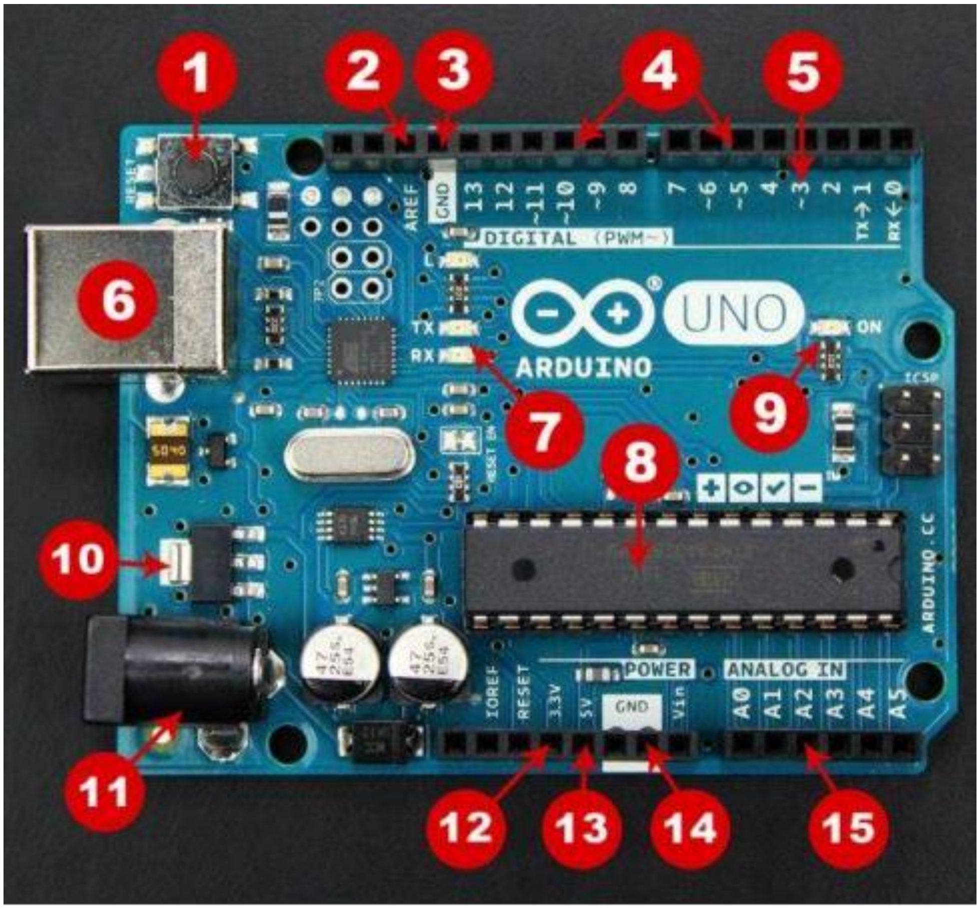
DAY 2

- Arduino hardware introduction.
- Introduction to Arduino IDE.
- First Arduino Blink program.
- Some built-in commands.

GETTING STARTED

Arduino Uno One of the most popular Arduino boards out there is the Arduino Uno. While it was not actually the first board to be released, it remains to be the most actively used and most widely documented on the market. Because of its extreme popularity, the Arduino Uno has a ton of project tutorials and forums around the web that can help you get started or out of a jam. We're big fans of the Uno because of its great features and ease of use.



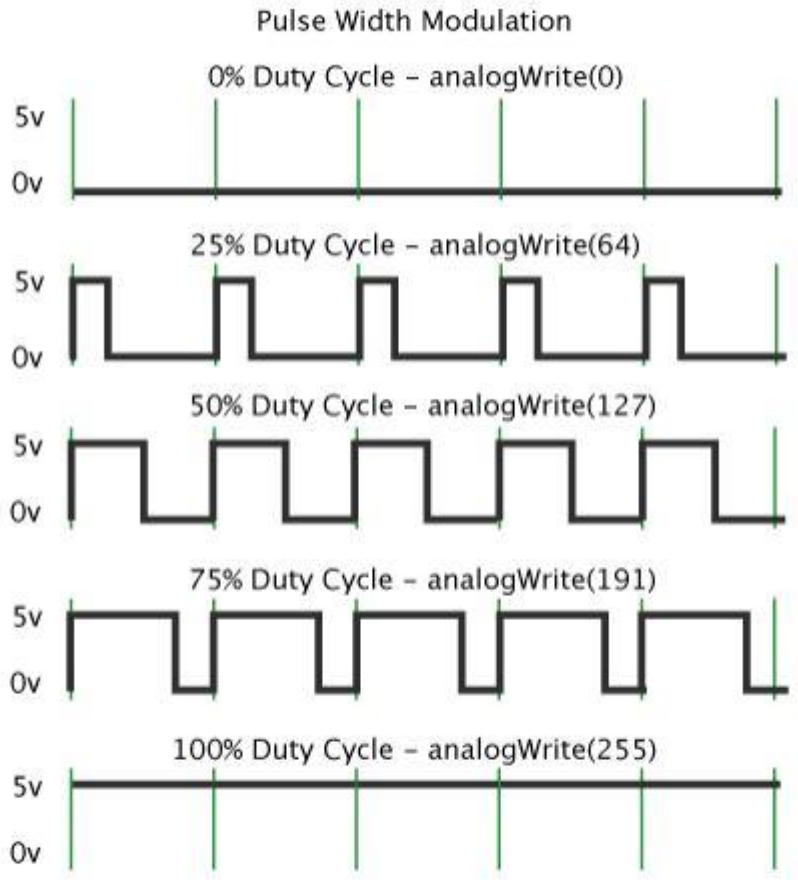
Board Breakdown Here are the components that make up an Arduino board and what each of their functions are.

1. Reset Button – This will restart any code that is loaded to the Arduino board
2. AREF – Stands for "Analog Reference" and is used to set an external reference voltage
3. Ground Pin – There are a few ground pins on the Arduino and they all work the same
4. Digital Input/Output – Pins 0-13 can be used for digital input or output
5. PWM – The pins marked with the (~) symbol can simulate analog output
6. USB Connection – Used for powering up your Arduino and uploading sketches
7. TX/RX – Transmit and receive data indication LEDs
8. ATmega Microcontroller – This is the brains and is where the programs are stored
9. Power LED Indicator – This LED lights up anytime the board is plugged in a power source

10.Voltage Regulator – This controls the amount of voltage going into the Arduino board
11.DC Power Barrel Jack – This is used for powering your Arduino with a power supply
12.3.3V Pin – This pin supplies 3.3 volts of power to your projects
13.5V Pin – This pin supplies 5 volts of power to your projects
14.Ground Pins – There are a few ground pins on the Arduino and they all work the same
15.Analog Pins – These pins can read the signal from an analog sensor and convert it to digital

What is PWM?

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends ON versus the time that the signal spends OFF. The duration of "on time" is called the pulse width. To get varying analog values, we change, or modulate, that pulse width.


