



# Welcome to the **CREATIVE ROBOTICS CLUB**



Want to talk?  
Need help?

**Join us on Discord**

# WHAT DO WE DO AT THE CREATIVE ROBOTICS CLUB?

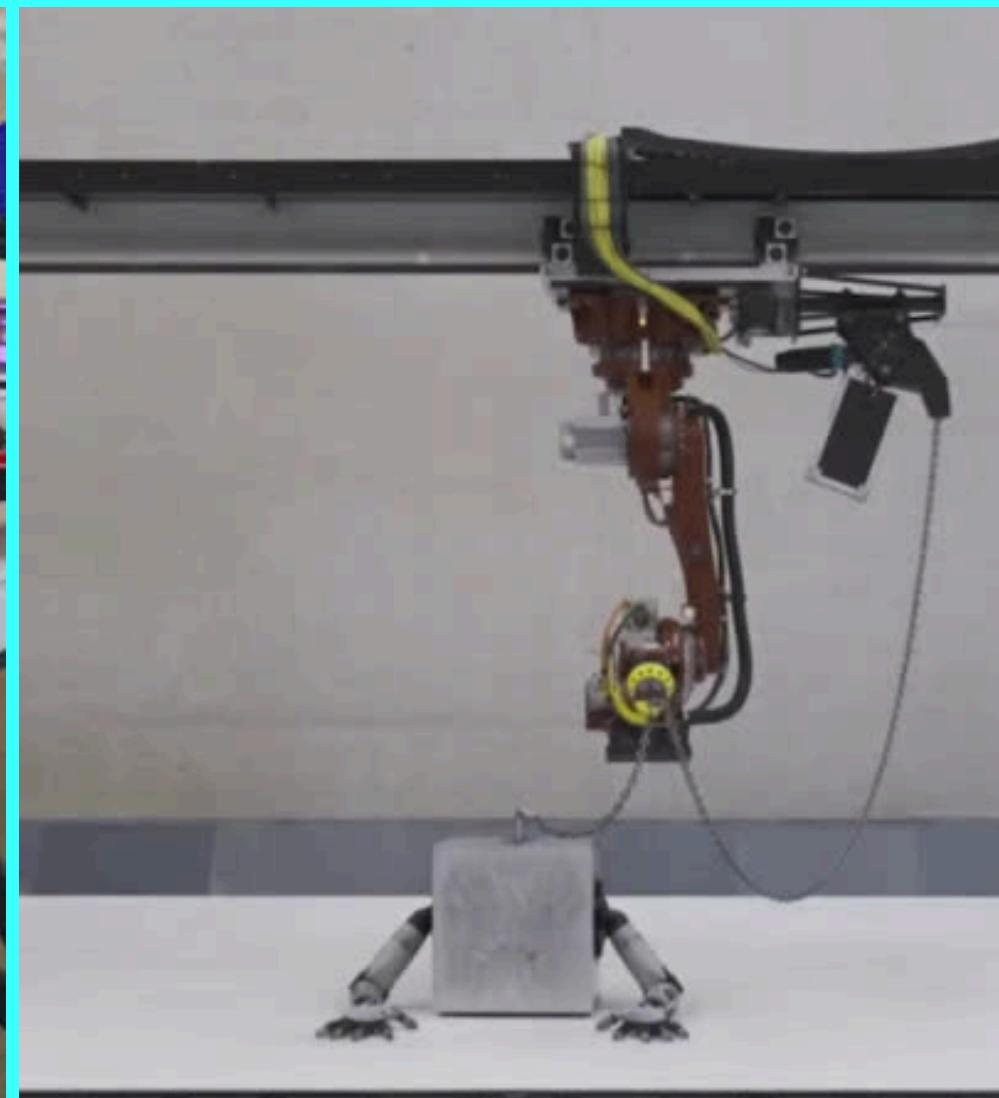
We learn how to use electricity,  
robotics and code to make things

We make art, design, or social robotics  
– we support all disciplines

We reuse and repurpose where we can

We have fun

# WHAT ARE WE DOING TODAY AT THE CREATIVE ROBOTICS CLUB?



## SERVO MOTORS

BUT FIRST LETS TALK ABOUT...

LAST WEEK

# Last week:



DC gear motor



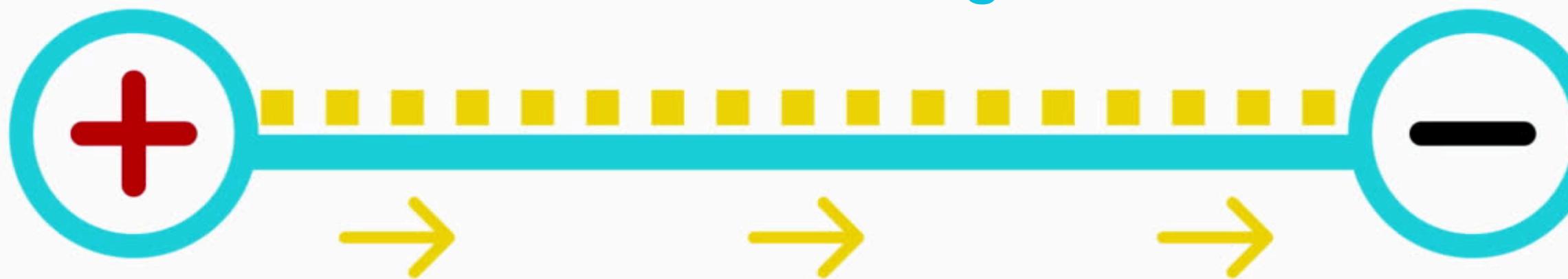
AA Battery [1.5v]

