do is blink continuously (we're not finished yet!).

Sketch A3.9 the if() statement

Now, to add in the `if()` loop and change the starting count to `1`; otherwise, you will get `11` blinks, and we can't have that.

```
int delayPeriod = 500;
int count = 1;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
  if (count == 10)
  {
    count = 0;
    delay(2000);
  }
  count = count + 1;
}
```

🗒️ Notes

You should get 10 blinks, a short pause, and then ten more, and so on. We use two equal signs == because it is a comparison. If you use just one equal sign = (by mistake), it is an assignment. If you get an error message at this point, that is probably why.

🛠️ Code Explanation

`if (count == 10)`	An if() statement, if this is true (count is 10) then do that (whatever is in the loop

Sketch A3.10 compound operator

Introducing a compound operator. This one will appear quite often, ++. We will replace `count = count + 1;` with `count++;`. It does exactly the same thing, adding 1 each time to the `count` variable.

```
int delayPeriod = 500;
int count = 1;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
  if (count == 10)
  {
    count = 0;
    delay(2000);
  }
  count++;
}
```

🗒️ Notes

This is so very useful and very common, so it is worth introducing it now so that you can see how it works. Below I have included other ones that you will come across (but less so).

🛠️ Code Explanation

`count++;`	Shorthand for count plus 1

Compound Operators

These are very useful shortcuts.

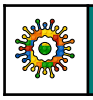
++	Incrementally adds 1
−−	Incrementally subtracts 1
+=	Adds a value e.g. x += 5 adds 5 to x each time
−=	Subtracts a value e.g. x -= 5 subtracts 5 from x each time
*=	Multiplies a value e.g. x *= 5 multiplies x by 5 each time
/=	Divides by a value e.g. x /= 5 divides x by 5 each time
%=	Remainder of a value e.g. x %= 5 will give you the remainder from divided by 5 each time

Sketch A3.11 another way

Introducing the `for()` loop. You will see this a lot in coding, so now is as good a time to introduce it. It may look a bit bewildering at first, but it is perfectly logical, and we will spend a bit of time helping you to become familiar with this very important loop. First, let us go back to this starting point.

❗ Remove all unnecessary code until you have the following...

```
int delayPeriod = 500;


void setup()
{
  pinMode(2, OUTPUT);
}


void loop()
{
  digitalWrite(2, HIGH);
  delay(delayPeriod);
  digitalWrite(2, LOW);
  delay(delayPeriod);
}
```



Sketch A3.12 the for() loop

The `for()` loop has three parts separated by semi-colons `;`.

1. The first part is to initialise a variable, which we will call `i`, and give it an initial value of `0`: `int i = 0;`
2. Next, we put a limit on how big `i` is going to get: `i < 10;`
3. Finally, we determine what increments we increase `i` by on each iteration (loop): `i++`

```
int delayPeriod = 500;

void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    digitalWrite(2, HIGH);
    delay(delayPeriod);
    digitalWrite(2, LOW);
    delay(delayPeriod);
  }
  delay(2000);
}
```

🗒️ Notes

This produces exactly the same result as the other loops in a completely different way.

🛠️ Code Explanation

`for (int i = 0; i < 10; i++)`	A for() loop, which acts as a counter for each loop, starts at zero, adds one each loop (iteration and continues until it reaches nine (not ten) which is ten loops altogether