Pulse Width Modulation
0% Duty Cycle - analogWrite(0)
25% Duty Cycle - analogWrite(64)
50% Duty Cycle - analogWrite(127)
75% Duty Cycle - analogWrite(191)
100% Duty Cycle - analogWrite(255)
5v
0v

To use PWM value on a pin: analogWrite(pin, value)
Arduino has 8 bit PWM module. 2e8=256 so we can give a number in between (0-255)

If PWM =0, Voltage=0v If PWM=255, Voltage= 5v If PWM=128, Voltage=2.5v

CODE:

```
int ledPin = 9; // LED connected to digital pin 9
int analogPin = 3; // potentiometer connected to analog pin 3
int val = 0; // variable to store the read value

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

void loop() {
 val = analogRead(analogPin); // read the input pin
analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite values from 0
to 255
 }
```

Note: ADC resolution is of 10 bit so it will return a number in between 0 to 1023, but Resolution of PWM is 8 bit so we can write only a number in between 0 to 255.

HARDWARE REQUIREMENTS

Arduino board Jumper wires USB cable

SOFTWARE REQUIREMENTS

- Arduino IDE (can be downloaded from https://www.arduino.cc/en/Main/Software)

ARDUINO BOARDS

There are various Arduino Boards available at market. All are same in functionality but differ from each other in number of I/O pins, communication pins, PWM pins, ADCs etc. Following are some of Arduino boards:

Arduino UNO:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7V-12V
Digital I/O Pins	14
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20mA
Clock Speed	16MHz

Arduino Mega:

Microcontroller	ATmega1280
Operating Voltage	5V
Input Voltage (recommended)	7V-12V
Digital I/O Pins	54
PWM Digital I/O Pins	15
Analog Input Pins	16
DC Current per I/O Pin	40mA
Clock Speed	16MHz

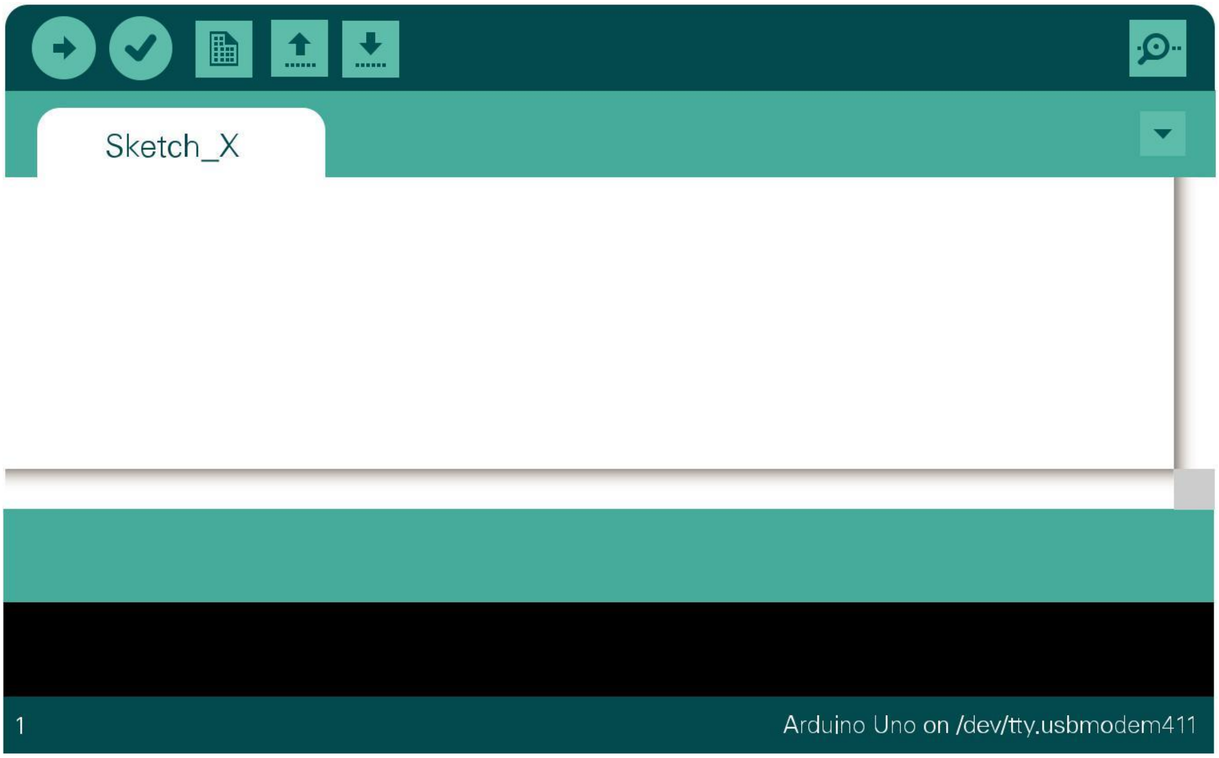
Arduino Mini:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7V-12V
Digital I/O Pins	14

PWM Digital I/O Pins	6
Analog Input Pins	8
DC Current per I/O Pin	40mA
Clock Speed	16MHz

INTRODUCTION TO IDE:

First, we're going to have a look at the Arduino software and how we're going to create our own programs.



The Arduino IDE is a very simple, easy to use programming environment, with most of the key features visible on the toolbar.

➜ Checks if your code is correct

✓ Uploads your program to the connected Arduino

Creates a new program

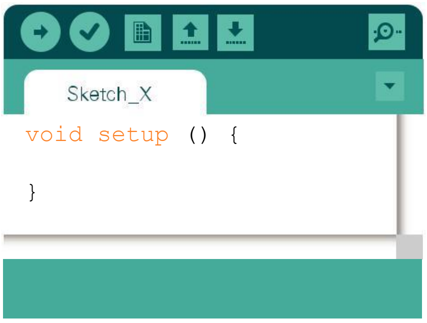
Open a program

Saves your program on your computer

To upload a program to the Arduino, simply connect the arduino using a USB port. This should prompt you that a new network interface has been detected. Once this has appeared we can click the upload button and our program will be uploaded from our computer to our Arduino. As we had not programmed anything yet, our program will not show any function on our Arduino. In Arduino, we call a program a Sketch.

Basics of Arduino IDE

Let’s have a brief look at how to write code in Arduino IDE.

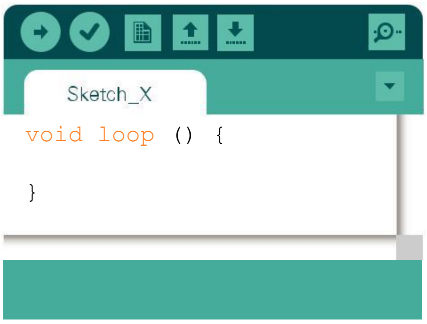


```
void setup () {

}
```

Void Setup

This is the first thing called in Arduino and is only called once. This is where we set our initial settings.