## AA Battery holder



# HOW DOES ELECTRICITY WORK?

Current moves from  
Positive to negative



Positive: 5v, 3.3v, +, Vin, etc

Negative: GND, Ground, -, ↴

Positive



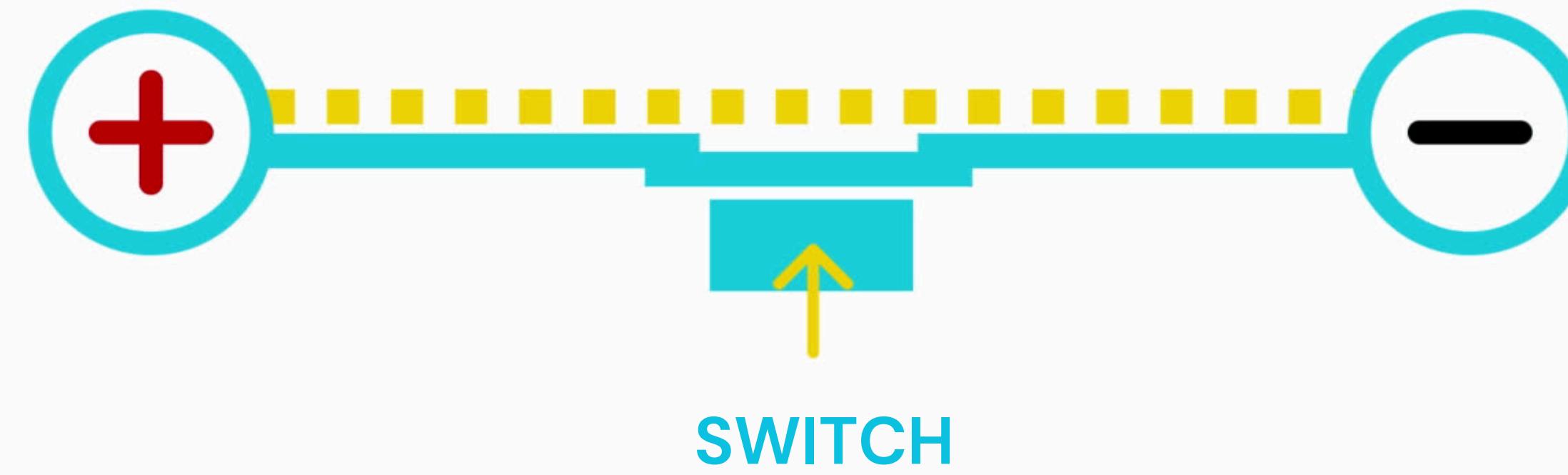
Negative

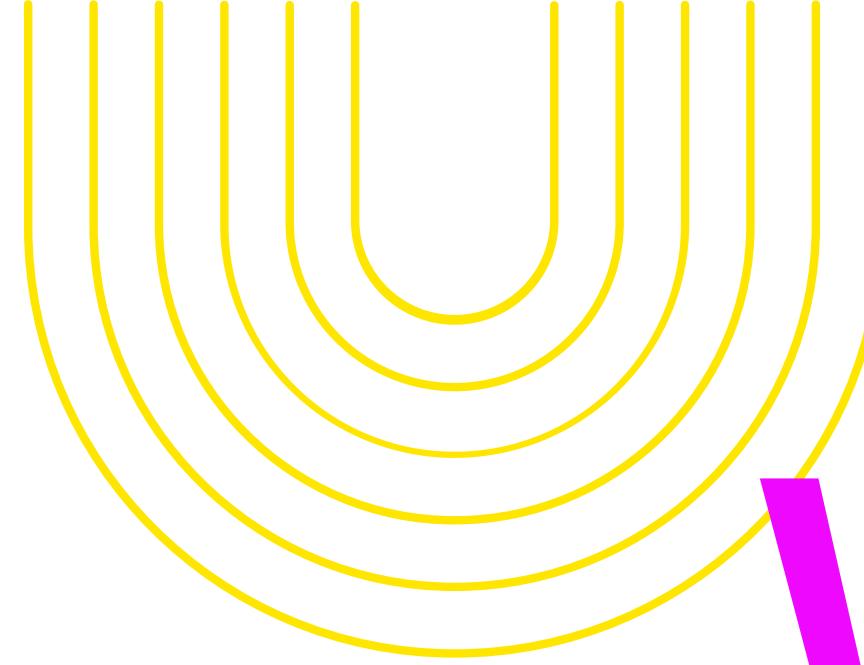
**Positive and negative must be connected for electricity to flow**

When the connection is  
broken nothing will work



We can use this to our advantage to add switches, or know why our project isn't working



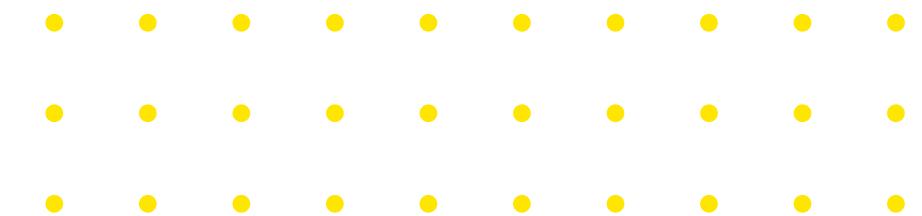


# WHAT IF?

We want to flick 3 switches and turn a dimmer all at the same time in response to a single input?



FOLLOW ALONG USING THE QR CODE OR AT:  
[https://github.com/CreativeRoboticsClub/T3\\_2024/tree/main/CRC\\_Week\\_03](https://github.com/CreativeRoboticsClub/T3_2024/tree/main/CRC_Week_03)

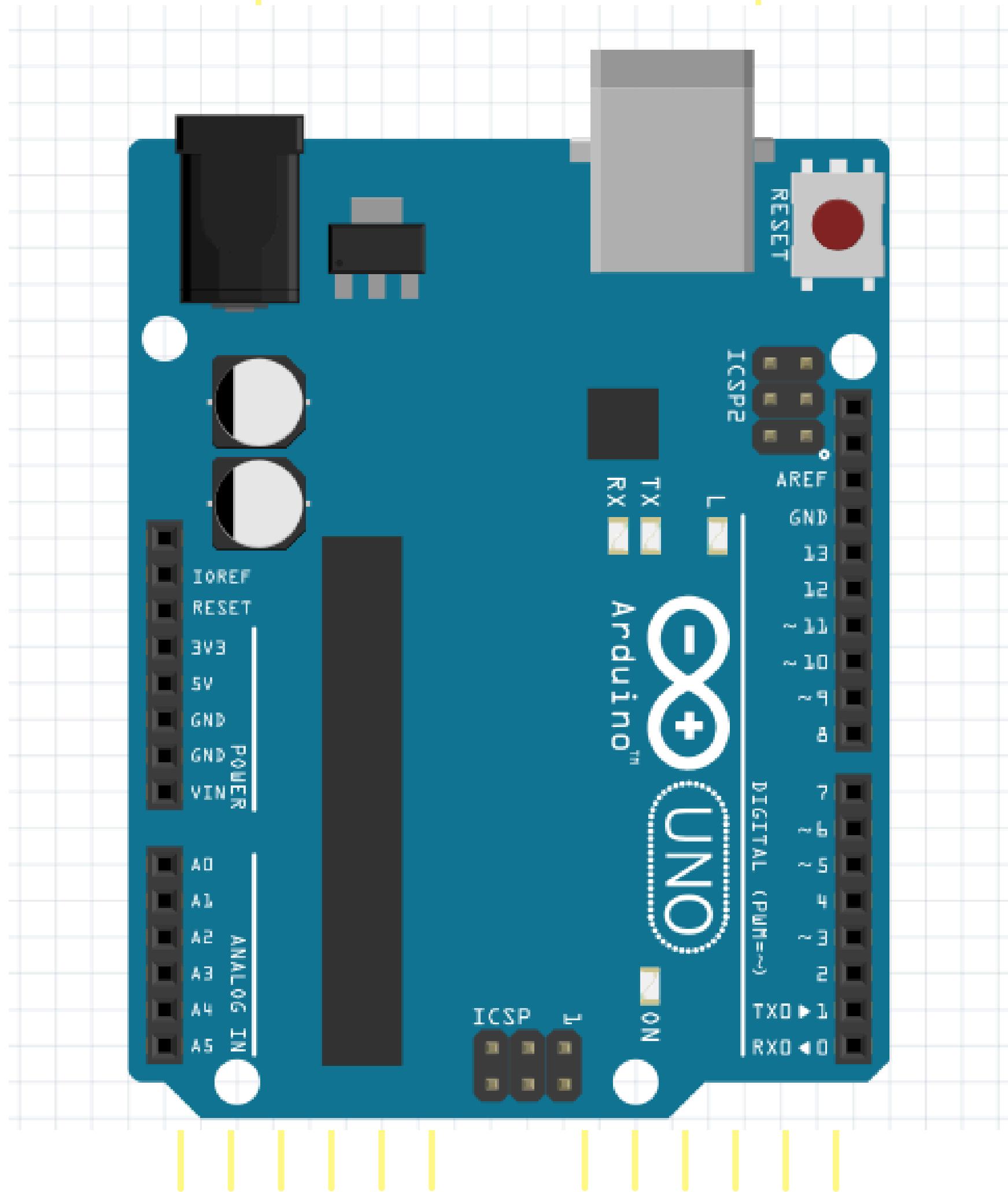


# ARDUINO

This is an **Arduino Uno**

The easiest way to think about it is as a box of dimmers and switches

It can also read in information. We can use that information to drive things with electricity, we'll talk more about that next week



