

Electronics
Programming 101

Programming 101>> Software vs hardware

software | hardware

The cables, the loads and the breadboard are our hardware.

The arduino board is also hardware, but it is the computer controlling the hardware. We tell the arduino **how** to power and control the circuits using a **computer programme**, which is a software.

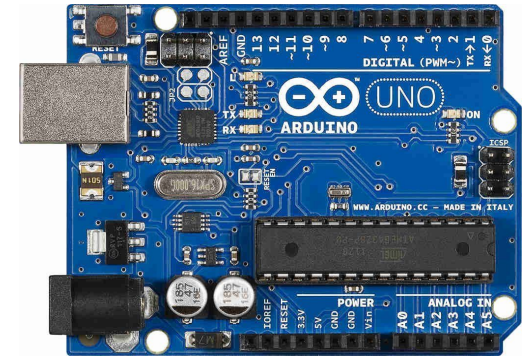
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
  */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```



Programming 101>> Software vs hardware

software | hardware

Programmes are the way we tell computers what we want them to do and how they should behave.

There are different programming languages, and like spoken languages each has its own rules, and there are different ways to tell the computer to do things.

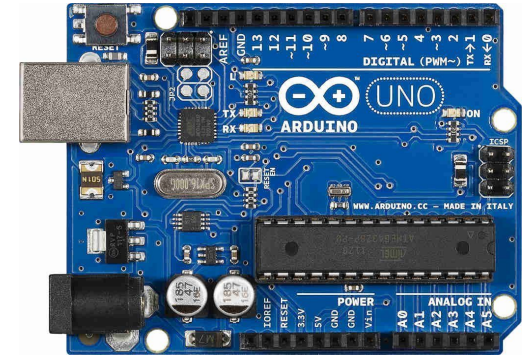
Gcode, Python etc

```
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

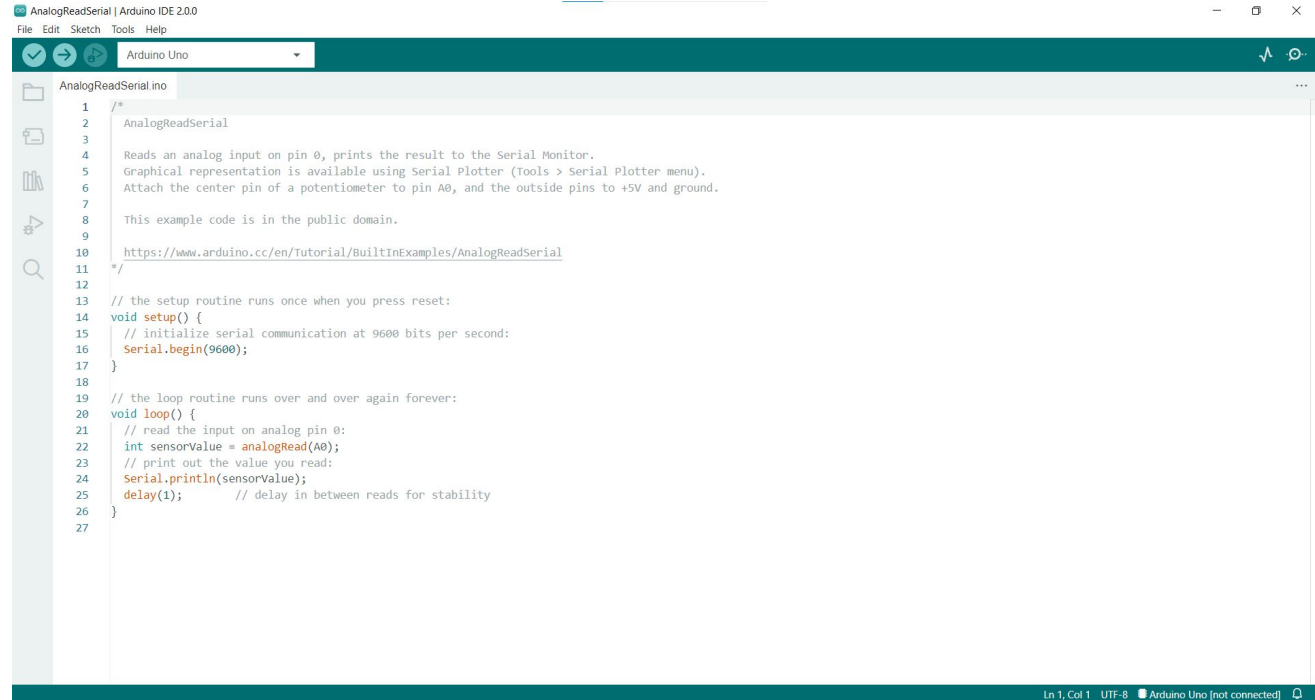


Programming 101>> Arduino IDE

The Arduino IDE (integrated development environment) is an open-source platform where we write, test and upload programmes to the board.

It can be downloaded here:
<https://www.arduino.cc/en/software>

It uses a programming language called C++.

The screenshot shows the Arduino IDE 2.0.0 interface. The title bar reads "AnalogReadSerial | Arduino IDE 2.0.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening files, saving, and running. A dropdown menu shows "Arduino Uno". The main editor area displays the code for "AnalogReadSerial.ino". The code includes a comment block describing the sketch, followed by the setup and loop functions. The status bar at the bottom indicates "Ln 1, Col 1 - UTF-8" and "Arduino Uno [not connected]".