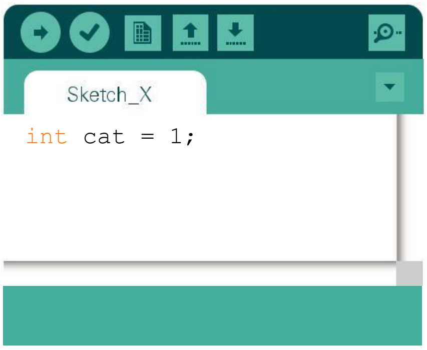


```
void loop () {

}
```

Void Loop

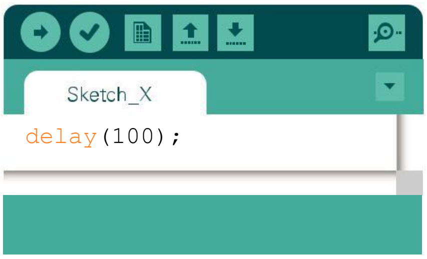
This is where we put the program’s main loop. The void loop is repeated forever.



```
int cat = 1;
```

Variable

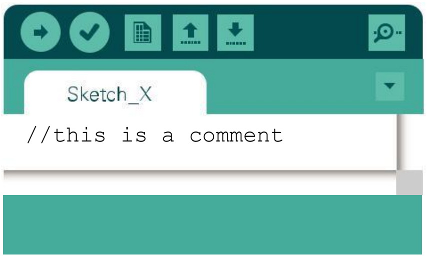
We use variables to store values so we can keep them in one place to access them. This is a integer variable, which means it is storing a whole number. The int part refers to the type of variable, followed by the chosen name used to store the variable. A variable can be named practically anything. Once we have done this and have initialised the variable, we won’t need to refer to it using int, just its name. This makes reading and writing the code more intuitive.



```
delay(100);
```

Function

This is a function. These are used to do different procedures in our program loop. You can always tell a function by two things; the colour of the function text changes to orange when typed correctly, and the use of brackets. Functions almost always need user input to change the outcome of the function, these are placed within the brackets.



```
//this is a comment
```

Comment

Comments are how we explain code without using Arduino language. By adding two forward slashes, the line is not read by the Arduino and is just for humans to understand. You don’t have to write the comments, but they will help you (and others) understand the code a lot quicker!

OUR FIRST BLINK PROGRAM

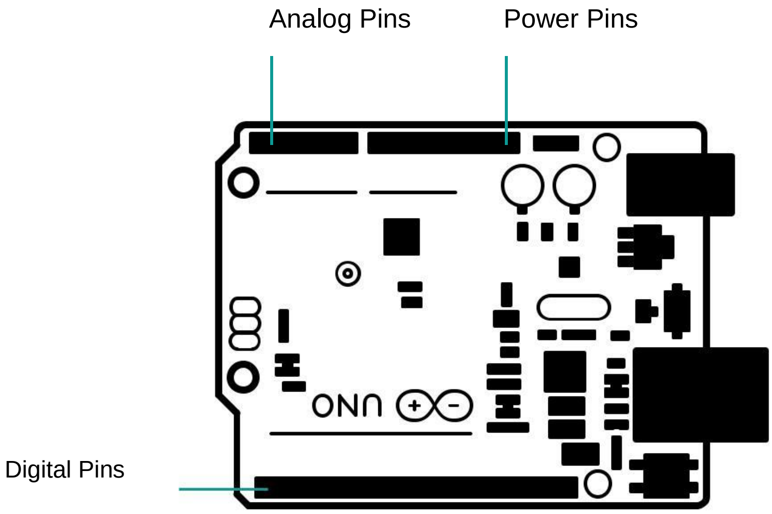
THINGS WE NEED:

The Arduino

Arduinos use different symbols and numbers for each pin to signal what each pin does.

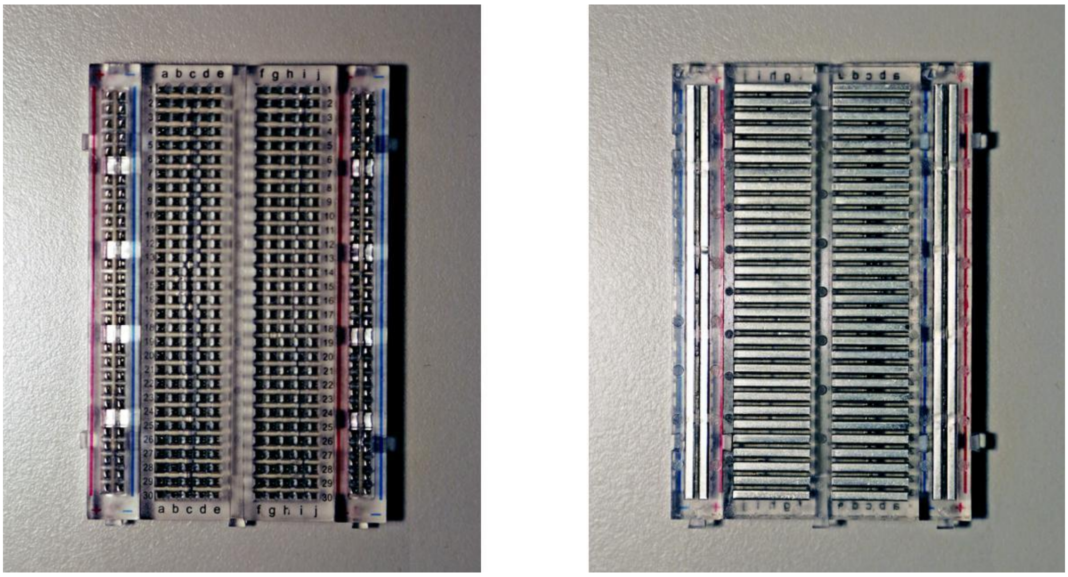
Digital Pins - These are all labelled 0 to 13. These pins are all output and input pins, meaning they are capable of outputting a voltage of 5 volts or sensing an input voltage of 5 volts. As these are digital pins, they have two states, HIGH or LOW and nothing in between.

Analog pins - These are input pins that can detect voltage between 0 and 5 volts. As these are analog pins, they can have an output value of between 0 and 1023.



The Breadboard

This is one of the most important pieces of equipment we will be using. We use this to create circuits without the need to use a soldering iron.



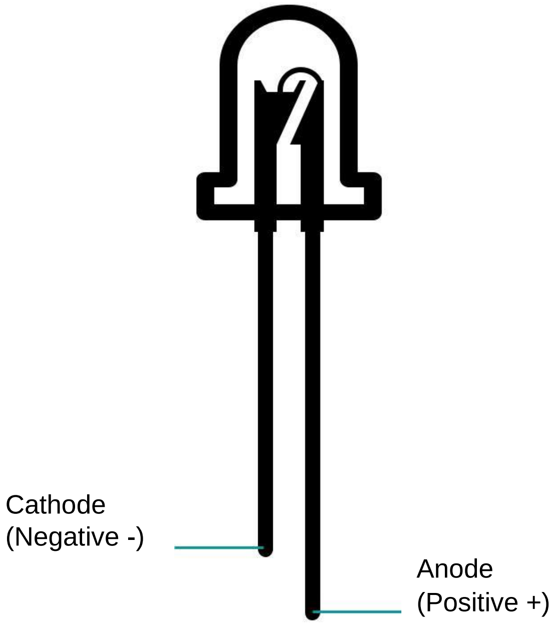
Each row (numbered 1 - 30) is backed horizontally with a plate of metal underneath. This means components placed on the same row are always connected, and we can create connections between rows using jumper wires and components, making our circuits! The picture above demonstrates the connections between each row. (Note: the edges, known as power rails, are connected by column instead of row, so that power is accessible from the whole breadboard.)

LED

Start Arduino has lots of LEDs. Their real name, Light Emitting Diode, is commonly shortened to LED. These are some of the most common pieces of equipment used in Arduino projects.

A Light Emitting Diode (LED) is a light that only lets electricity through it one way. If you look at the LED, you will notice that there is a difference in length of the legs. This is how we know which way to use the LED in the circuit.

The longer leg is to signify the anode. This is the side we connect to the positive of a power source. The shorter leg is the cathode. This side needs to be connected to the negative of a power source.



