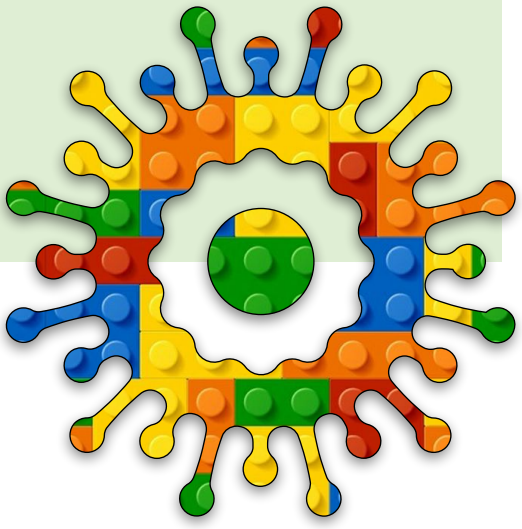
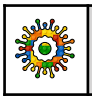
Introducing Robotics Module A Unit #2 Blinking





Content

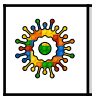
Module A Unit #2 blinking

What you will need

1 x Arduino Uno
1 x breadboard
1 x LED traffic light
4 x male to male jumper leads

The components are quite minimal for the first several units, and you will find out soon enough if you have connected everything correctly. Practice makes perfect; just keep trying and learning. You can't really break anything, so do not worry if you get wires in the wrong holes.

Also, if you are familiar with the p5.js tutorials you will see some very familiar features, I have included the explanations for those who might be starting here for the first time.



Introduction to Unit #2 Blinking

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.