```
1  /*
2  AnalogReadSerial
3
4  Reads an analog input on pin 0, prints the result to the Serial Monitor.
5  Graphical representation is available using Serial Plotter (Tools > Serial Plotter menu).
6  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.
7
8  This example code is in the public domain.
9
10 https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogReadSerial
11 */
12
13 // the setup routine runs once when you press reset:
14 void setup() {
15   // initialize serial communication at 9600 bits per second:
16   Serial.begin(9600);
17 }
18
19 // the loop routine runs over and over again forever:
20 void loop() {
21   // read the input on analog pin 0:
22   int sensorValue = analogRead(A0);
23   // print out the value you read:
24   Serial.println(sensorValue);
25   delay(1);        // delay in between reads for stability
26 }
27
```

Programming 101>> Code structure

A programme is made of three parts:

Definitions

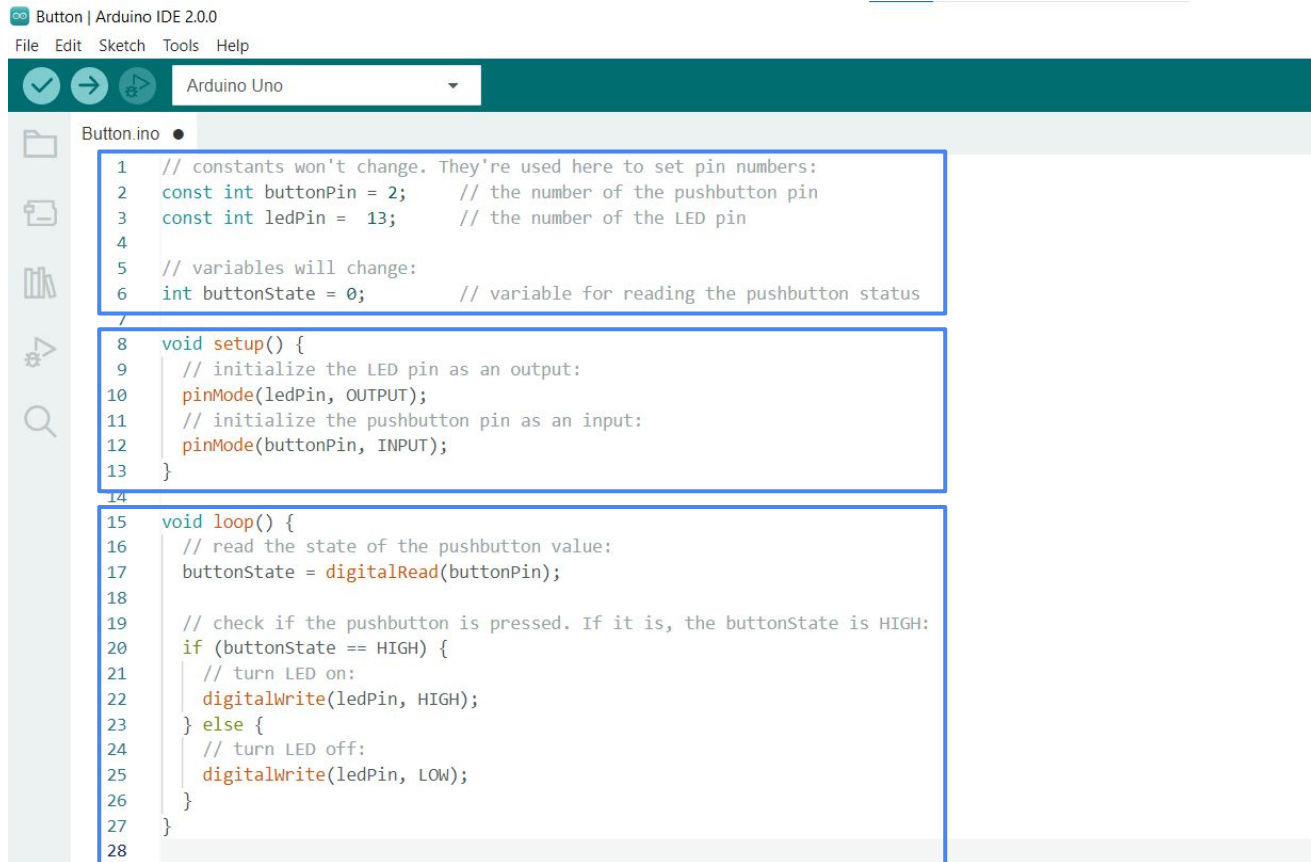
This is where we set variables, create objects and import libraries

Setup()

Is code that we run once at the beginning of each programme. Setting the pin modes, starting serial monitor etc

Loop()

The main part of the code, that runs forever