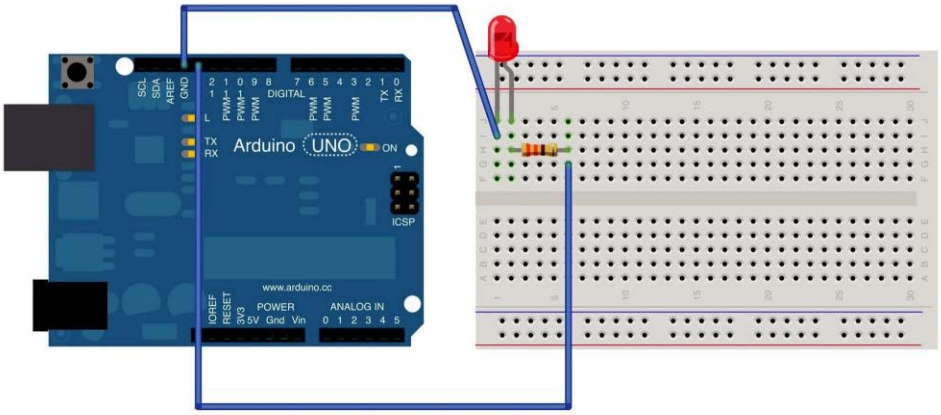
We are going to make a series circuit, using the 5v pin of our Arduino to make a circuit and turn a light on.

For this, we will need:

1x Arduino Uno
1x USB cable
1x Breadboard
1x LED
1x 220 Ohm Resistor
(Red Red Brown Gold)
2x Jumper wires

In series circuits, we have to make sure we have two things:

A voltage : 5V
A Ground : GND



Once you have created this circuit, plug the USB cable into the Arduino and connect it to the computer and you should see your LED light up with no coding needed.

Stop! Is my circuit working?

- Make sure the LED is the correct way (longest leg connected to the resistor)
- Make sure the right Arduino pins are connected (5v and GND)
- Make sure your are connecting to the same rows of your breadboard
- Is your resistor the right value

Why use resistors?

Once you’re circuit is working you may wonder why we are using a resistor. This is because we need to make sure that the voltage from the Arduino power out is correct for the LED, therefore we resist it. We can work out the necessary resistor using formula:

Voltage = Current x Resistance

As we know the voltage (5v) and the current draw of the LED is ~20mA, we can work out the resistance.

Resistance = Voltage / Current

Resistance(Ohm) = 5(V) / 0.020(A)

250(OHM) = 5(V) / 0.020(A)

As the LED itself has it’s own slight resistance, 220 Ohm Resistor is perfect.

SINGLE LED BLINKING CIRCUIT:

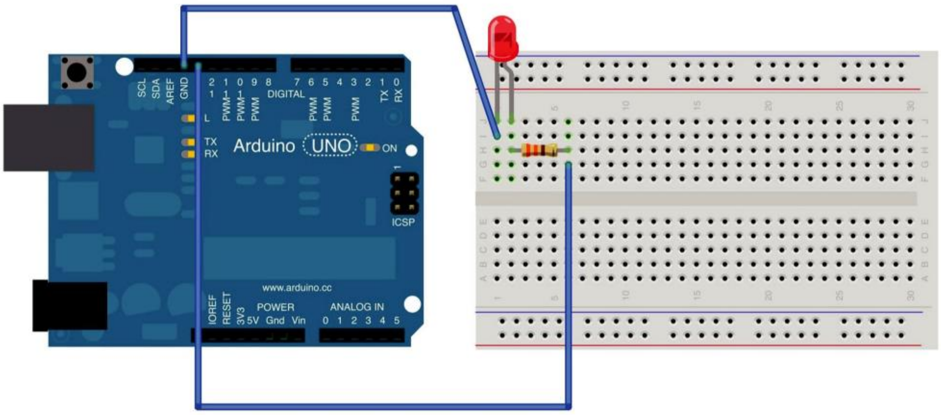
Next you are going to program the Arduino to control the LED.

For this, we will need:

1x Arduino Uno
1x USB cable
1x Breadboard
1x LED
1x 220 Ohm Resistor
(Red Red Brown Gold)
2x Jumper wires

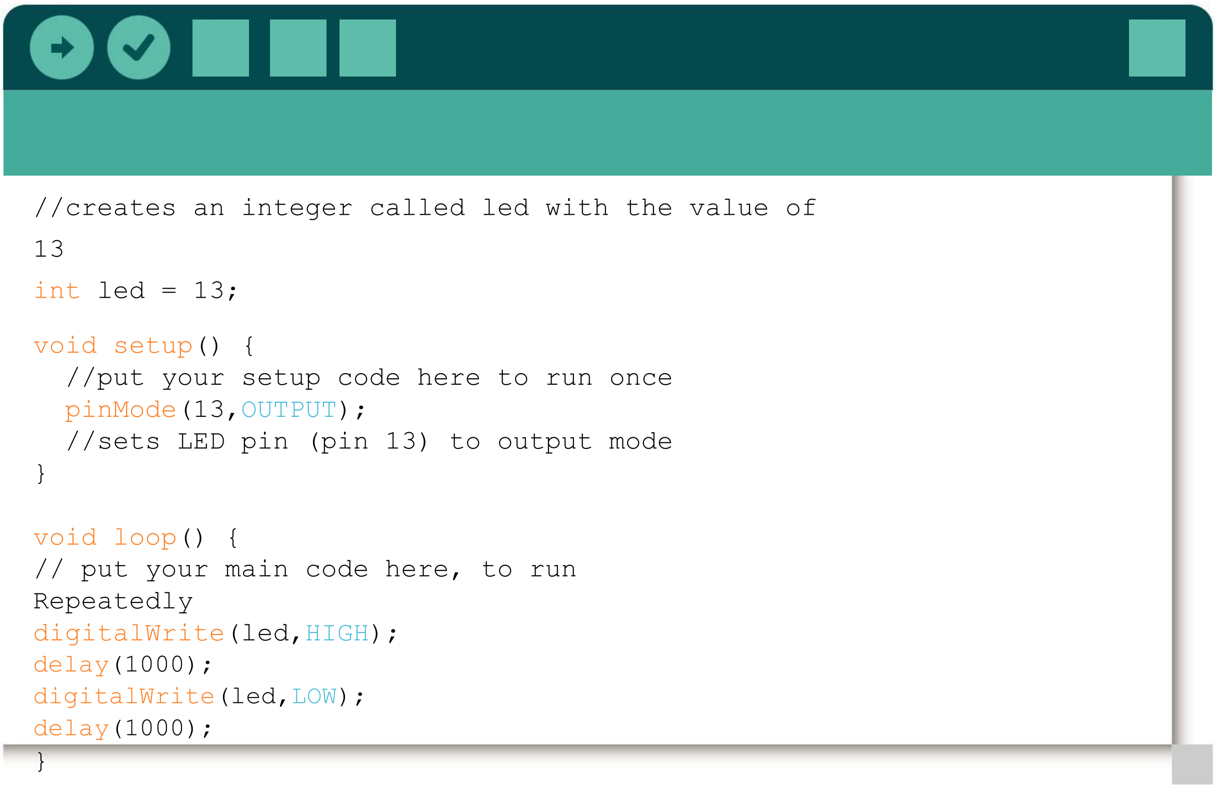
In series circuits, we have to make sure we have two things:

A voltage : 5V
A Ground : GND



The difference between this circuit and your last one is the we have moved the jumper wire from 5v to digital pin 13. This connects the longest leg of the LED to Pin 13. This will allow us to control the output to pin 13 to make the led flash .