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

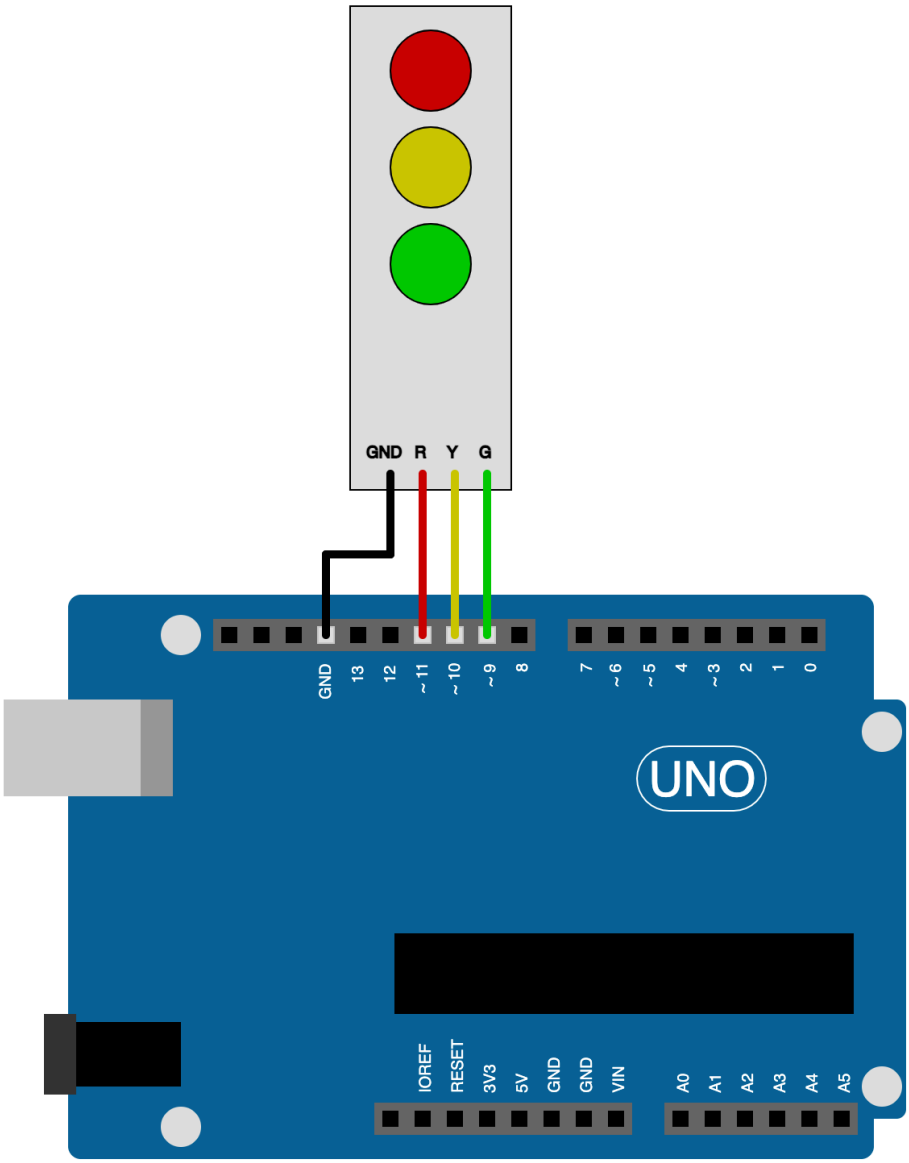
Circuit Diagram for unit #2 blinking

The diagram (see fig.1 below) and the photo (fig.2) will help you connect up your Traffic Lights LED modules. Take your time if you have never done this before. Take note that this will be the same circuit diagram for units #3 through to #6.

Traffic Light Pins	Arduino Pins
GND	GND
R (red)	11
Y (yellow)	10
G (green)	9

GND goes to GND
R (red) goes to digital pin 11
Y (yellow) goes to digital pin 10
G (green) goes to digital pin 9