```
Button | Arduino IDE 2.0.0
File Edit Sketch Tools Help

Button.ino
1 // constants won't change. They're used here to set pin numbers:
2 const int buttonPin = 2;    // the number of the pushbutton pin
3 const int ledPin = 13;     // the number of the LED pin
4
5 // variables will change:
6 int buttonState = 0;        // variable for reading the pushbutton status
7
8 void setup() {
9   // initialize the LED pin as an output:
10  pinMode(ledPin, OUTPUT);
11  // initialize the pushbutton pin as an input:
12  pinMode(buttonPin, INPUT);
13 }
14
15 void loop() {
16   // read the state of the pushbutton value:
17   buttonState = digitalRead(buttonPin);
18
19   // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
20   if (buttonState == HIGH) {
21     // turn LED on:
22     digitalWrite(ledPin, HIGH);
23   } else {
24     // turn LED off:
25     digitalWrite(ledPin, LOW);
26   }
27 }
28
```

Programming 101>> Functions

Functions are used to tell the board to do something.

For example:

`delay()` is used to tell the programme to wait before it does something

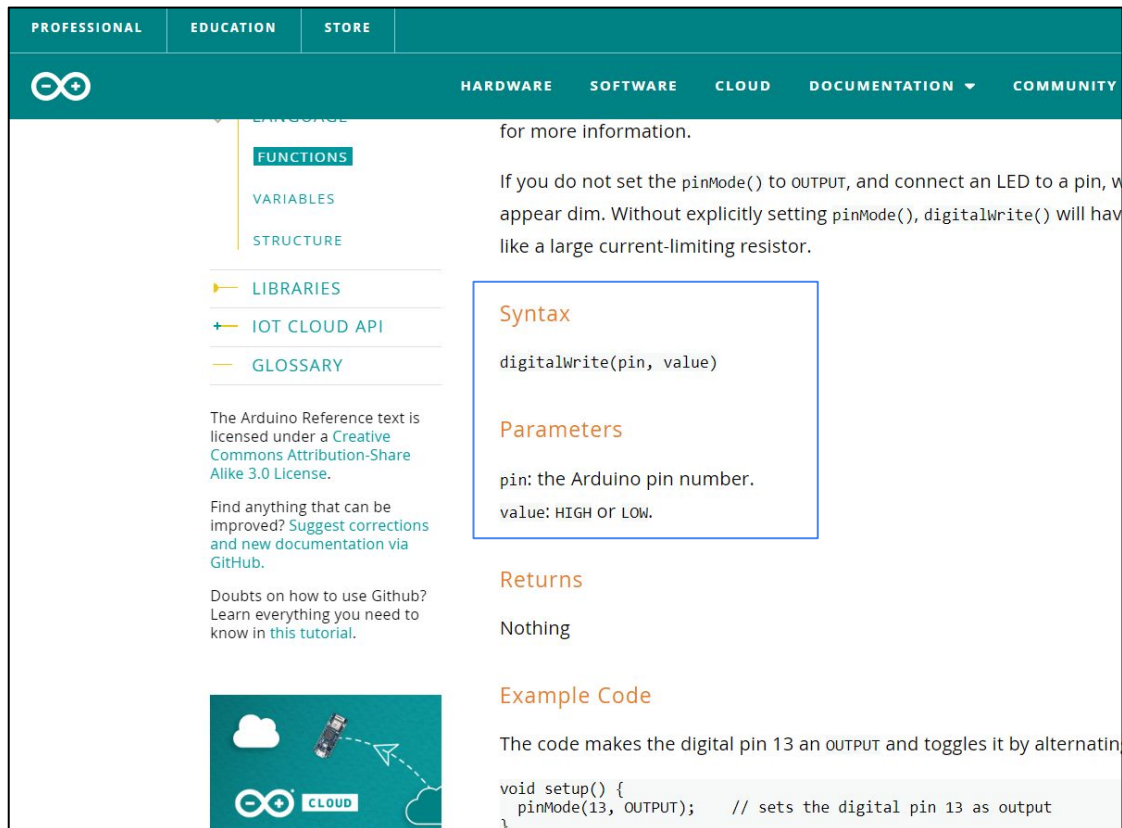
`digitalWrite()` is used to tell a digital pin to send a current, or not to.

Programming 101>> Functions

Each function has specific syntax, which determines what information we need to give it in order to execute the operation. The `digitalWrite()` function will need to know what pin we want to control, and whether we want to send current or not.

The correct syntax is:
`digitalWrite(pin, value)`

The syntax for each function can be found on the arduino website:
<https://www.arduino.cc/reference/en/>



The screenshot shows the Arduino Reference website. The top navigation bar includes links for PROFESSIONAL, EDUCATION, STORE, and a main menu with HARDWARE, SOFTWARE, CLOUD, DOCUMENTATION, and COMMUNITY. The left sidebar contains a navigation menu with LANGUAGE, FUNCTIONS (highlighted), VARIABLES, STRUCTURE, LIBRARIES, IOT CLOUD API, and GLOSSARY. The main content area displays the `digitalWrite()` function page. It includes a brief description, a note about the Creative Commons license, and a link to suggest improvements. The function's syntax is shown as `digitalWrite(pin, value)`. The parameters section explains that 'pin' is the Arduino pin number and 'value' is HIGH or LOW. The Returns section states 'Nothing'. The Example Code section shows a code snippet that sets pin 13 as an output and toggles it. A blue box highlights the Syntax and Parameters sections.

PROFESSIONAL EDUCATION STORE

HARDWARE SOFTWARE CLOUD DOCUMENTATION COMMUNITY

LANGUAGE

FUNCTIONS

VARIABLES

STRUCTURE

LIBRARIES

IOT CLOUD API

GLOSSARY

The Arduino Reference text is licensed under a [Creative Commons Attribution-Share Alike 3.0 License](#).

Find anything that can be improved? [Suggest corrections and new documentation via GitHub](#).

Doubts on how to use Github? Learn everything you need to know in [this tutorial](#).

for more information.

If you do not set the `pinMode()` to OUTPUT, and connect an LED to a pin, it will appear dim. Without explicitly setting `pinMode()`, `digitalWrite()` will have like a large current-limiting resistor.

Syntax

`digitalWrite(pin, value)`

Parameters

pin: the Arduino pin number.
value: HIGH OR LOW.

Returns

Nothing

Example Code

The code makes the digital pin 13 an OUTPUT and toggles it by alternating