Make a new sketch in the Arduino IDE 🟩 and name this ‘blink’.



```
//creates an integer called led with the value of
13
int led = 13;

void setup() {
  //put your setup code here to run once
  pinMode(13,OUTPUT);
  //sets LED pin (pin 13) to output mode
}

void loop() {
// put your main code here, to run
Repeatedly
digitalWrite(led,HIGH);
delay(1000);
digitalWrite(led,LOW);
delay(1000);
}
```

Once you have copied the code, press ➜ (compile) and if no errors appear, press ✓
(upload) and watch the results!

My code won’t compile!

- Check your spelling.
Is everything spelt correctly?

- Check your punctuation.
Are all your lines ending in a semi-colon?
Do you have the correct capital letters?

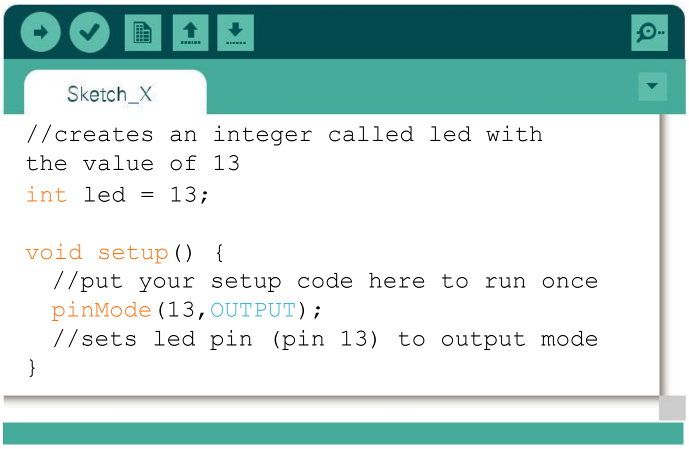
- Did you close the curly brackets? Make sure your void setup and void loop both have everything surrounded by the open and closed curly brackets.

Is your computer asking you to choose the right Serial port?

This can be solved by going to:
Tools > Port > dev/tty.usb.... for Mac
Tools > Port > COM... for Windows.

How does it work?

Let’s breakdown the code you have just written to find out how it works.

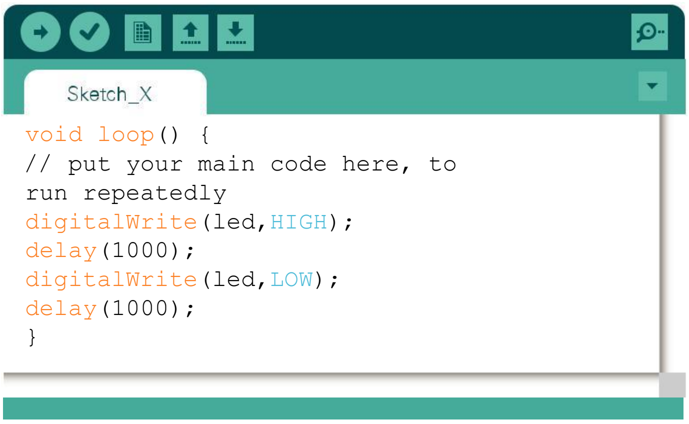


```
//creates an integer called led with
the value of 13
int led = 13;

void setup() {
  //put your setup code here to run once
  pinMode(13,OUTPUT);
  //sets led pin (pin 13) to output mode
}
```

Void Setup

This is the first thing called in Arduino and is only called once. This is where we set our initial settings.

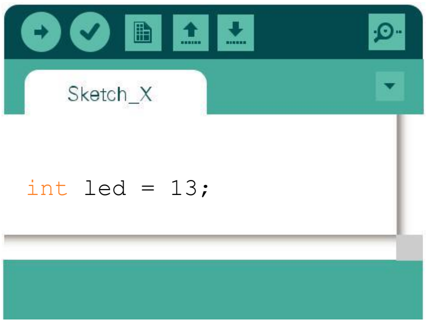


```
void loop() {
// put your main code here, to
run repeatedly
digitalWrite(led,HIGH);
delay(1000);
digitalWrite(led,LOW);
delay(1000);
}
```

Void Loop

This is where we put the program’s main loop. The void loop is repeated forever, or until the Arduino has no power.

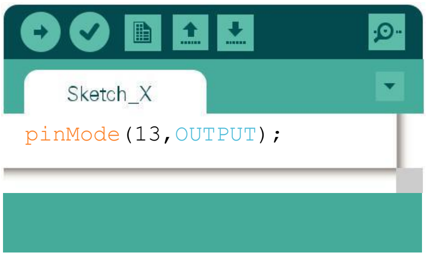
In void setup and void loop we have a 4 different parts to look at.



```
int led = 13;
```

Integer / global variable

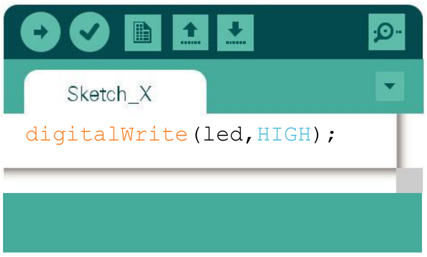
This is used to stored values we want to keep using. It is called outside both void loop and void setup as this is called a global variable, meaning it can be used by both void loop and void setup. In this instance we have created an integer called led and assigned it the number 13. By using names in global variables, from now on in the code you can refer to the component by their given name (i.e. led) not port numbers. This makes writing and reading code much more intuitive.



```
pinMode(13,OUTPUT);
```

Function

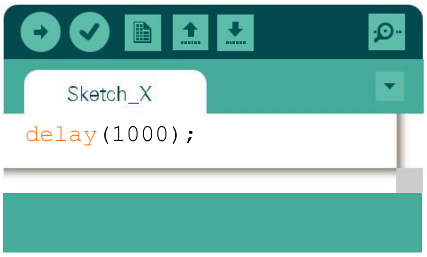
This is an Arduino function we use to set digital pins to either accept an input voltage or output one. We have set led pin (13) to be an output pin.



```
digitalWrite(led,HIGH);
```

Function

This is another Arduino function used to turn digital pins high or low. High outputting a 5 volt signal and low outputting nothing, essentially on or off. To set a pin we simply state which pin we would like to change (led pin) and what state we would like it to be (HIGH or LOW).



```
delay(1000);
```

Delay

The final part of this program is the delay function. Delay is used to pause the program for a length of time. In this instance, we are delaying our program for 1000 milliseconds (1 second). This means the program will not move to the next step until it has waited for 1000 milliseconds.

Once we are happy our code works and is uploaded to our Arduino, we can save this program

to our computer using the save button Naming it ‘Blink’.

- What would you need to change to make your LED blink twice as fast?