# ARDUINO

It has 4 main sections:

**Power**

**Analog in** (Read sensors)

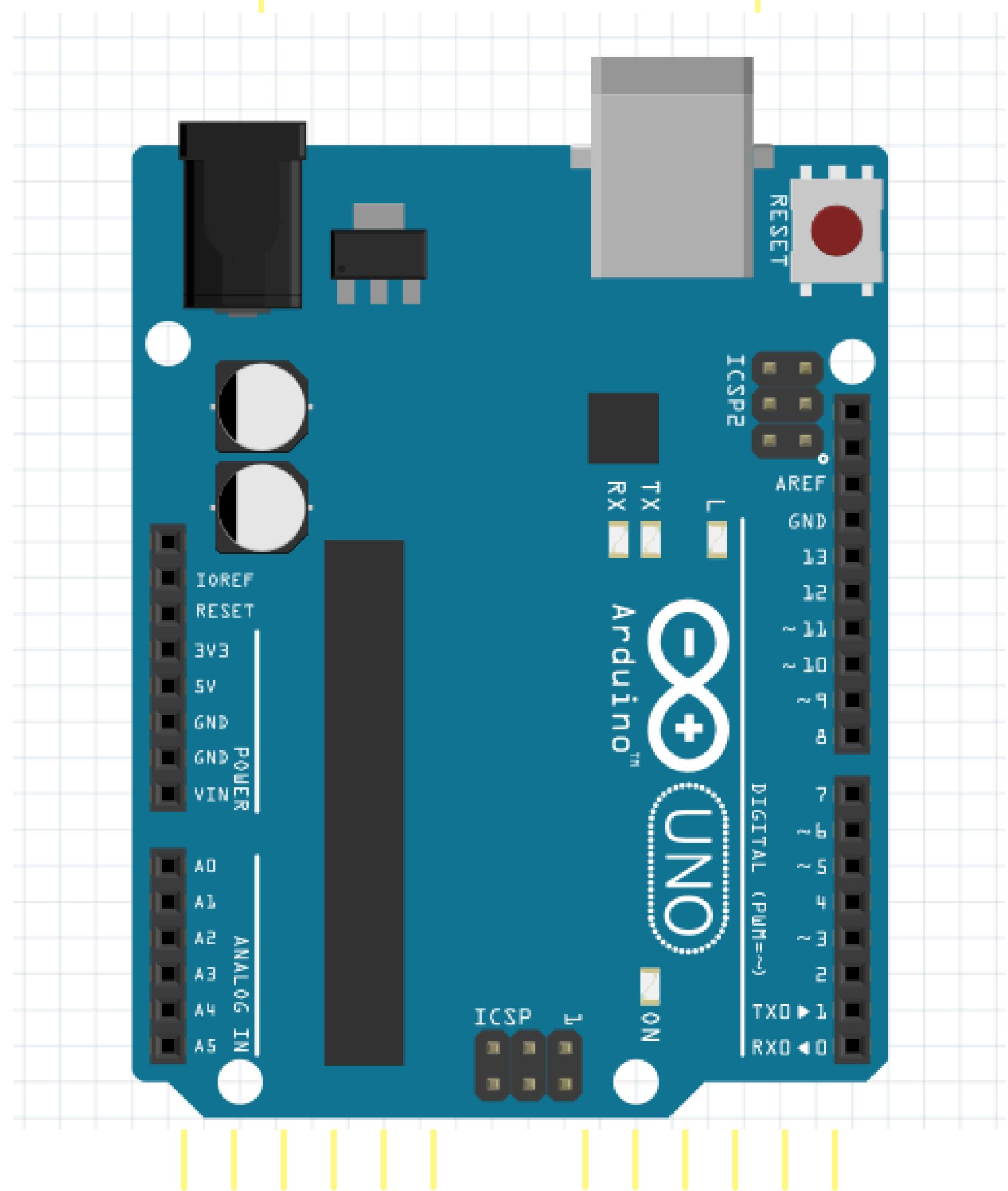
These read in dimmer  
information

**Digital I/O** (Input / Output)

These are our switches

**PWM~** (Fake analog out)

These are our dimmers



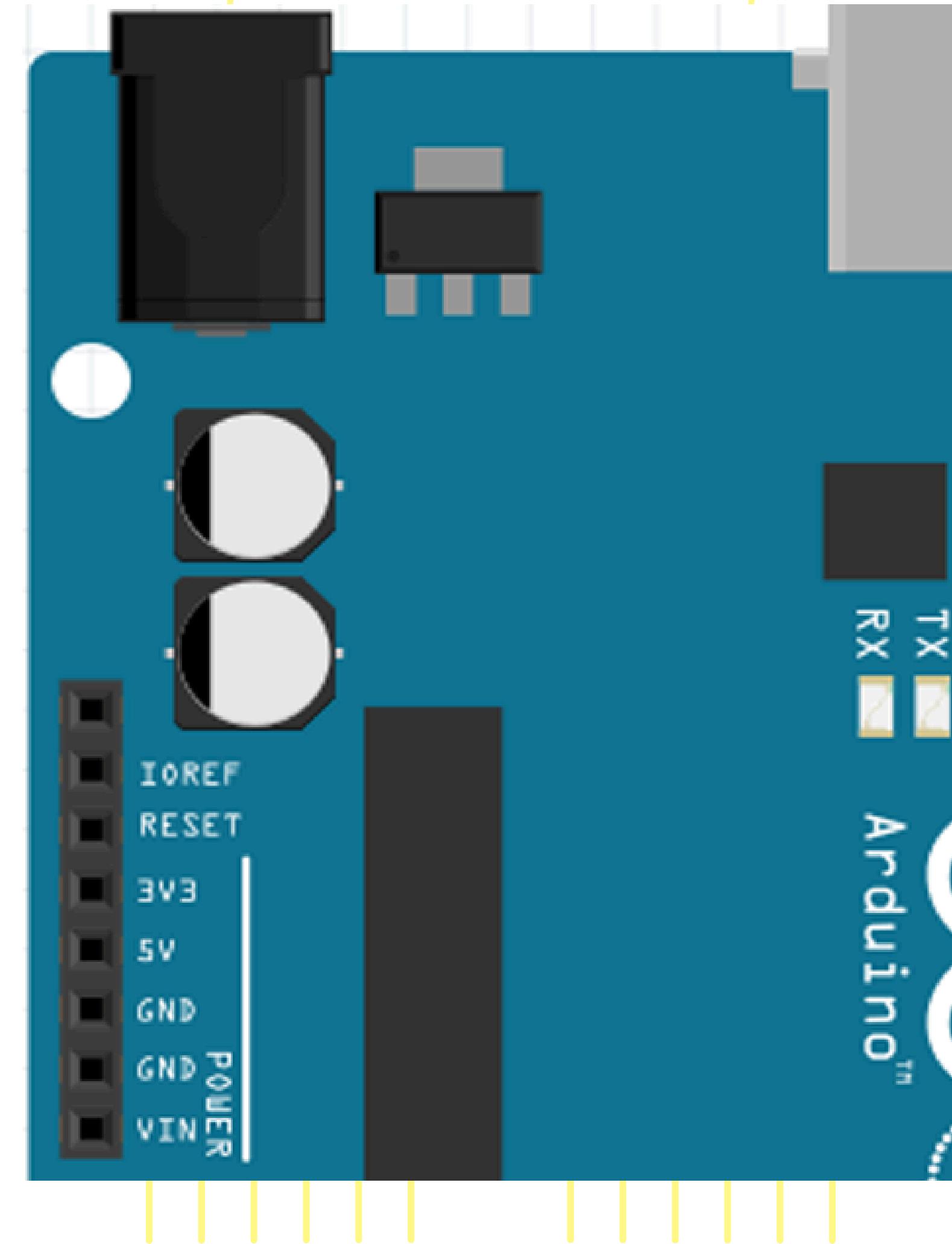
# ARDUINO

Power pins are located in the top left

You will notice how they relate to basic electrical ideas we discussed earlier

There are 2-3 different voltage sources (**5v**, **3.3v**, **Vin**) and two GND (Ground / -) pins