```
void setup() {  
  pinMode(13, OUTPUT); // sets the digital pin 13 as output  
}
```

Programming 101>> Functions

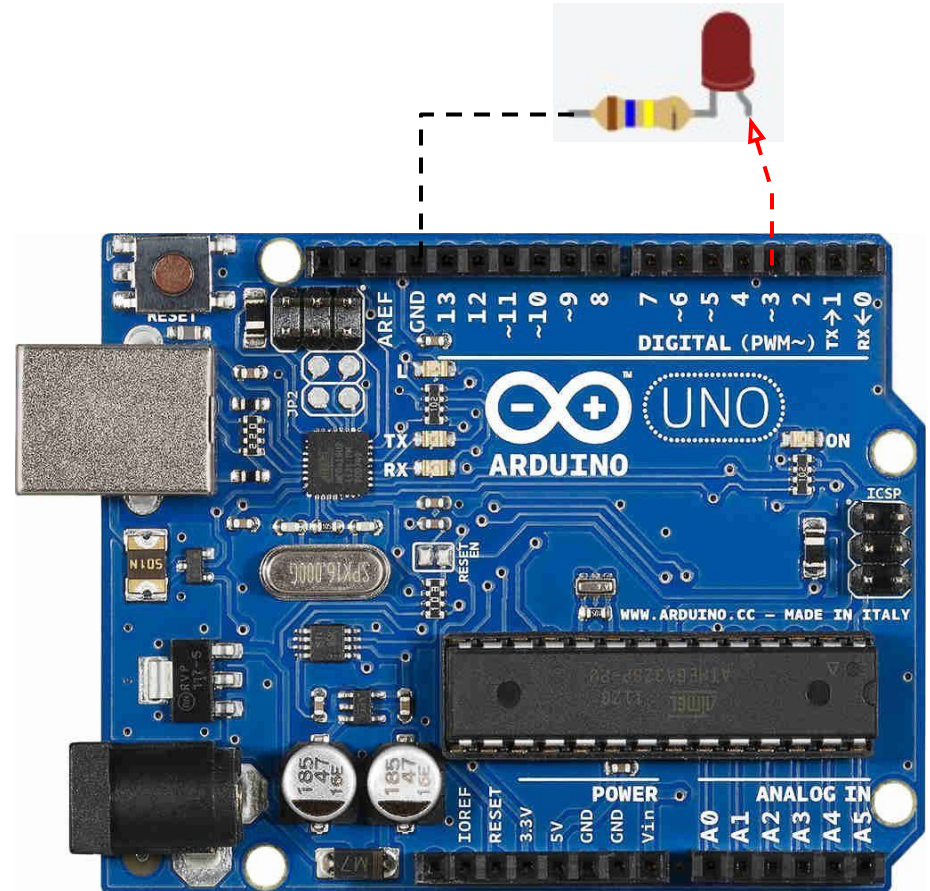
So if we write:

`digitalWrite(3, HIGH)`

The programme will set digital pin number 3 to HIGH, which means it will send 5V.

We can use the same function to then turn it off, by writing:

`digitalWrite(3, LOW)`



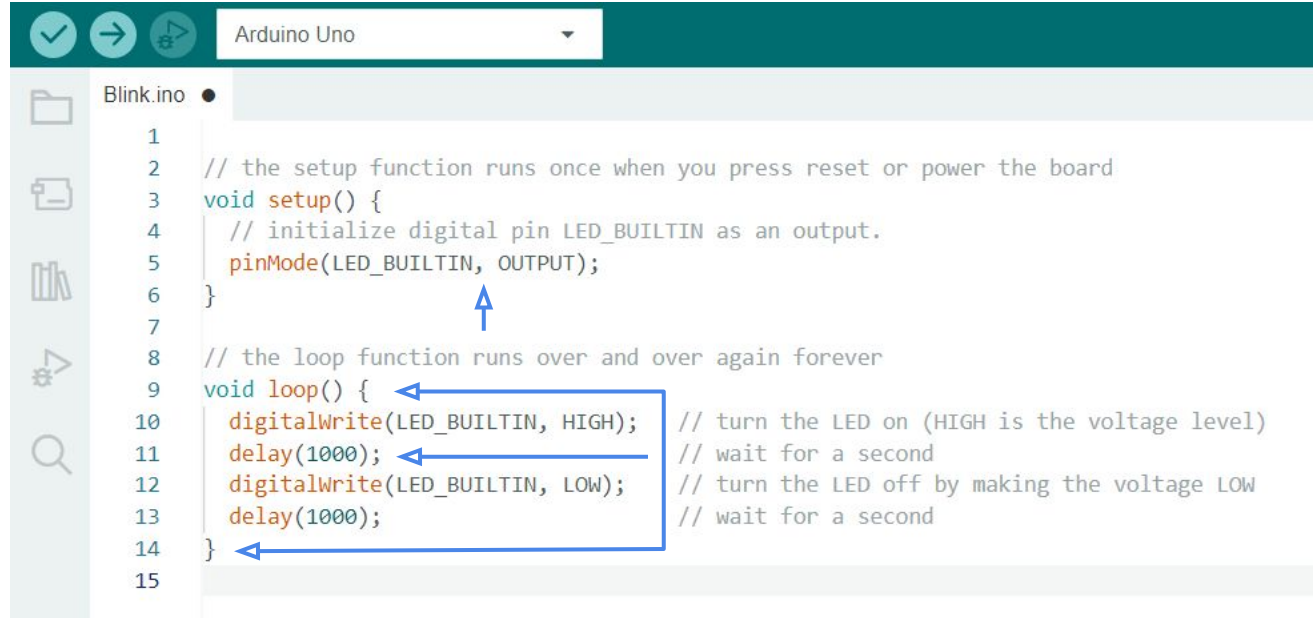
