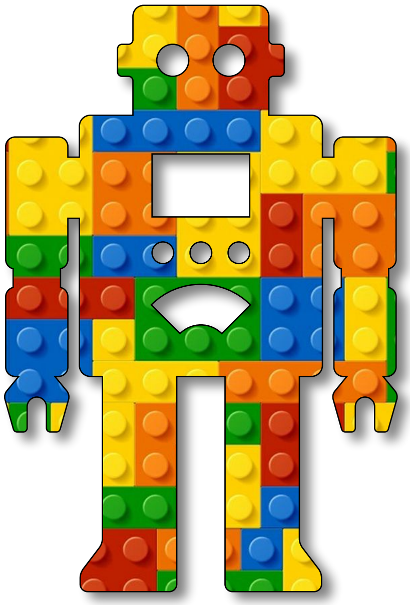
Intelligent Machines Module A Unit #3 blinking



Content

Module A Unit #3 Blinking

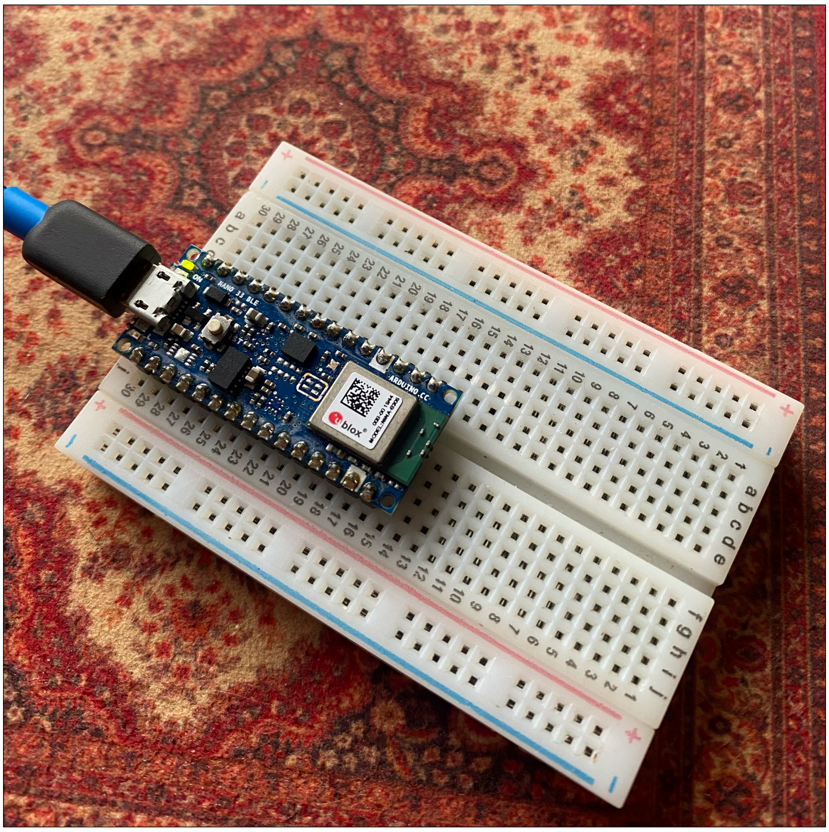
Introduction to blinking

Built-in LED: This is the Hello World! of coding a microcontroller, to blink an LED. The beauty of this particular board is that it has two other LEDs (apart from the power-on LED). One is situated on the other side of the USB connector. This LED is bright yellow when it is on.

RGB LED: The other LED is next to the RESET button and is an RGB LED, which means we can give it whatever colour we like, red, green, or blue, and a few colours in between.

Connect your Arduino Nano 33 BLE to your computer's USB port like so:

Figure 1: power LED should be on



Introduction to blinking

The first sketch is a simple sketch where we are going to turn on the built-in LED. The built-in LED is also connected to digital pin 13 called D13, although we just use 13 without the D (it seems to know what you mean). We can also call it BUILTIN_LED, but for now, we will just use pin 13 as a reference.