Figure 1: Circuit Diagram (#1) LED Traffic Lights module

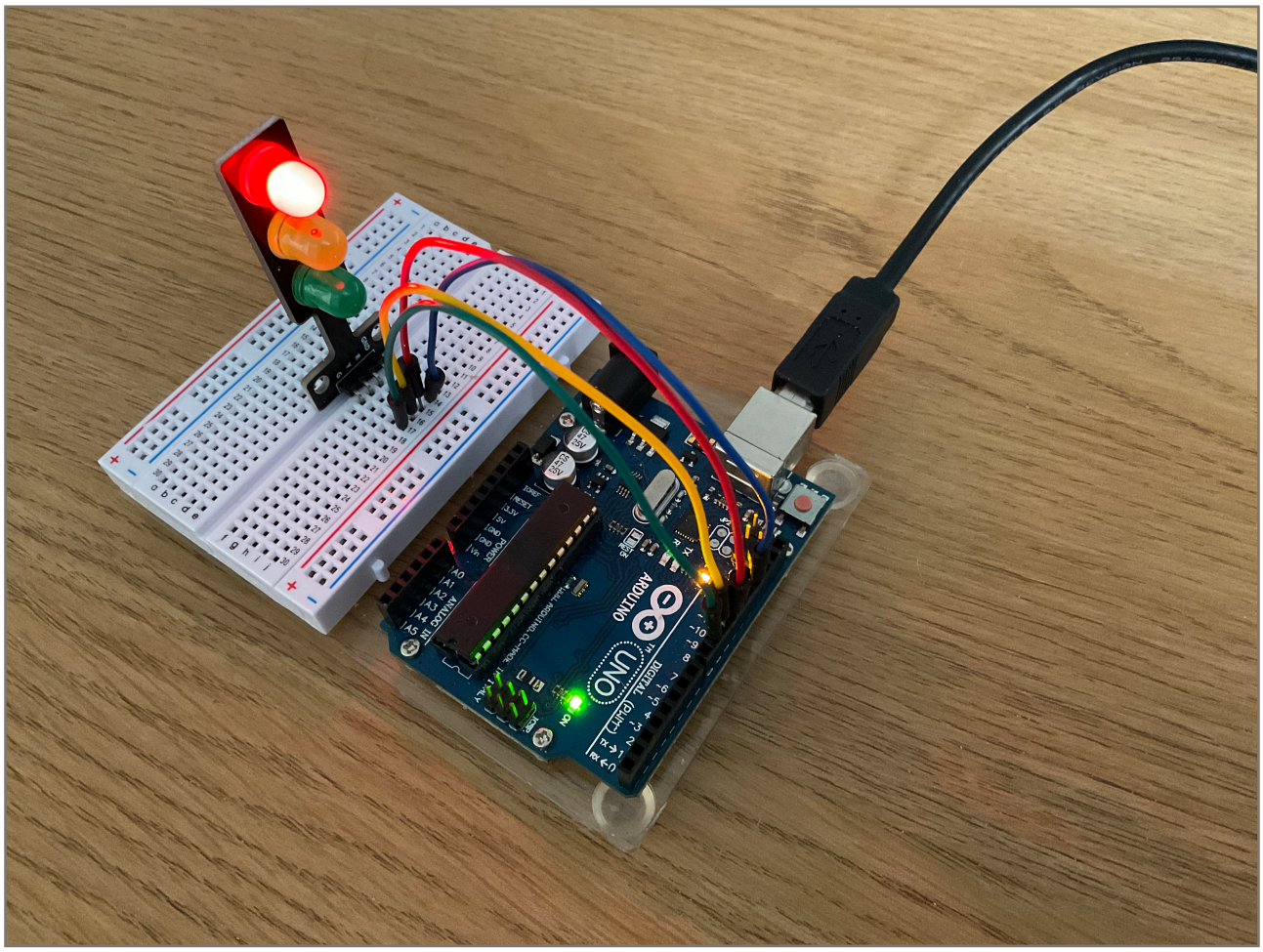


Connecting your Arduino Uno

The photo below shows one end of the USB cable connected to the Arduino Uno. The other end will connect to your computer. The LED Traffic Lights module sits on the breadboard, and the wires form the connection between the module pins and the Arduino.

The breadboard and Arduino Uno sit on an acrylic base that holds everything together neatly and tidily.

Figure 2: This is what it should look like, although the red LED most likely won't be on



Sketch A2.1 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(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
}
```

Notes

If it isn't lit, 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 11.
2. Check that your ground is connected to the GND on the Arduino.
3. Ensure that the jumper lead pin for the red LED lines up with the traffic light LED pin.
4. If all else fails, just upload it again, checking your code for any minor mistakes!

Challenges

Switch the other LEDs on pins 10 and 9 on, for instance:

```
pinMode(10, OUTPUT);
digitalWrite(10, HIGH);
```

🛠️ Code Explanation

`void setup()`	This happens once
`pinMode(11, OUTPUT);`	The pin on D11 is set to output as opposed to receiving an input from a device
`void loop()`	This is a continuous loop
`digitalWrite(11, HIGH);`	Then we tell pin 11 to be high which means 'on'

Figure A2.1a

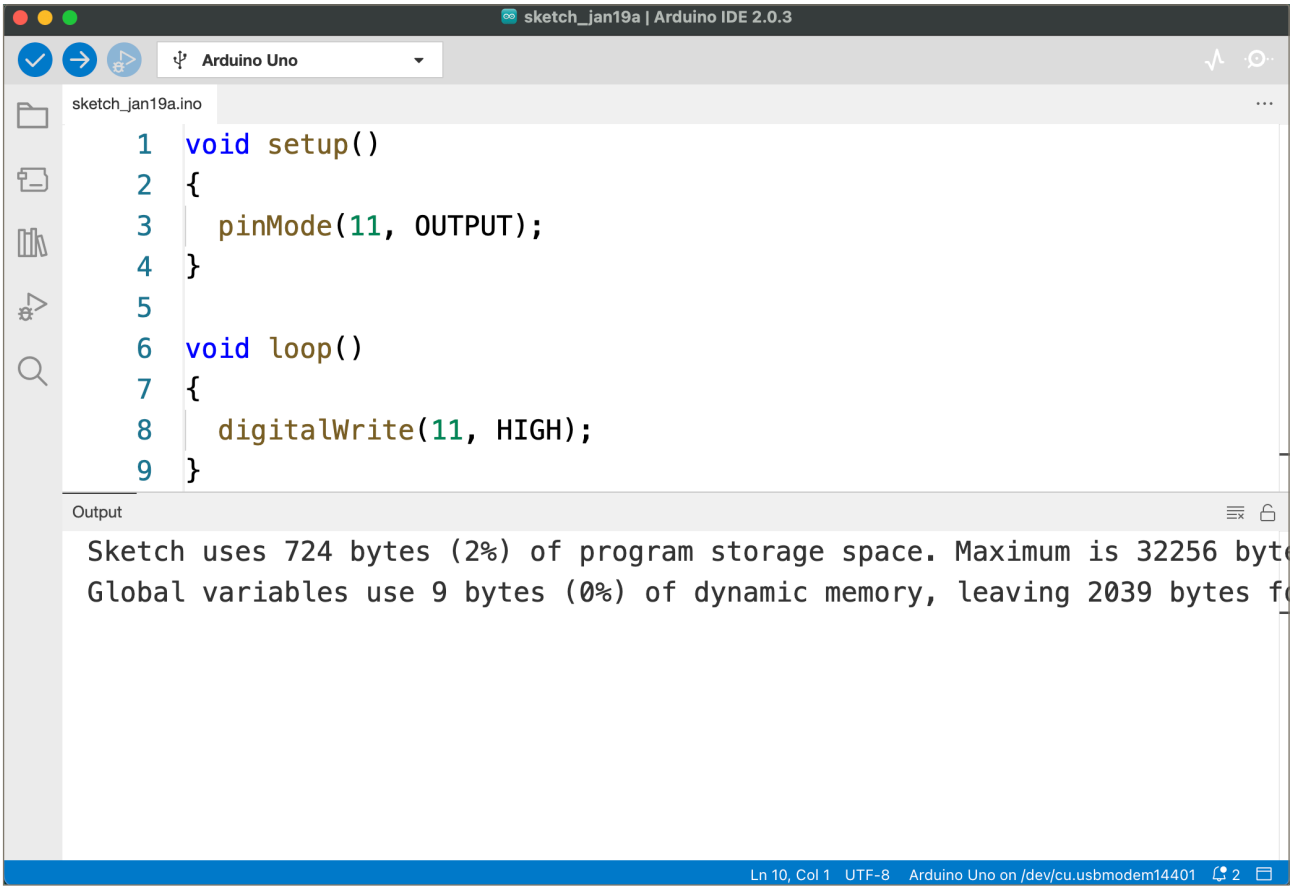
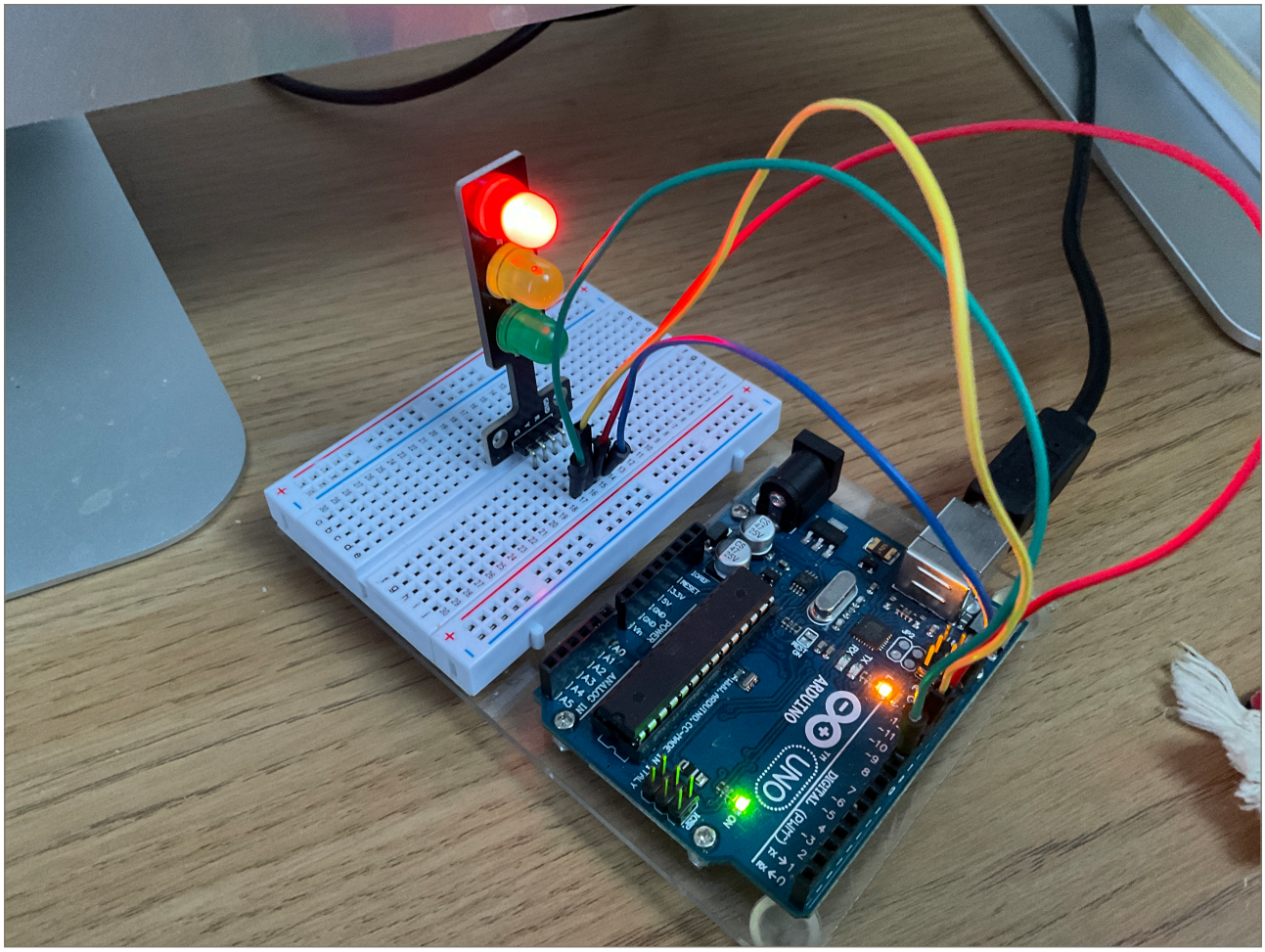

sketch_jan19a | Arduino IDE 2.0.3
Arduino Uno
sketch_jan19a.ino
1 void setup()
2 {
3 pinMode(11, OUTPUT);
4 }
5
6 void loop()
7 {
8 digitalWrite(11, HIGH);
9 }
Output
Sketch uses 724 bytes (2%) of program storage space. Maximum is 32256 byt
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes f
Ln 10, Col 1 UTF-8 Arduino Uno on /dev/cu.usbmodem14401 2

Figure A2.1b



Sketch A2.2 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 is a capital letter(s), 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(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, LOW);
}
```