Programming 101>> Syntax

The programme itself has syntax as well, which if we don't write properly will prevent it from running.

At the end of every line of code there needs to be a semicolon ';'

The parameters of a function are put inside normal brackets '()', separated by ','

Blocks of code are organised inside curly brackets '{}'

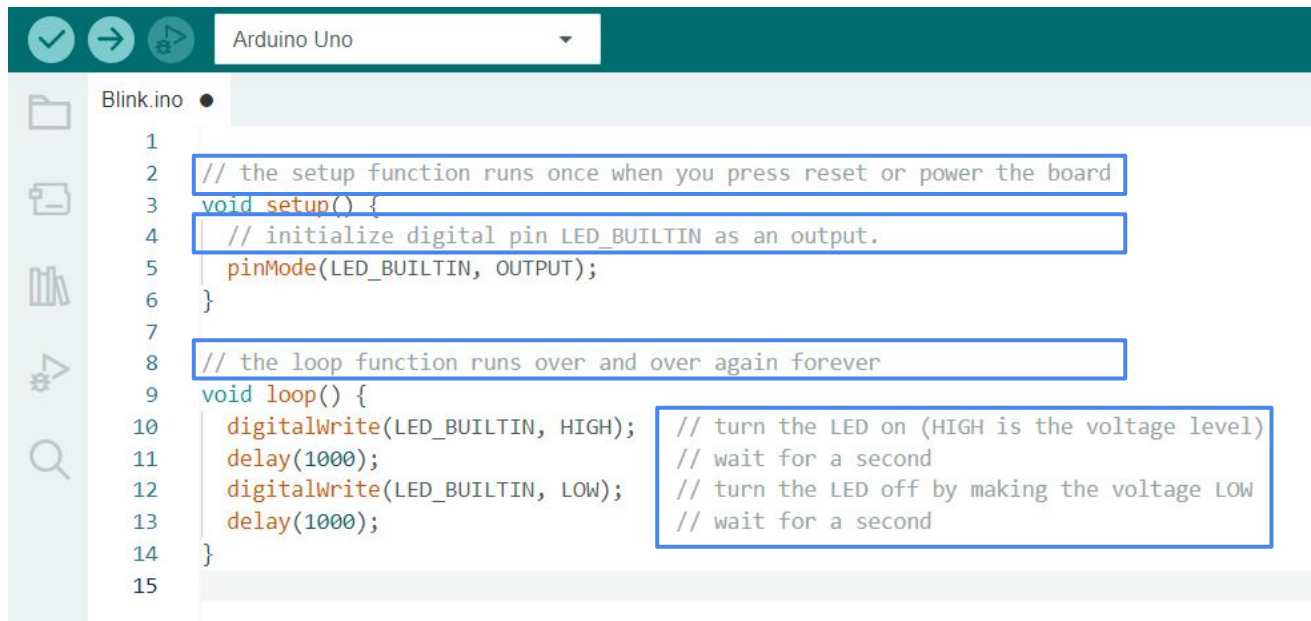