Step 1: Delete the default code.

Step 2: Type in the code.

Step 3: Make sure your Arduino is connected.

Step 4: Check the board and port (under Tools).

Step 5: Press the upload button.

Step 6: Once everything is set up and working, you only need to do steps 2 and 5.

Sketch A3.1 LED on

As we have already stated the built-in LED is attached to pin 13. The following sketch will switch the LED on. Firstly, type the code below into the IDE as is and press the upload button (middle right pointing arrow). Make sure that you put the colons (;) and curly brackets { } where they should be.

❗ You may need to press the RESET button (grey) on the device after you have uploaded. Be aware that everything is case sensitive.

Arduino sketch

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
}
```

🗒 Notes

The first thing that happens is you get a little box appearing, which tells you it is compiling the code. This is when it might throw up any errors you have (if any). Once compiled, it will then upload, and you will be told when the upload is complete. At the same time, you will get a lot of information being printed to the console at the bottom.

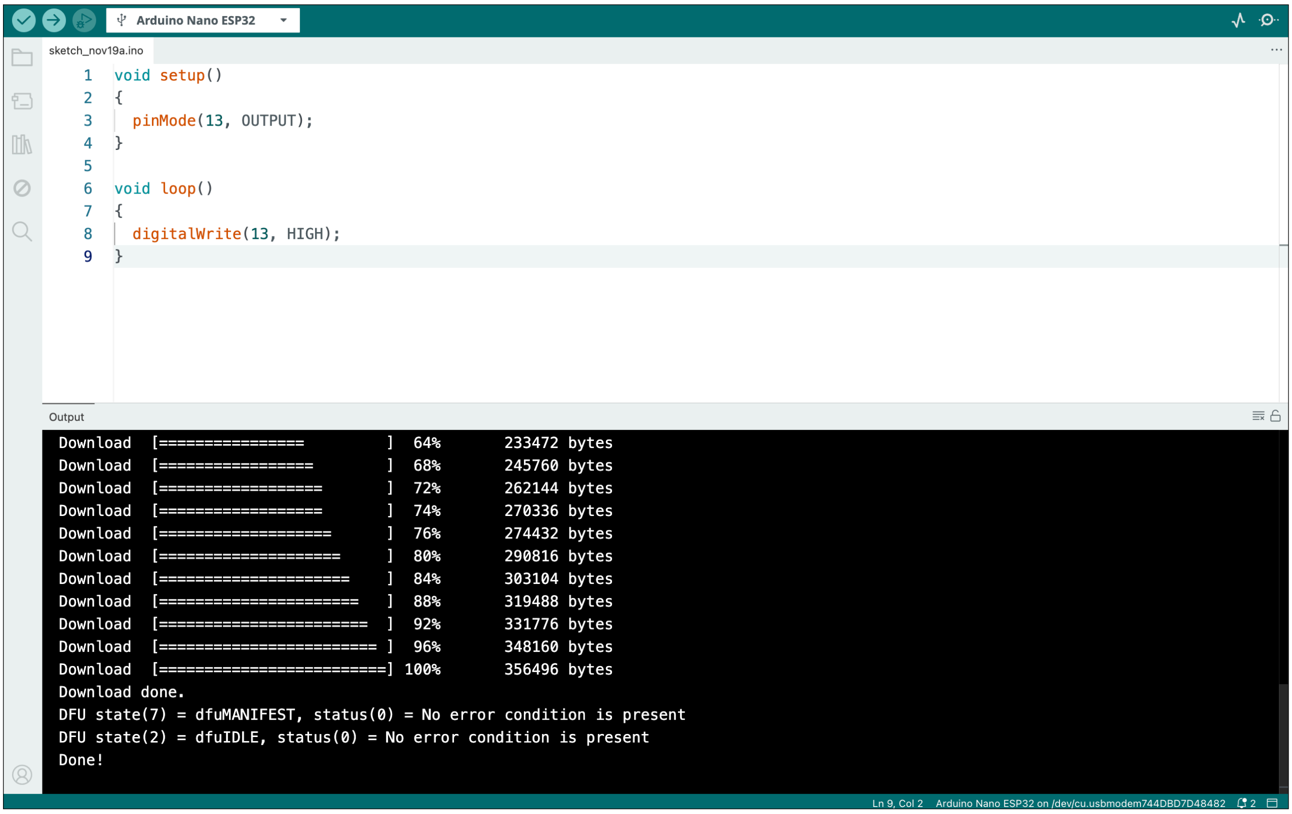
🌻 Challenge

Replace the 13 with LED_BUILTIN.

🛠 Code Explanation

`void setup()`	This happens once
`pinMode(13, OUTPUT);`	The pin on D13 is set to output as opposed to receiving an input from a device
`void loop()`	This is a continuous loop
`digitalWrite(13, HIGH);`	Then we tell pin 13 to be high which means ‘on’

Figure 2: after you have uploaded the code successfully