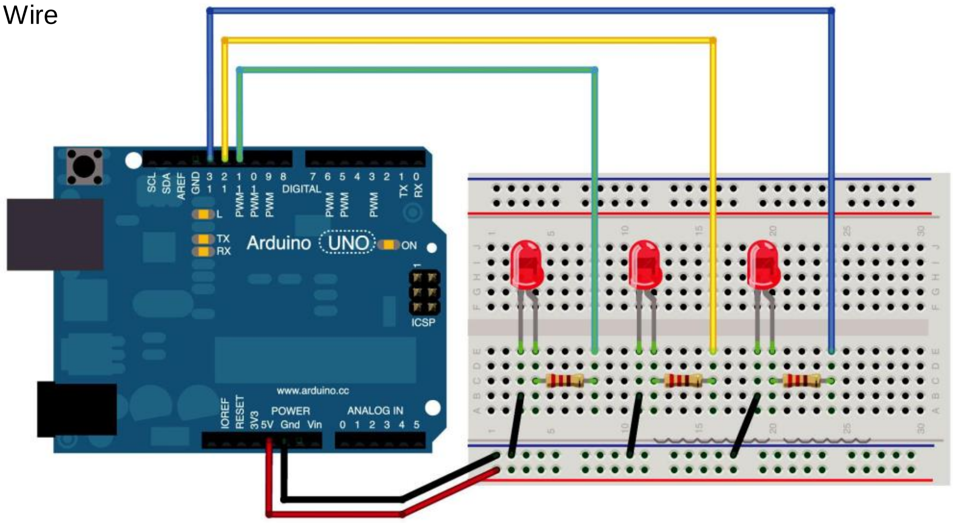
- Now we have managed to program our LED to
- circuit on and off.
- Let’s look at different ways of turning it on and off.

MORE THAN ONE LED

Now you’ve managed this with one LED, let’s try and do it with three lEDs.

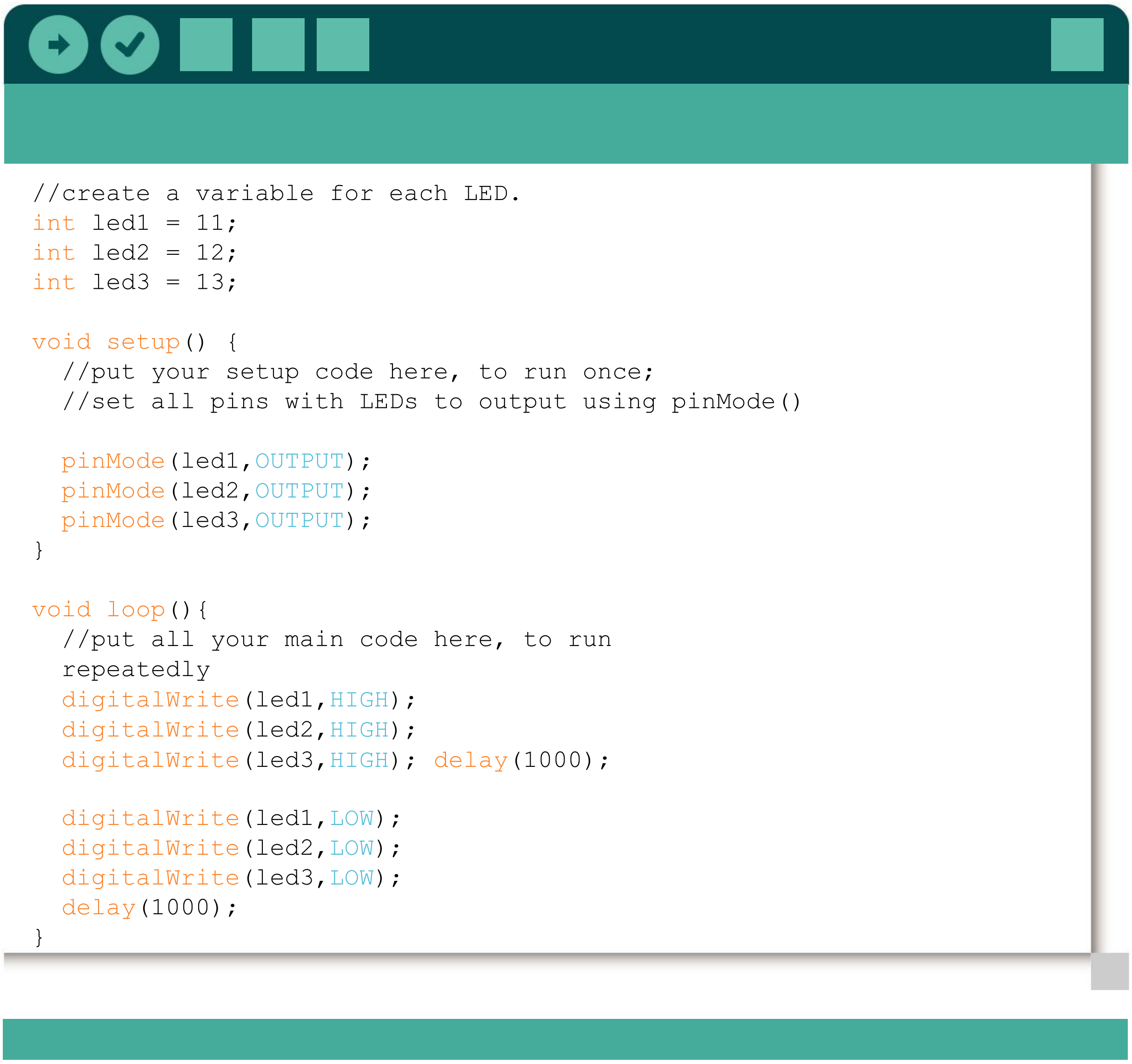
For this, we will need:

1x Arduino Uno
1x USB cable
1x Breadboard
3x LED
3x 220 Ohm Resistor
(Red Red Brown Gold)
8x Jumper Wire



As you can see, we have created the same circuit as the blink project, but multiplied 3 times. We have used the power rail of the breadboard, so we can share one ground point on the Arduino.

Make a new sketch in the Arduino IDE ■ and name this ‘Multi_Blink’.



```
//create a variable for each LED.
int led1 = 11;
int led2 = 12;
int led3 = 13;

void setup() {
  //put your setup code here, to run once;
  //set all pins with LEDs to output using pinMode()

  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
}

void loop(){
  //put all your main code here, to run
  repeatedly
  digitalWrite(led1,HIGH);
  digitalWrite(led2,HIGH);
  digitalWrite(led3,HIGH); delay(1000);

  digitalWrite(led1,LOW);
  digitalWrite(led2,LOW);
  digitalWrite(led3,LOW);
  delay(1000);
}
```

Once you have copied the code, press ✓ (compile) and if no errors appear, press ➜ (upload) and watch the results!

My code won’t compile!

- LEDs the right way?

- Everything on the same row of breadboard that is supposed to connect?