```
1
2 // the setup function runs once when you press reset or power the board
3 void setup() {
4   // initialize digital pin LED_BUILTIN as an output.
5   pinMode(LED_BUILTIN, OUTPUT);
6 }
7
8 // the loop function runs over and over again forever
9 void loop() {
10  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
11  delay(1000); // wait for a second
12  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
13  delay(1000); // wait for a second
14 }
15
```

Programming 101>> Syntax

Anything written after a `//` is ignored by the programme, and is for you or anyone else who reads the code in the future.

Codes can be complicated, so it's always a good idea to make sure they are organised and have comments along the way to explain what each part of the code is doing.



```
1
2 // the setup function runs once when you press reset or power the board
3 void setup() {
4   // initialize digital pin LED_BUILTIN as an output.
5   pinMode(LED_BUILTIN, OUTPUT);
6 }
7
8 // the loop function runs over and over again forever
9 void loop() {
10  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
11  delay(1000); // wait for a second
12  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
13  delay(1000); // wait for a second
14 }
15
```

Programming 101>> Variables and Constants

Variables and Constants

are used to store data inside of them.

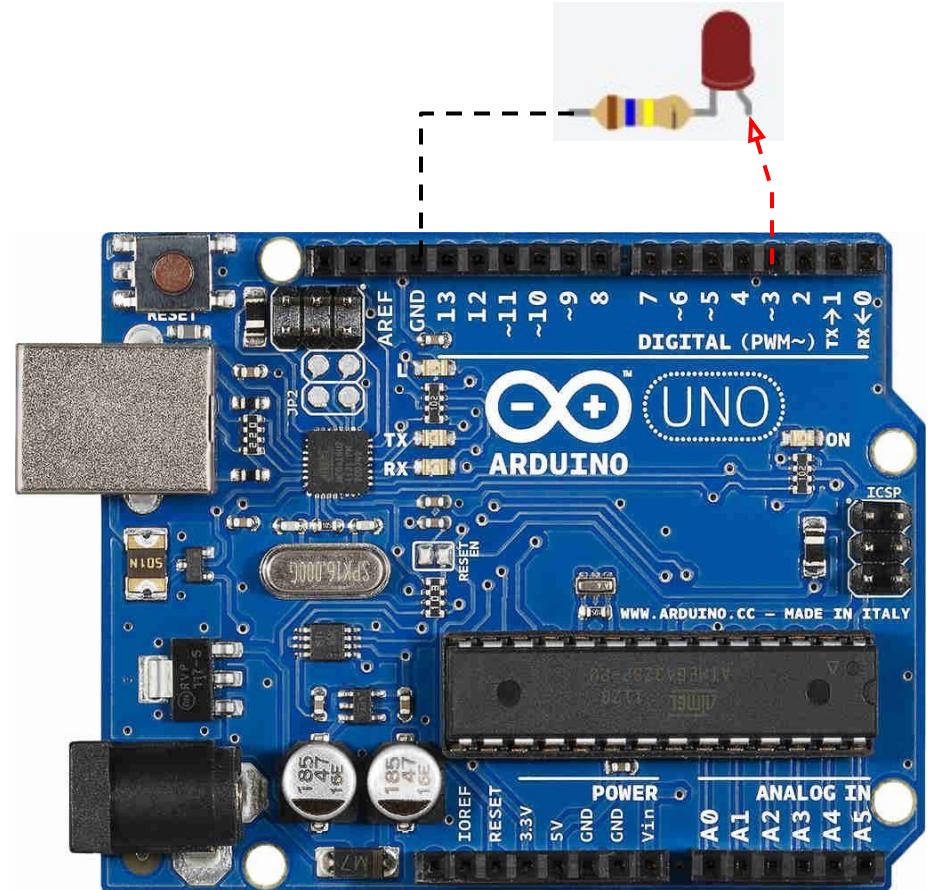
Variables will be used to store data that changes, and we can name our own to make the code more readable.

Constants will have the same value for the entire programme. Some of these constants are already known to the programme.
For example: HIGH and LOW.

Programming 101>> Variables and Constants

For example, I can declare a constant `Red_LED_Pin`, which I'll use throughout the programme instead of writing the number 3. This makes the code more readable, especially when there are several pins in use for different loads.

This also allows flexibility, because if I need to connect the LED to a different pin, I only have to change the pin number once where I declared the constant's value.



Programming 101>> Variables and Constants

There are two ways to declare a constant, each with different syntax:

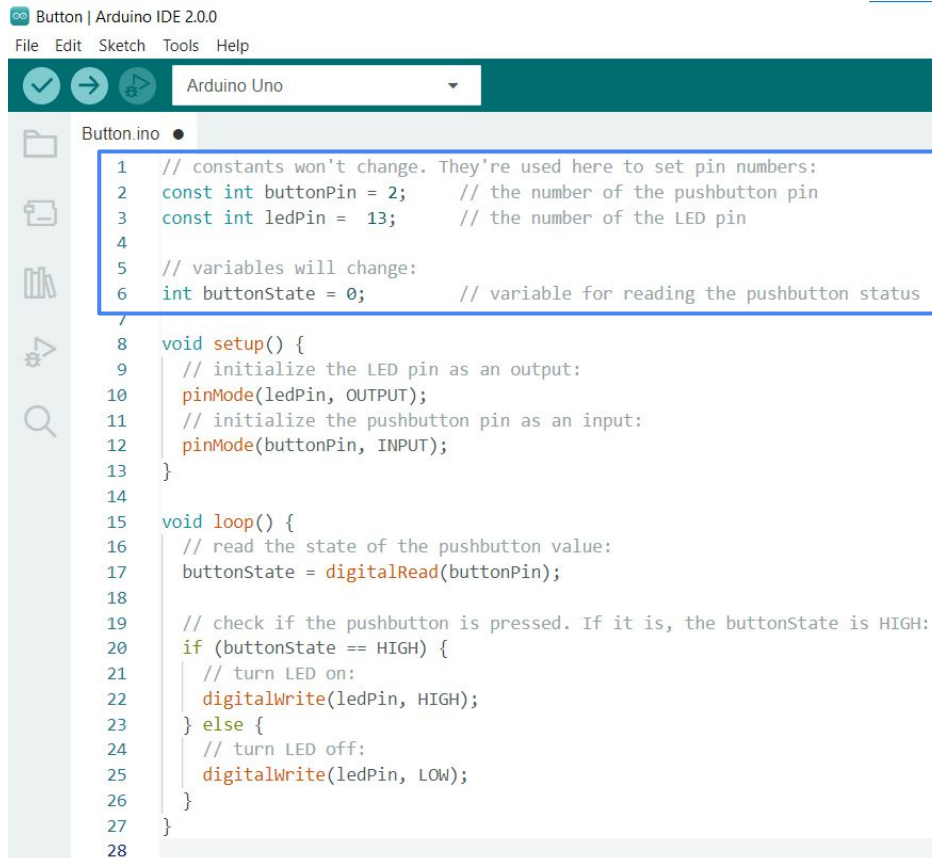
`#define Red_LED_Pin 3`

or

`const int Red_Led_Pin = 4;`

The second option is preferred, so we'll use that one.

The declaration goes in the first part of the code.