Use these to power sensors and low voltage components



# ARDUINO

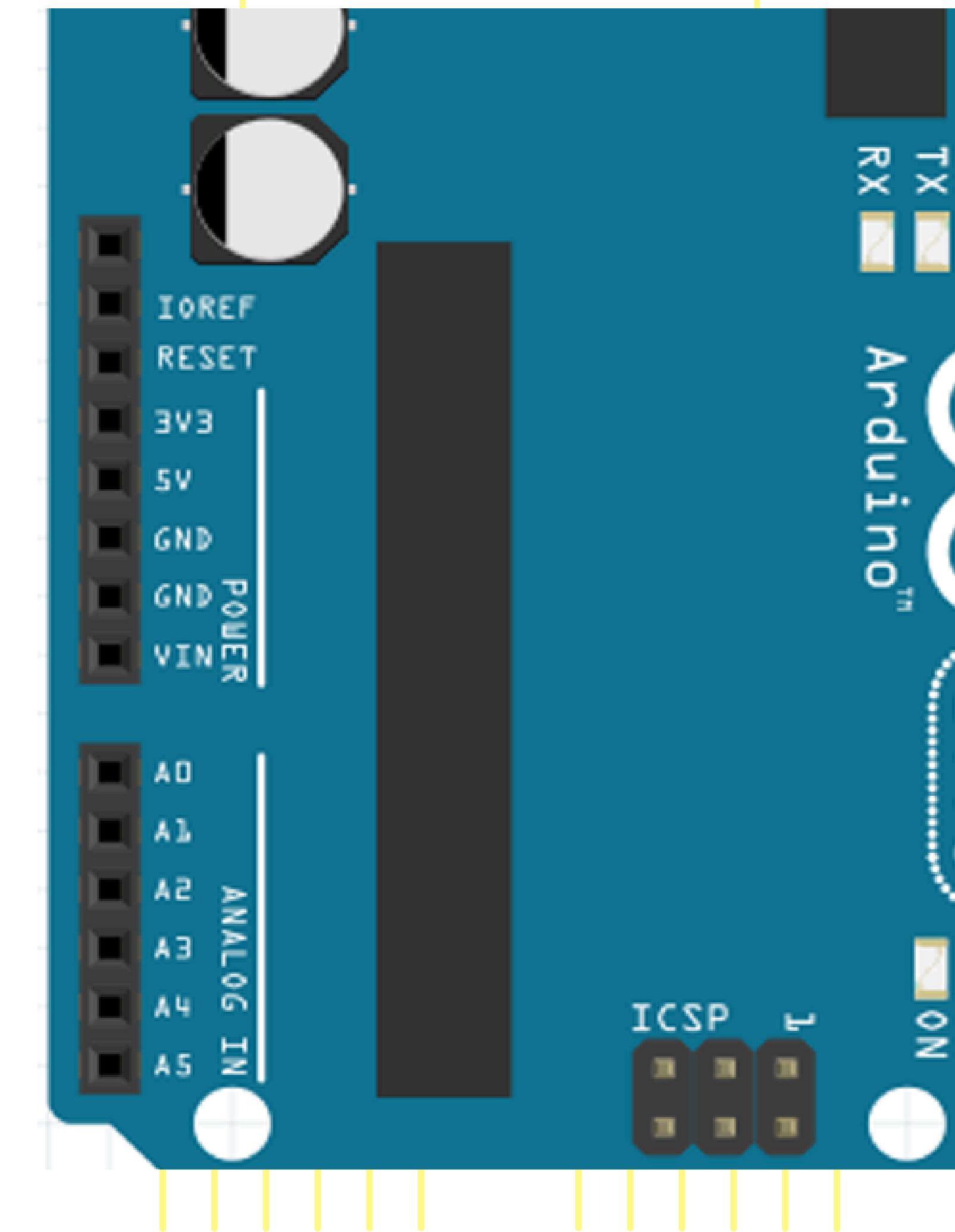
Analog In pins are located in the bottom left

This section can be thought of as being used for reading dimmer / volume knobs

They turn electrical signals into numbers between  
0 - 1023

Accessed in code with  
`analogRead(#);`

# is the pin number (0 - 5)

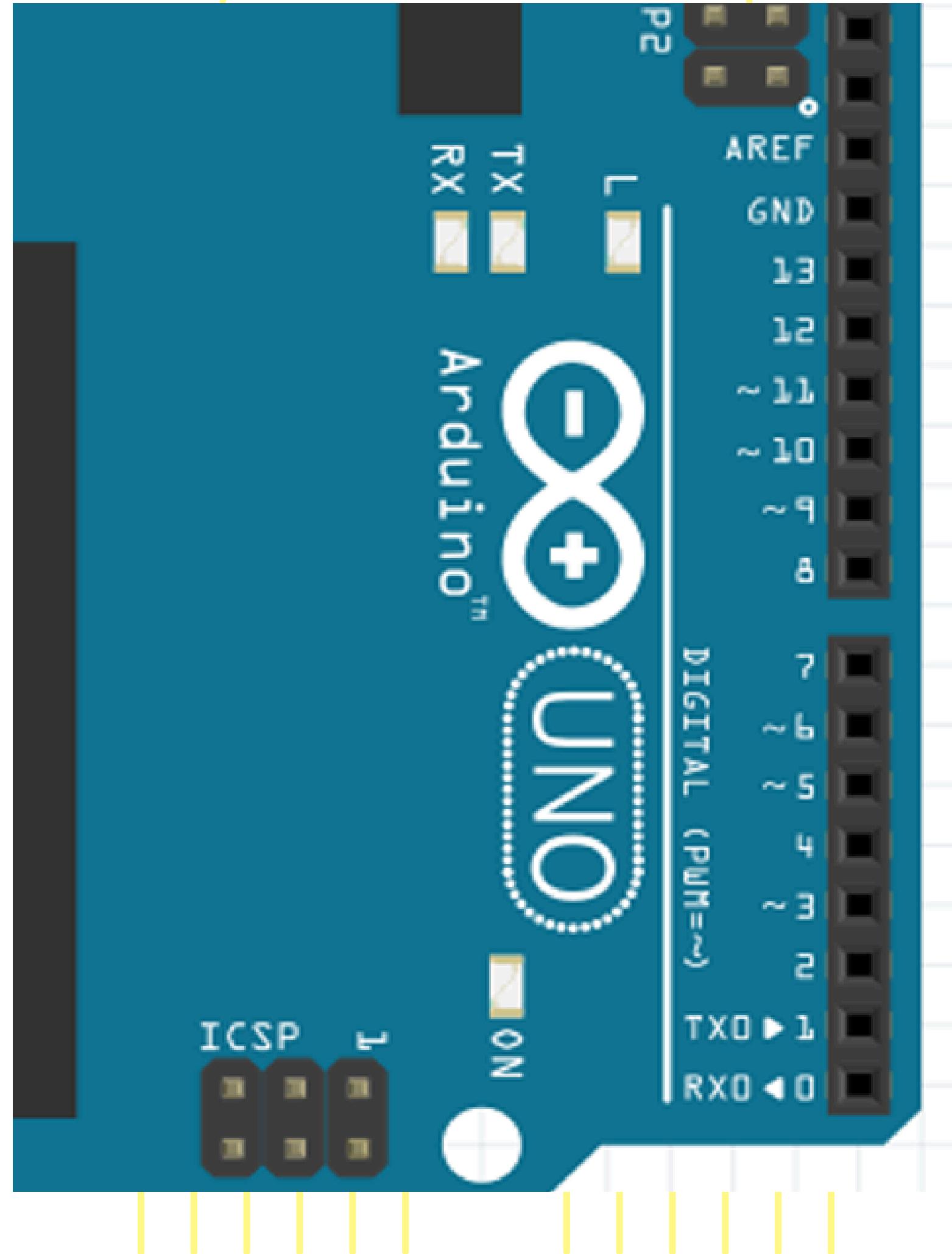


# ARDUINO

Digital I/O pins are located on the right

They can read signals as 0 or 1 (**LOW** or **HIGH**, off or on)