Notes

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

Code Explanation

`digitalWrite(11, LOW);`	The output to pin 11 (D11) is set to LOW which means off

Sketch A2.3 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()` 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(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(1000);
  digitalWrite(11, 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.

🌻 Challenges

1. Change the number of milliseconds.
2. Blink the green or the yellow.

🛠️ Code Explanation

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

Sketch A2.4 variables

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 250.

```
int delayPeriod = 250;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(1000);
  digitalWrite(11, LOW);
  delay(1000);
}
```

🛠️ Code Explanation

`int delayPeriod = 250;`	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 250.

Sketch A2.5 the delay

Instead of typing in the number every time, 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(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, 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 who 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 A2.6 count

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

```
int delayPeriod = 250;
int count = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
  count = count + 1;
}
```

🌻 Challenges

Instead of `count = count + 1`, you could use `count++`, as a shortcut.

🛠️ 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 A2.7 while loop

Now this is the tricky bit: how does it know when there have been ten blinks? We introduce a `while()` loop into the `void loop()`. In other words, it will loop through while the count is less than ten. That is what the `<` means.

```
int delayPeriod = 250;
int count = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  while (count < 10)
  {
    digitalWrite(11, HIGH);
    delay(delayPeriod);
    digitalWrite(11, LOW);
    delay(delayPeriod);
    count = count + 1;
  }
}
```

Notes

The red LED will blink ten times and then switch off. The `<=` is called a comparison operator.

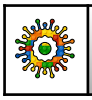
Challenges

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

3. If you change the `while()` loop conditional to `<=` (less than or equal to), what else would you have to change?

🛠️ Code Explanation

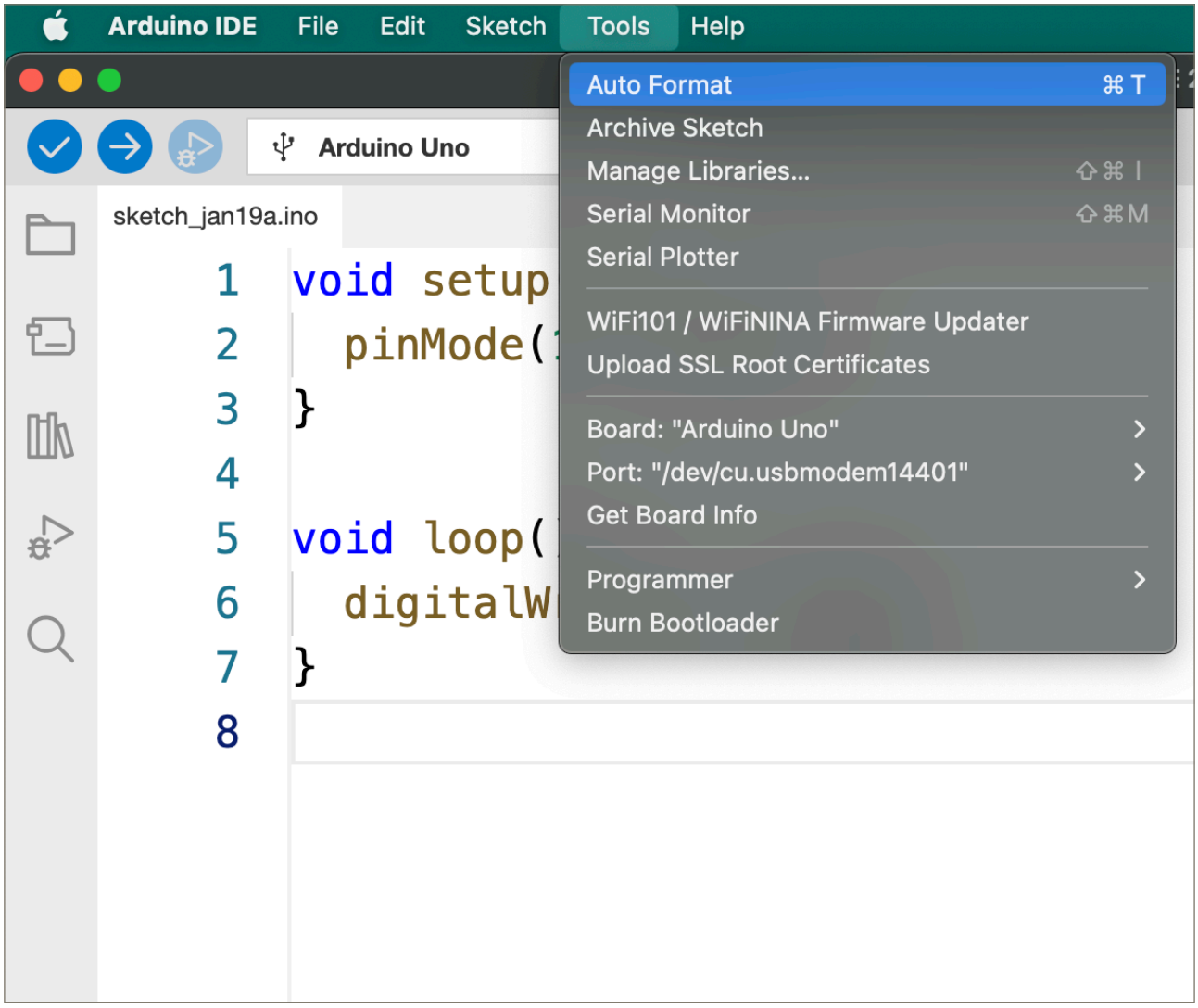
`while (count < 10)`	The while() loop happens for as long as the condition is true i.e. while count is less than 10.