```
Button | Arduino IDE 2.0.0
File Edit Sketch Tools Help
Arduino Uno
Button.ino
1 // constants won't change. They're used here to set pin numbers:
2 const int buttonPin = 2;    // the number of the pushbutton pin
3 const int ledPin = 13;     // the number of the LED pin
4
5 // variables will change:
6 int buttonState = 0;       // variable for reading the pushbutton status
7
8 void setup() {
9   // initialize the LED pin as an output:
10  pinMode(ledPin, OUTPUT);
11  // initialize the pushbutton pin as an input:
12  pinMode(buttonPin, INPUT);
13 }
14
15 void loop() {
16   // read the state of the pushbutton value:
17   buttonState = digitalRead(buttonPin);
18
19   // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
20   if (buttonState == HIGH) {
21     // turn LED on:
22     digitalWrite(ledPin, HIGH);
23   } else {
24     // turn LED off:
25     digitalWrite(ledPin, LOW);
26   }
27 }
28
```

Programming 101>> Data types

Each variable or constant has a type of information they are allowed to store.

In this case we're declaring a constant to hold a pin number. The number of a pin will always be a 'whole number', which is called integer, `int` for short:

```
const int Red_Led_Pin = 4;
```

Programming 101>> Data types

A constant could also hold a number with a decimal point, a `float`:

```
const float distance = 1.8;
```

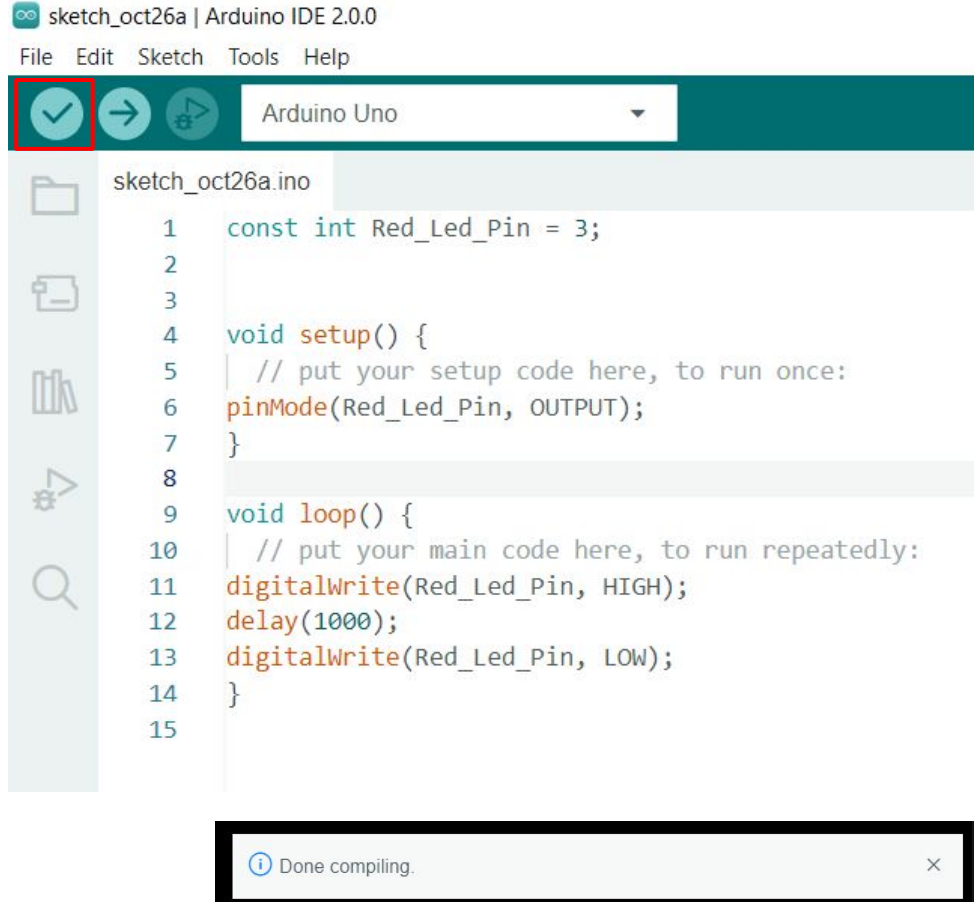
Another data type is a `string`, which means text.
For example, `HIGH` and `LOW` which we use to set pin modes, are string constants the are saved for the programme to use.

Programming 101>> Compiling and uploading

Once we've written a programme we need to make sure that it's written correctly, that we spelled things properly etc.

The verify button will run through the code and make sure it makes sense.

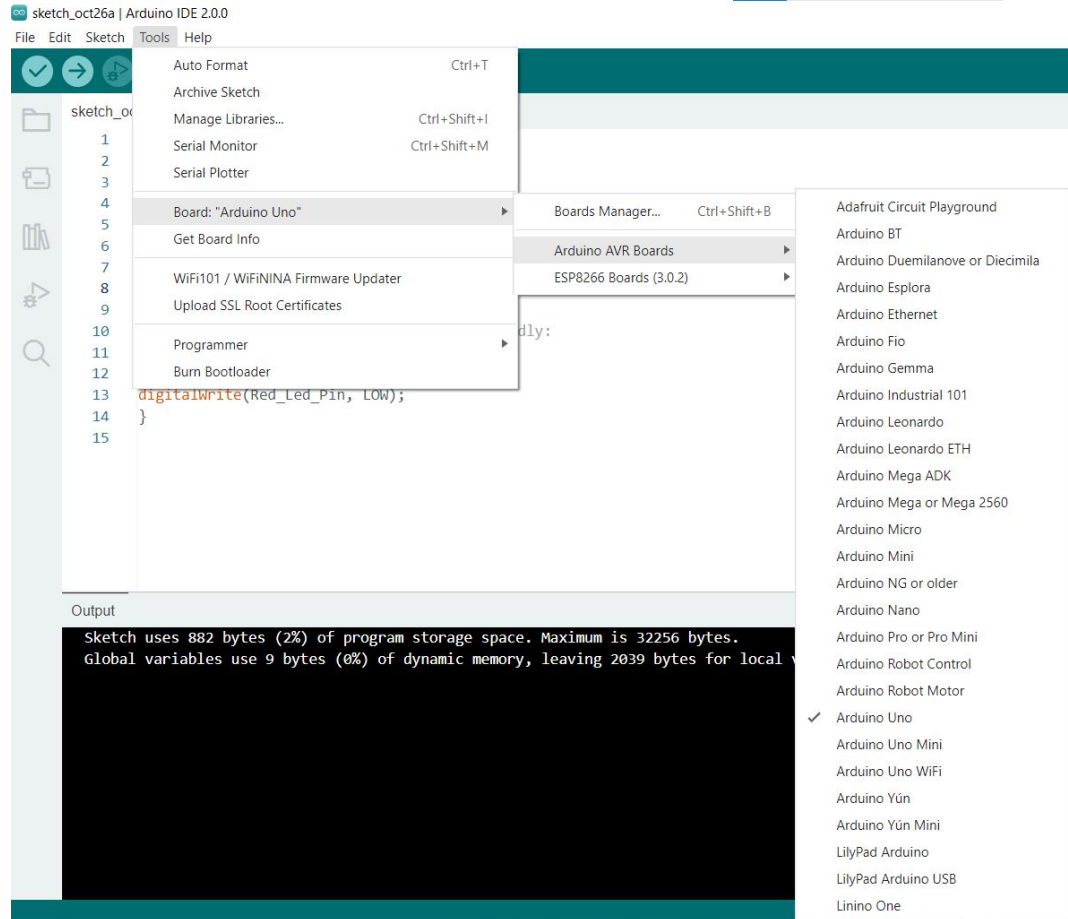
If everything is okay it will say: Done compiling.



Programming 101>> Compiling and uploading

We then need to upload the programme onto the Arduino board.

For that we need to first choose which board we are uploading to:
Tools> Board> Arduino AVR Boards> Arduino Uno



Programming 101>> Compiling and uploading

Then tell the IDE which USB port the Arduino board is connected to.