They can be used to turn devices on (**HIGH / 5v**) and off (**LOW / GND**), or check for button presses and more



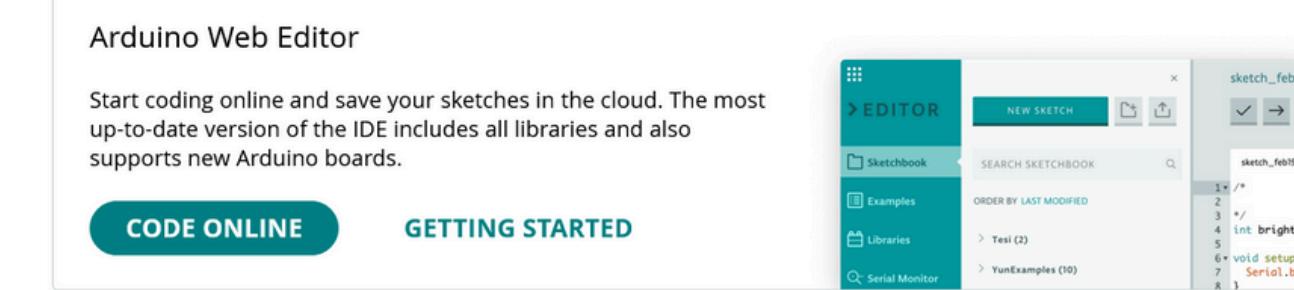
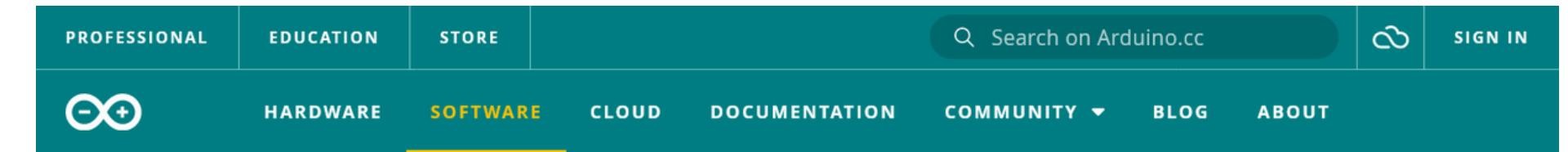
**“Okay, but how do I use it??”**

# ARDUINO

First download the  
Arduino IDE

This is the easiest way to  
program the Arduino

You can get it from:  
[arduino.cc/en/software](https://arduino.cc/en/software)



Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

[CODE ONLINE](#) [GETTING STARTED](#)

## Downloads



### Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

#### SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is

**DOWNLOAD OPTIONS**

**Windows** Win 10 and newer, 64 bits  
**Windows** MSI installer  
**Windows** ZIP file

**Linux** AppImage 64 bits (X86-64)  
**Linux** ZIP file 64 bits (X86-64)

**macOS** Intel, 10.14: "Catalina" or newer, 64 bits  
**macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

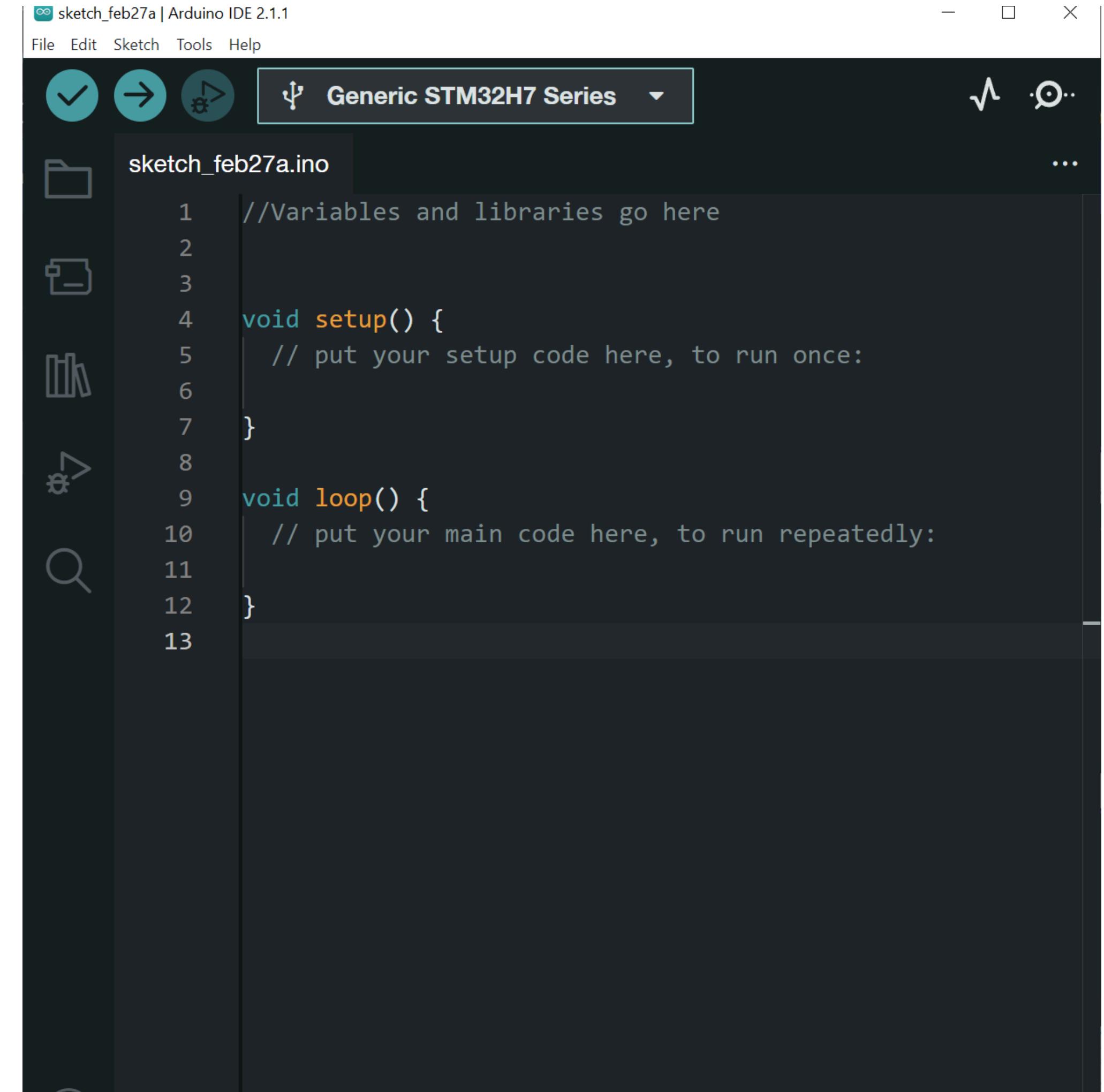
[Help](#)

# ARDUINO

Open the  
Arduino IDE

You will see something that  
looks a lot like this.