```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
}
```

```
Download  [================         ]  64%       233472 bytes
Download  [=================        ]  68%       245760 bytes
Download  [==================       ]  72%       262144 bytes
Download  [==================       ]  74%       270336 bytes
Download  [===================      ]  76%       274432 bytes
Download  [====================     ]  80%       290816 bytes
Download  [=====================    ]  84%       303104 bytes
Download  [======================   ]  88%       319488 bytes
Download  [=======================  ]  92%       331776 bytes
Download  [======================== ]  96%       348160 bytes
Download  [=========================] 100%       356496 bytes
Download done.
DFU state(7) = dfuMANIFEST, status(0) = No error condition is present
DFU state(2) = dfuIDLE, status(0) = No error condition is present
Done!
```

Ln 9, Col 2   Arduino Nano ESP32 on /dev/cu.usbmodem744DBD7D48482   2

Sketch A3.2 LED off

Now we have switched the LED on we want to switch the LED off. The blue highlighted line is the new bit of code, either to be added or changed (saves typing everything out again). Remember that you may need to press the RESET button (grey) on the Arduino after you have uploaded.

Arduino sketch

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, LOW);
}
```

Notes

The LED should be switched off. Also notice that even for a minor change to the code, the compiling and uploading seem to take an eternity. This is one of the drawbacks of the Arduino IDE, but one we have to live with when using certain boards.

Code Explanation

`digitalWrite(13, LOW);`	The output to pin 13 (D13) is set to LOW which means off

Sketch A3.3 blinking good

For the next step we will switch the LED on for 1 second, which is 1000 milliseconds. The delay() function is measured in milliseconds. Then, switch it off for 1 second and back on again.

❗ If it doesn't start blinking, then press the reset button.

Arduino sketch

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

🗒️ Notes

You will notice that it blinks on and off continuously. This is because it is in a loop and returns to the first line of the code inside the void loop() function, repeating indefinitely (similar to function draw() in p5.js).

🌻 Challenge

Change the number of milliseconds from 1000 to 500.

🛠️ Code Explanation

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

Sketch A3.4 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 variable name. We are going to give it an initial value of 500.

Arduino sketch

```
int delayPeriod = 500;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Notes

This doesn't do or change anything yet. We will make use of this variable soon.

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 500.

Sketch A3.5 using a variable

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

Arduino sketch

```
int delayPeriod = 500;


void setup()
{
  pinMode(13, OUTPUT);
}


void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
}
```

🗒️ Notes

Notice that it is blinking much faster now; each blink is half 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 one of the conventions commonly used.

🛠️ Code Explanation

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

Sketch A3.6 count

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

Arduino sketch

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

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
  count = count + 1;
}
```

Notes

This has no effect yet except to count.

Challenges

1. If you have done p5.js coding, you will know a shorthand version to add 1 to a variable.
2. Also, can you think of a way of stopping it after 10 blinks?