Tidy the code

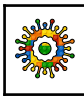
If the code gets a bit messy and you are prone to being a little bit OCD (like me) towards having it look nice and tidy, then you can use the Auto Format tab/button at the top under Tools. See Fig.3 below.

Figure 3: Click on tools, then Auto Format to line everything up nicely



You will find out that it moves your curly brackets (braces) into a different position. Many people code the way I have shown you, but also many code the way the auto format shows you. The way I have shown you in the code, I think, is a bit clearer so you can see it line by line, but the automated way is equally good and you might prefer to do it that way (see Fig.4). The choice is yours.

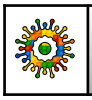
Figure 4: The auto format is another popular way to write the code, both ways are perfectly acceptable but you may find examples in this format



Comparison Operators

There are more of them; below is a list of some of the comparison operators.

!=	not equal to
<	less than
<=	less than or equal to
==	equal to (not to be confused with =)
>	greater than
>=	greater than or equal to



Arithmetic Operators

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

*	multiplication
+	addition
–	subtraction
/	division
=	assignment operator (equals, not to be confused with ==)
%	modulo (remainder)

Sketch A2.8 the if statement

Another scenario is for there to be ten blinks and then a break (of, say, two seconds) followed by another burst of ten blinks, etc. For this, we will need an `if()` loop.

❗ Remove the `while loop()` and start with this...

```
int delayPeriod = 250;
int count = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
  count = count + 1;
}
```

Notes

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

Sketch A2.9 adding the loop

Now, to add in the `if()` loop.

```
int delayPeriod = 250;
int count = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
  count = count + 1;
  if (count == 10)
  {
    count = 0;
    delay(2000);
  }
}
```

Notes

You should get ten blinks, a short pause, and then ten more, and so on. We use two equals signs `==` because it is a comparison. If you use just one equal sign `=`, 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 A2.10 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 one each time to the count variable.

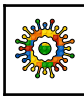
```
int delayPeriod = 250;
int count = 0;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
  count++;
  if (count == 10)
  {
    count = 0;
    delay(2000);
  }
}
```

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 others that you will come across (but less so).

🛠️ Code Explanation

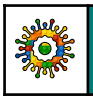
`count++;`	Shorthand for count plus 1



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
%=	Multiplies a value e.g. x *= 5 multiplies x by 5 each time

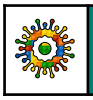


Sketch A2.11 the for loop

Introducing a `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 = 250;

void setup()
{
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(11, HIGH);
  delay(delayPeriod);
  digitalWrite(11, LOW);
  delay(delayPeriod);
}
```



Sketch A2.12 adding another loop

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

A. The first part is to initialise a variable which we will call `i` and give it an initial value of `0` (`int i = 0;`).
B. Next, we put a limit on how big `i` is going to get (`i < 10;`).
C. Finally, we increment `i` by `1` on each iteration of the loop (`i++`).

```
int delayPeriod = 250;


void setup()
{
  pinMode(11, OUTPUT);
}


void loop()
{
  for (int i = 0; i < 10; i++)
  {
    digitalWrite(11, HIGH);
    delay(delayPeriod);
    digitalWrite(11, LOW);
    delay(delayPeriod);
  }
  delay(2000);
}
```

🗒️ Notes

This produces exactly the same result as the other loops in a completely different way (remember to use the `Auto Format` in Tools).

🛠️ 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