## Don't be scared!



The screenshot shows the Arduino IDE interface with a dark theme. At the top, the title bar reads "sketch\_feb27a | Arduino IDE 2.1.1". Below the title bar are menu options: File, Edit, Sketch, Tools, and Help. To the right of the menu are three circular icons: a checkmark, a right-pointing arrow, and a gear. Next to these is a USB icon followed by the text "Generic STM32H7 Series" with a dropdown arrow. On the far right are three small icons: a checkmark, a magnifying glass, and an ellipsis. The main area is a code editor with a dark background and light-colored text. It displays a file named "sketch\_feb27a.ino". The code is as follows:

```
//Variables and libraries go here
void setup() {
    // put your setup code here, to run once:
}
void loop() {
    // put your main code here, to run repeatedly:
}
```



**Coding is like  
cooking**



## Pumpkin soup

2 servings    15 minutes

### INGREDIENTS

100 ml milk  
50 g butter  
3 eggs  
1 tbs cocoa  
2 tsp baking soda  
a pinch of salt  
3 eggs

### NOTES

Nunc nulla velit, feugiat vitae ex quis, lobortis porta leo. Donec dictum lectus in ex accumsan sodales. Pellentesque habitant morbi tristique.

### DIRECTIONS

1. Nunc nulla velit, feugiat vitae ex quis, lobortis porta leo.
2. Donec dictum lectus in ex accumsan sodales. Pellentesque habitant morbi tristique.
3. Nunc nulla velit, feugiat vitae ex quis, lobortis porta leo. Donec dictum lectus in ex. lentesque habitant morbi tristique. Nunc nulla velit, feugiat vitae ex quis, lobortis porta leo. Donec dictum lectus in ex.
4. Habitant morbi tristique. Nunc nulla velit, feugiat vitae ex quis, lobortis porta leo. Donec dictum lectu,
5. Donec dictum lectus in ex accumsan sodales. Pellentesque habitant morbi tristique.
6. Nunc nulla velit, feugiat vitae ex quis, lobortis porta leo. Donec dictum lectus in ex. lobortis porta leo.

# RECIPES HAVE 3 SECTIONS

## Ingredients

What are we cooking with?

## Preperation

How do we prepare these ingredients before we cook them?

## Cooking

What steps do we need to take to make the meal?

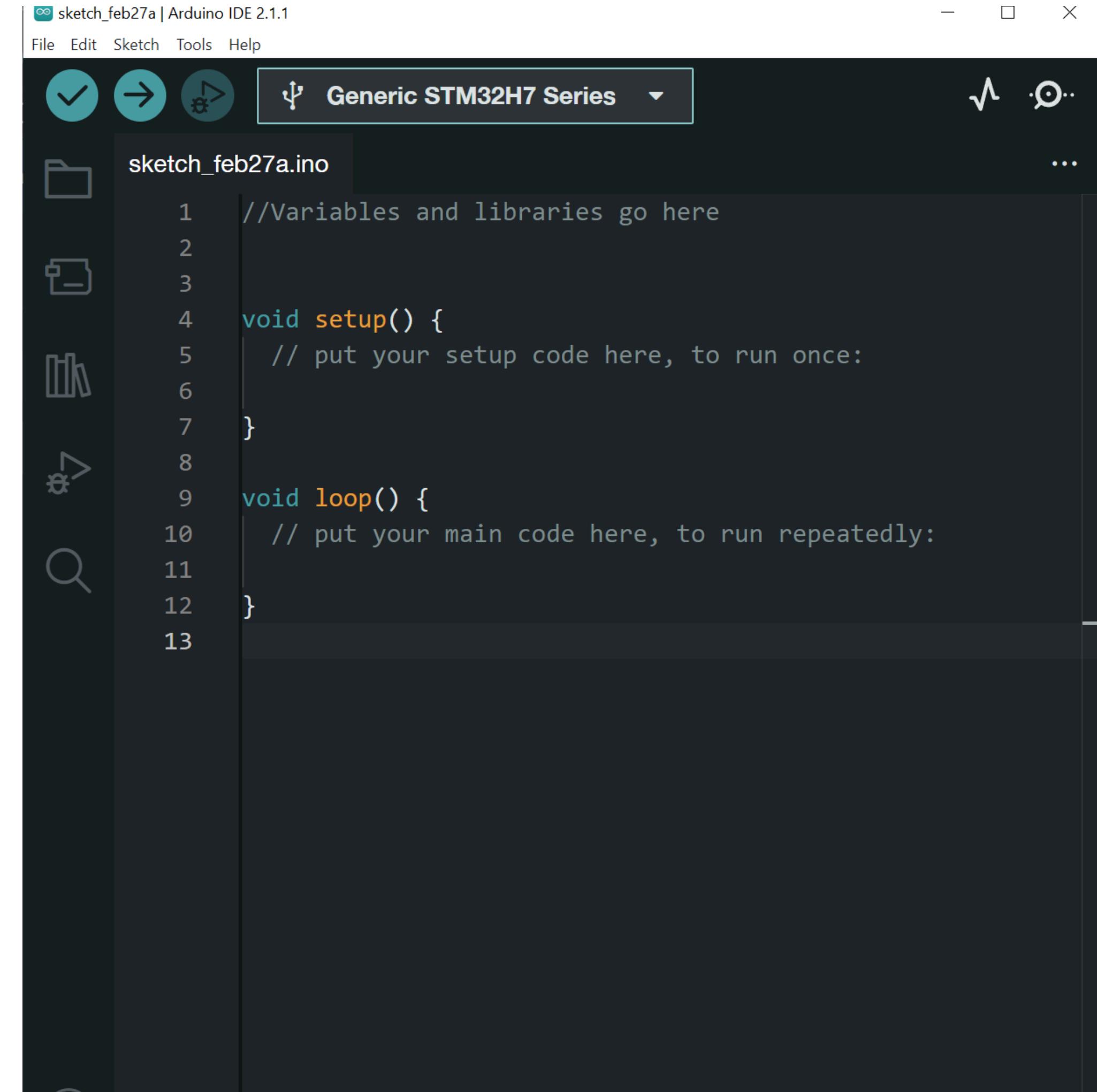
# PROGRAMMING IS THE SAME!

# THINK OF IT LIKE A RECIPE

At the top we tell the program what ingredients we need. We call this **importing libraries** and declaring **variables**.

In **void setup()** we tell it how to prepare those ingredients. What are the starting values for our variables?

And in **void loop()** we tell it what it is we're doing.