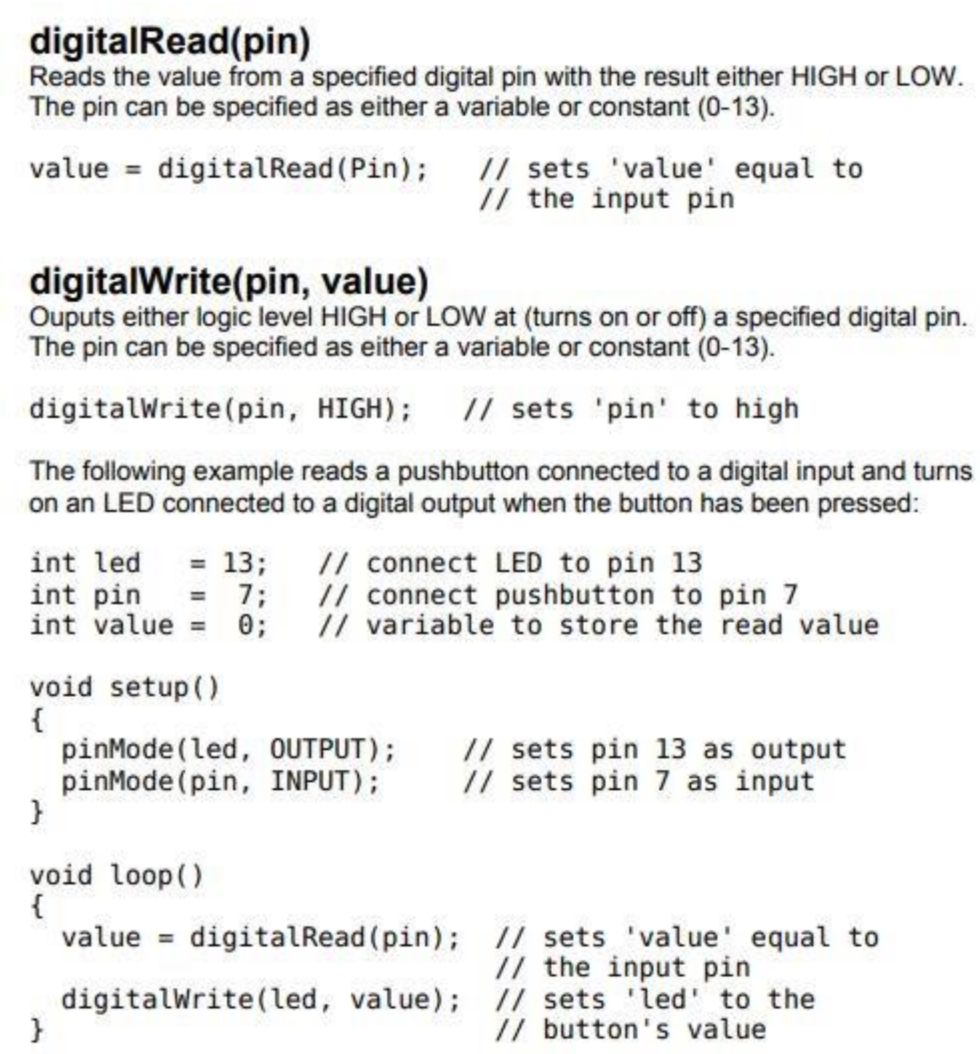
- 5v and Ground connected?

What is happening?

Our code is doing exactly the same of the original blink sketch, but with 3 LEDs.

- **Can you alternate leds turning on and off (LED1 and LED3 on when LED2 is off?**
- **Can you get them to blink one after the other?**

SOME BUILT IN COMMANDS:



digitalRead(pin)

Reads the value from a specified digital pin with the result either HIGH or LOW. The pin can be specified as either a variable or constant (0-13).

```
value = digitalRead(Pin);    // sets 'value' equal to
                             // the input pin
```

digitalWrite(pin, value)

Ouputs either logic level HIGH or LOW at (turns on or off) a specified digital pin. The pin can be specified as either a variable or constant (0-13).

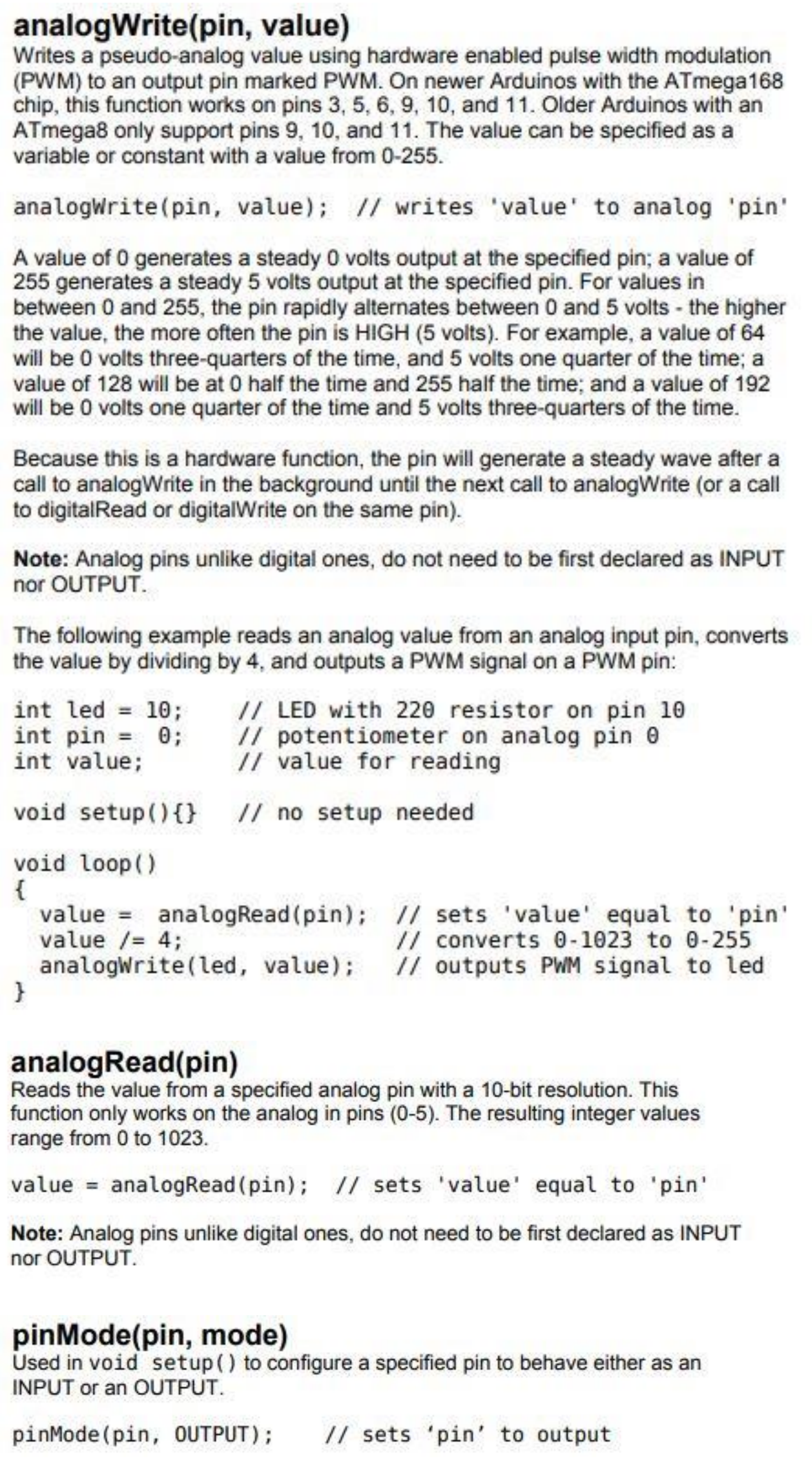
```
digitalWrite(pin, HIGH);   // sets 'pin' to high
```