🛠️ 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 while loop

Now this is the tricky bit: how does it know when there have been 10 blinks? We introduce a while() loop straight into the void loop(). In other words, it will loop through while the count is less than 10. That is what the < means.

Arduino sketch

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

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  while (count <= 10)
  {
    digitalWrite(13, HIGH);
    delay(delayPeriod);
    digitalWrite(13, LOW);
    delay(delayPeriod);
    count = count + 1;
  }
}
```

Notes

To reorganise the code so it is formatted nicely, go to Tools and click Auto Format. The < is called a comparison operator.

Notice that when you auto-format it changes the overall format to:

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

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

void loop() {
  while (count <= 10) {
    digitalWrite(13, HIGH);
    delay(delayPeriod);
    digitalWrite(13, LOW);
    delay(delayPeriod);
    count = count + 1;
  }
}
```

🌻 Challenges

Change the number of times it blinks.

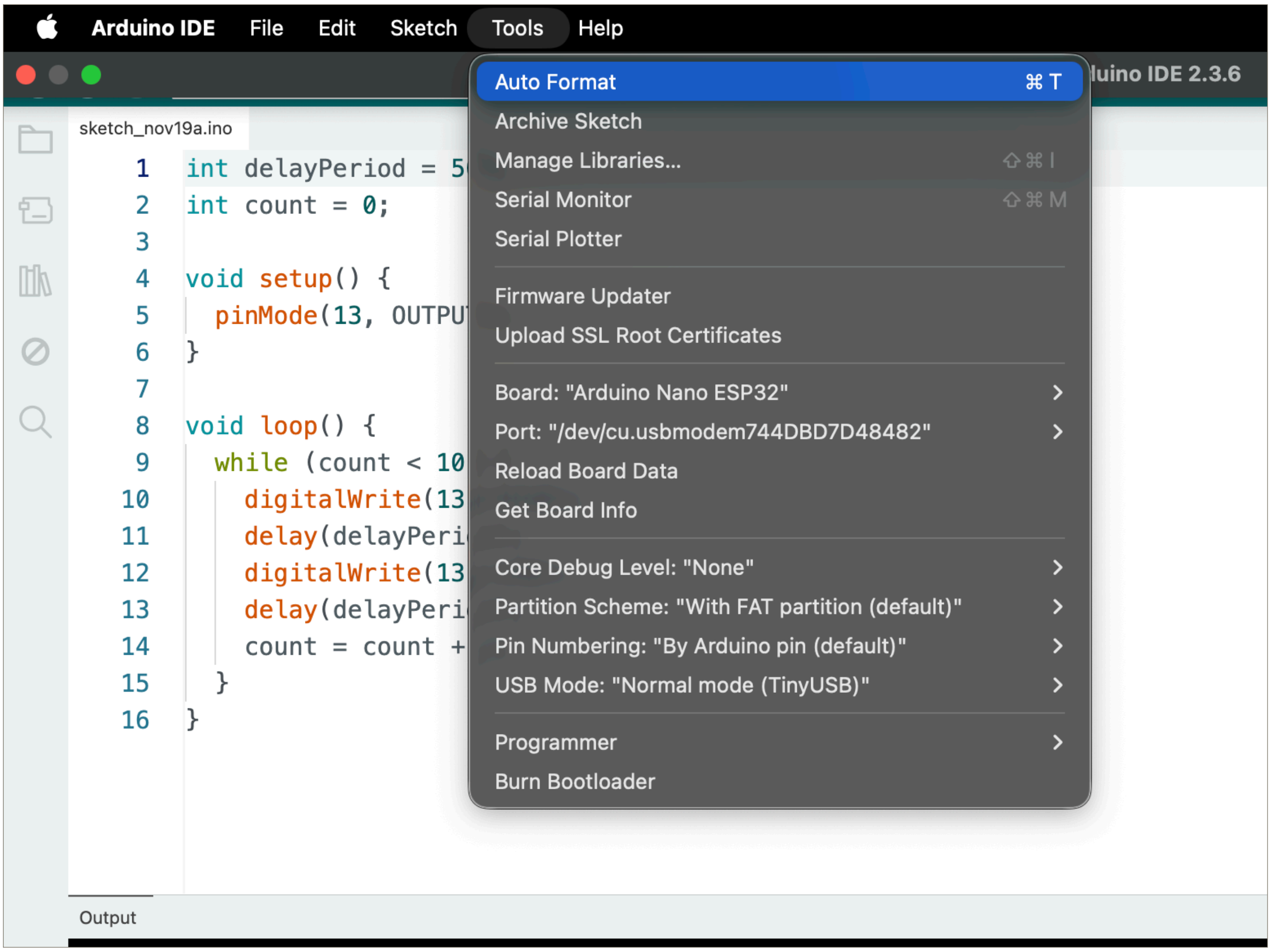
Question: Why does it blink 10 times when it stops before it reaches 10? Surely it should blink 9 times?