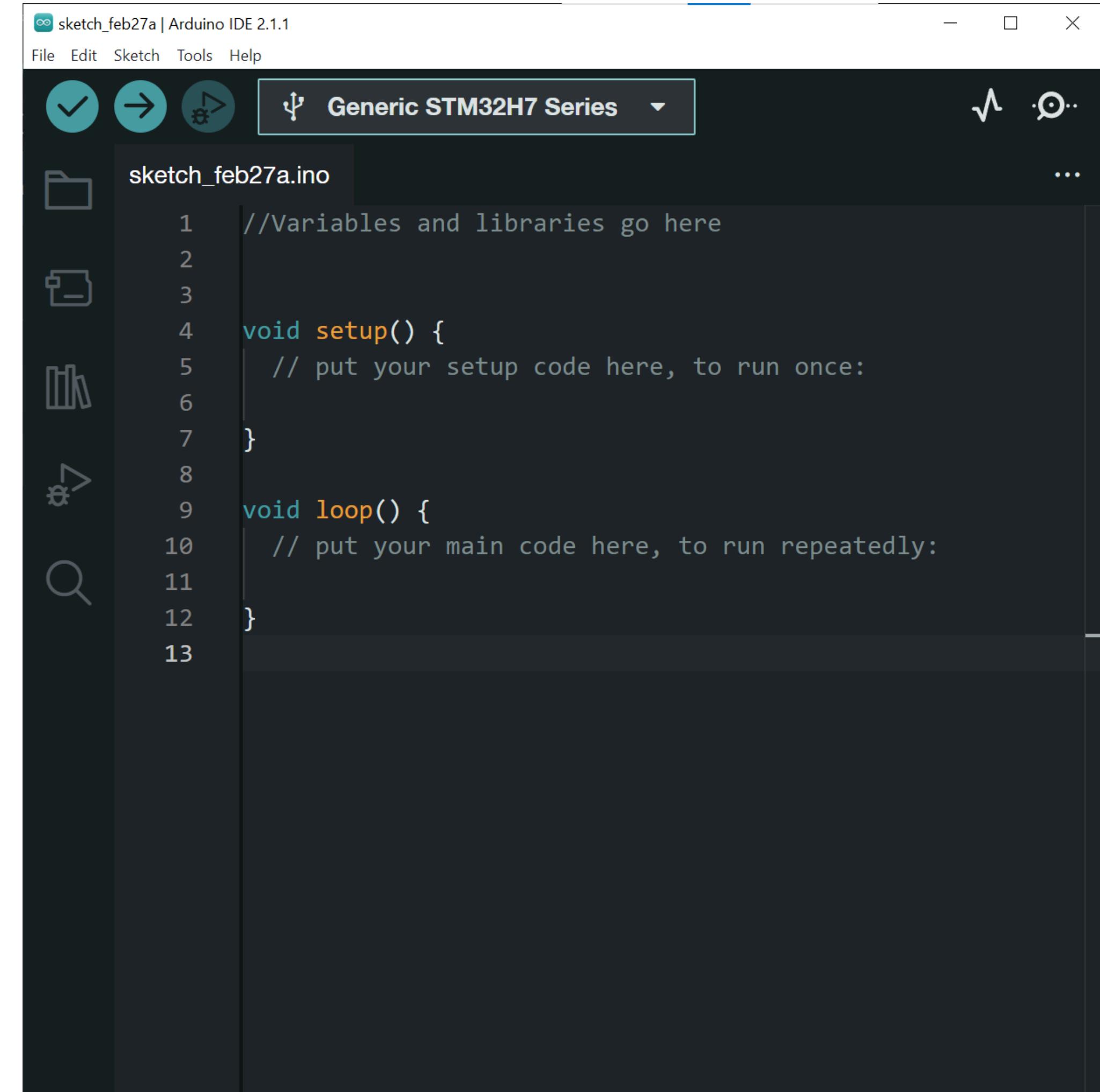
```
sketch_feb27a | Arduino IDE 2.1.1
File Edit Sketch Tools Help
✓ → 🔍 USB Generic STM32H7 Series ...
sketch_feb27a.ino
1 //Variables and libraries go here
2
3
4 void setup() {
5     // put your setup code here, to run once:
6
7 }
8
9 void loop() {
10    // put your main code here, to run repeatedly:
11
12 }
13
```

# THINK OF IT LIKE A RECIPE

We declare our **variables** once.

**void setup()** only runs at the start of our program – when the board powers on. It prepares our variables

**void loop()** runs after **void setup()**, looping over and over again while the board is powered on.



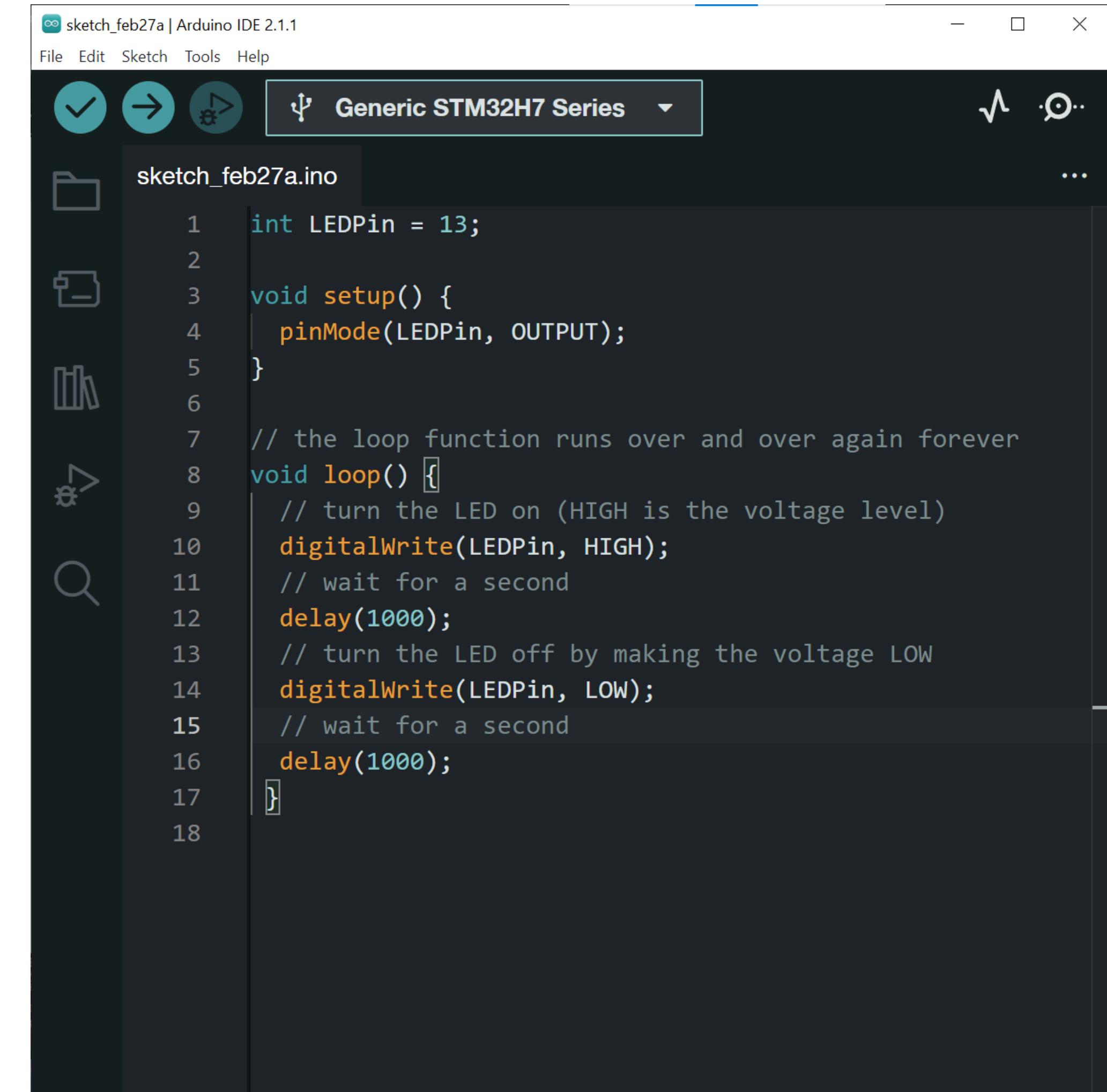
```
sketch_feb27a | Arduino IDE 2.1.1
File Edit Sketch Tools Help
USB Generic STM32H7 Series ...
sketch_feb27a.ino
1 //Variables and libraries go here
2
3
4 void setup() {
5     // put your setup code here, to run once:
6
7 }
8
9 void loop() {
10    // put your main code here, to run repeatedly:
11
12 }
13
```

# HERE'S AN EXAMPLE

At the top we define a variable called **LEDPin**. We will use this to tell the Arduino where the LED is.