The following example reads a pushbutton connected to a digital input and turns on an LED connected to a digital output when the button has been pressed:

```
int led   = 13;   // connect LED to pin 13
int pin   =  7;   // connect pushbutton to pin 7
int value =  0;   // variable to store the read value

void setup()
{
  pinMode(led, OUTPUT);     // sets pin 13 as output
  pinMode(pin, INPUT);      // sets pin 7 as input
}

void loop()
{
  value = digitalRead(pin);  // sets 'value' equal to
                             // the input pin
  digitalWrite(led, value);  // sets 'led' to the
}                            // button's value
```



analogWrite(pin, value)

Writes a pseudo-analog value using hardware enabled pulse width modulation (PWM) to an output pin marked PWM. On newer Arduinos with the ATmega168 chip, this function works on pins 3, 5, 6, 9, 10, and 11. Older Arduinos with an ATmega8 only support pins 9, 10, and 11. The value can be specified as a variable or constant with a value from 0-255.

```
analogWrite(pin, value);  // writes 'value' to analog 'pin'
```

A value of 0 generates a steady 0 volts output at the specified pin; a value of 255 generates a steady 5 volts output at the specified pin. For values in between 0 and 255, the pin rapidly alternates between 0 and 5 volts - the higher the value, the more often the pin is HIGH (5 volts). For example, a value of 64 will be 0 volts three-quarters of the time, and 5 volts one quarter of the time; a value of 128 will be at 0 half the time and 255 half the time; and a value of 192 will be 0 volts one quarter of the time and 5 volts three-quarters of the time.

Because this is a hardware function, the pin will generate a steady wave after a call to analogWrite in the background until the next call to analogWrite (or a call to digitalRead or digitalWrite on the same pin).

Note: Analog pins unlike digital ones, do not need to be first declared as INPUT nor OUTPUT.

The following example reads an analog value from an analog input pin, converts the value by dividing by 4, and outputs a PWM signal on a PWM pin:

```
int led = 10;      // LED with 220 resistor on pin 10
int pin =  0;      // potentiometer on analog pin 0
int value;         // value for reading

void setup(){}     // no setup needed

void loop()
{
  value =  analogRead(pin);  // sets 'value' equal to 'pin'
  value /= 4;                // converts 0-1023 to 0-255
  analogWrite(led, value);   // outputs PWM signal to led
}
```

analogRead(pin)

Reads the value from a specified analog pin with a 10-bit resolution. This function only works on the analog in pins (0-5). The resulting integer values range from 0 to 1023.

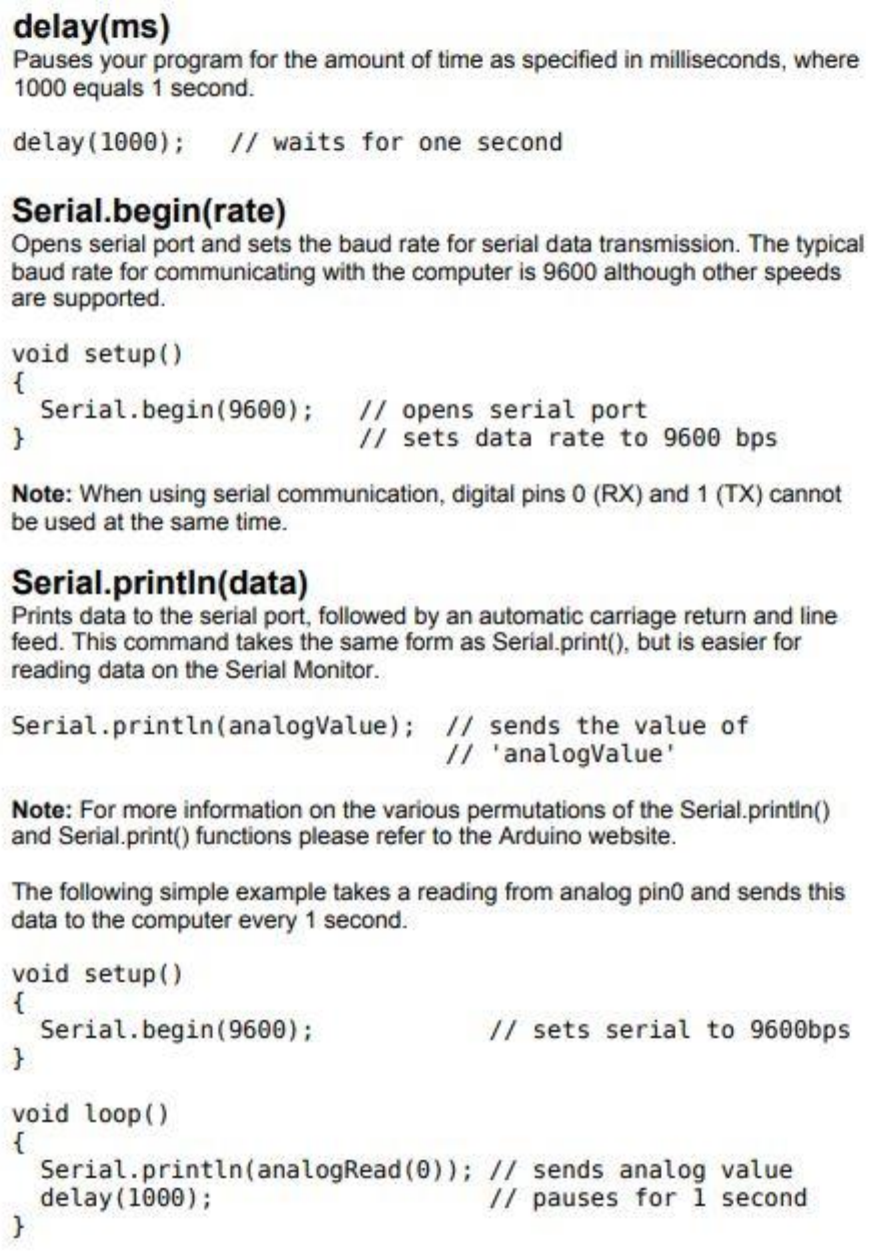
```
value = analogRead(pin);  // sets 'value' equal to 'pin'
```

Note: Analog pins unlike digital ones, do not need to be first declared as INPUT nor OUTPUT.

pinMode(pin, mode)

Used in `void setup()` to configure a specified pin to behave either as an INPUT or an OUTPUT.

```
pinMode(pin, OUTPUT);    // sets ‘pin’ to output
```



delay(ms)

Pauses your program for the amount of time as specified in milliseconds, where 1000 equals 1 second.

```
delay(1000);   // waits for one second
```

Serial.begin(rate)

Opens serial port and sets the baud rate for serial data transmission. The typical baud rate for communicating with the computer is 9600 although other speeds are supported.

```
void setup()
{
  Serial.begin(9600);   // opens serial port
}                       // sets data rate to 9600 bps
```

Note: When using serial communication, digital pins 0 (RX) and 1 (TX) cannot be used at the same time.

Serial.println(data)

Prints data to the serial port, followed by an automatic carriage return and line feed. This command takes the same form as Serial.print(), but is easier for reading data on the Serial Monitor.

```
Serial.println(analogValue);  // sends the value of
                              // 'analogValue'
```

Note: For more information on the various permutations of the Serial.println() and Serial.print() functions please refer to the Arduino website.

The following simple example takes a reading from analog pin0 and sends this data to the computer every 1 second.

```
void setup()
{
  Serial.begin(9600);            // sets serial to 9600bps
}

void loop()
{
  Serial.println(analogRead(0)); // sends analog value
  delay(1000);                   // pauses for 1 second
}
```