Answer: The counting starts from 0, not 1, so the first blink is when the count is 0.

🛠 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.

Figure 3: Auto Format



```
int delayPeriod = 5
int count = 0;

void setup() {
  pinMode(13, OUTPU
}

void loop() {
  while (count < 10
    digitalWrite(13
    delay(delayPeri
    digitalWrite(13
    delay(delayPeri
    count = count +
  }
}
```

Formatting: The change in format is really what it should look like. If you prefer that format (and most do), then please code that way. I only do it my way to make the code a bit clearer to read; it is a trade-off.

Troubleshooting: If, like me, you can't upload the new code for some reason, a simple solution is to disconnect the board from the computer and connect it again. This seems to do the trick. The board can be a bit temperamental.

Another solution is to double-click on the reset button and then upload the code. If this fails, try uploading a simple blink sketch or check that you haven't made some error, such as getting the for loop variables wrong and entering into an infinite loop, as I did!!

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, one second) followed by another burst of 10 blinks, etc. For this, we will need an if() loop.

! Remove the while() loop and start with this...

Arduino sketch

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

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
  count = count + 1;
}
```

Sketch A3.9 adding the if()

Now to add in the if() loop.

Arduino sketch

```
int delayPeriod = 250;
int count = 1;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
  count = count + 1;
  if (count == 10)
  {
    count = 0;
    delay(1000);
  }
}
```

Notes

You should get 10 blinks, a 1 second pause, and then 10 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 a mathematical 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

As I alluded to earlier, 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.

Arduino sketch

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

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
  count++;
  if (count == 10)
  {
    count = 0;
    delay(1000);
  }
}
```

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 (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 introducing 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.

Arduino sketch

```
int delayPeriod = 500;


void setup()
{
  pinMode(13, OUTPUT);
}


void loop()
{
  digitalWrite(13, HIGH);
  delay(delayPeriod);
  digitalWrite(13, LOW);
  delay(delayPeriod);
}
```

Sketch A3.12 a loop of three parts

The `for(int i = 0; i < 10; i++)` loop has three parts separated by semi-colons (`;`).

1. The first part initialises a variable which we will call `i` and gives it an initial value of `0` but it can be any value: `int i = 0;`
2. Next, we put a limit on how big `i` is going to get in this example: `i < 10;`
3. Finally, we specify what increments we increase `i` by on each iteration (loop), we could specify increments of any value: `i++`

Arduino sketch

```
int delayPeriod = 500;

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 10; i++)
  {
    digitalWrite(13, HIGH);
    delay(delayPeriod);
    digitalWrite(13, LOW);
    delay(delayPeriod);
  }
  delay(1000);
}
```

🗒️ Notes

This produces exactly the same result as the `while()` loop and `if()` statement in a completely different way. You can indent inside the loop, but it is not essential. If you think about it, we initialise the count (called `i`) to `0`, we have a conditional statement `<` less than `10`, and finally we increment the count (`i`) by `1`. We do all of that in one line of code.

🛠️ 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