In **void setup()** we use **pinMode()** tell the Arduino which pin we want to use, and what we want to use it for: **INPUT** or **OUTPUT**

The syntax is:  
**pinMode(pin number, INPUT or OUTPUT)**



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch\_feb27a | Arduino IDE 2.1.1
- Toolbar:** Includes icons for Save, Upload, and Preferences.
- Board Selector:** Generic STM32H7 Series
- Sketch List:** sketch\_feb27a.ino
- Code Editor:** Displays the following C++ code:

```
1 int LEDPin = 13;
2
3 void setup() {
4     pinMode(LEDPin, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     // turn the LED on (HIGH is the voltage level)
10    digitalWrite(LEDPin, HIGH);
11    // wait for a second
12    delay(1000);
13    // turn the LED off by making the voltage LOW
14    digitalWrite(LEDPin, LOW);
15    // wait for a second
16    delay(1000);
17 }
18
```

# HERE'S AN EXAMPLE

In `void loop()` we use `digitalWrite()` tell the Arduino to send **5v** or GND.

**HIGH** = **5v**

**LOW** = GND

`delay()` uses milliseconds. So `delay(1000);` means "Wait 1 second."

```
sketch_feb27a | Arduino IDE 2.1.1
File Edit Sketch Tools Help
Generic STM32H7 Series
sketch_feb27a.ino
1 int LEDPin = 13;
2
3 void setup() {
4     pinMode(LEDPin, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     // turn the LED on (HIGH is the voltage level)
10    digitalWrite(LEDPin, HIGH);
11    // wait for a second
12    delay(1000);
13    // turn the LED off by making the voltage LOW
14    digitalWrite(LEDPin, LOW);
15    // wait for a second
16    delay(1000);
17 }
18
```

# SOME MORE TALK ABOUT VARIABLES

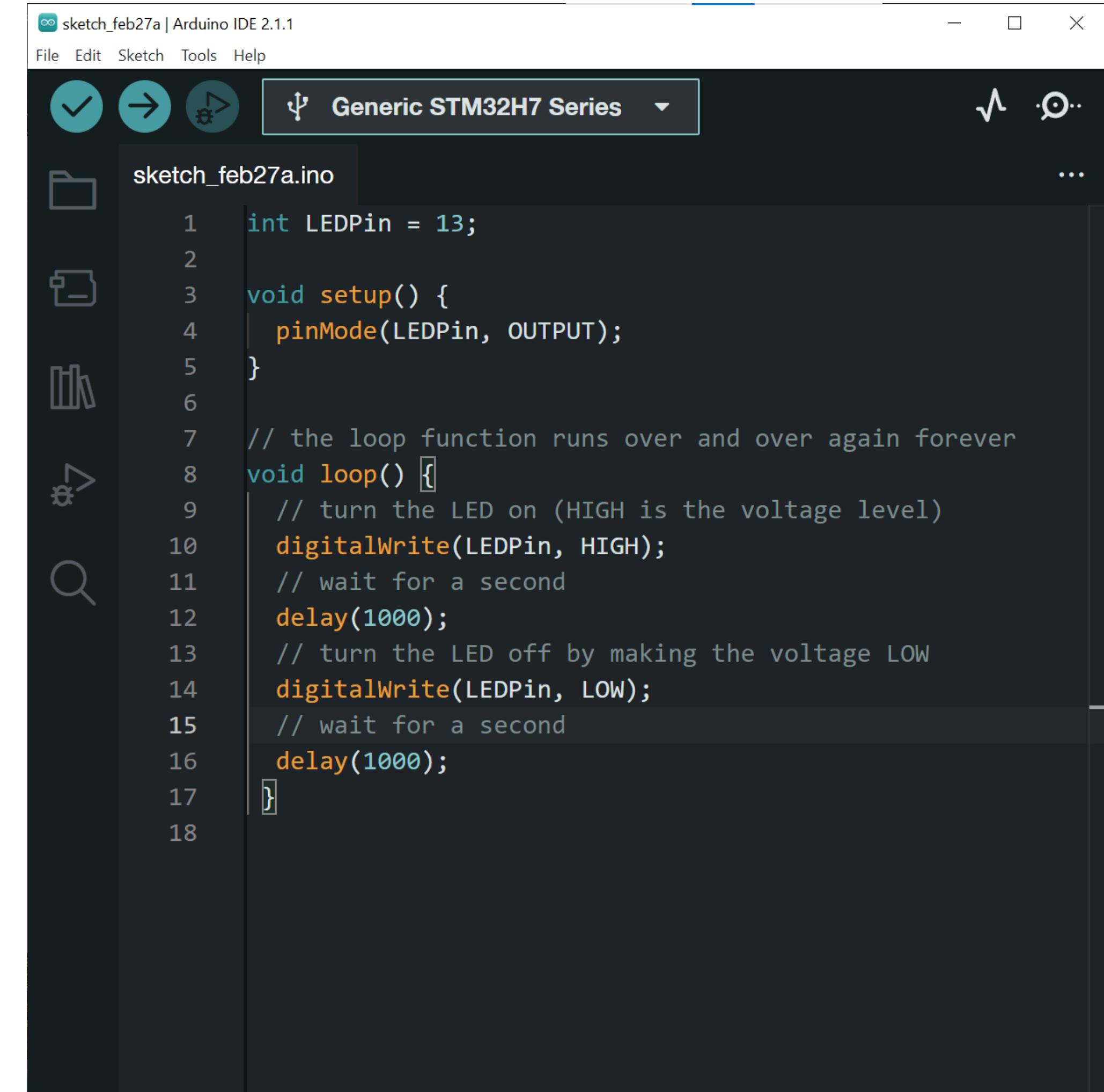
Variables tell the Arduino what to expect. Here are the two most common:

## int

Short for interger. A whole number, no decimal points allowed! e.g. 1, 256, 1025, 9, etc

## float

A number with a decimal place. e.g. 0.25; 12.6, 1033.444, etc



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch\_feb27a | Arduino IDE 2.1.1
- Toolbar:** Includes icons for Save, Upload, Refresh, and Preferences.
- Board Selector:** Generic STM32H7 Series
- Sketch List:** sketch\_feb27a.ino
- Code Editor:** Displays the following C++ code for an Arduino sketch:

```
1 int LEDPin = 13;
2
3 void setup() {
4     pinMode(LEDPin, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     // turn the LED on (HIGH is the voltage level)
10    digitalWrite(LEDPin, HIGH);
11    // wait for a second
12    delay(1000);
13    // turn the LED off by making the voltage LOW
14    digitalWrite(LEDPin, LOW);
15    // wait for a second
16    delay(1000);
17 }
18
```

# LET'S TRY IT OUT!

Type in the code and get it to upload.

Try changing the delay times.

Try adding more lines of code to change the pattern.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch\_feb27a | Arduino IDE 2.1.1
- Menu Bar:** File Edit Sketch Tools Help
- Board Selector:** Generic STM32H7 Series
- Sketch List:** sketch\_feb27a.ino
- Code Area:** The code listed below is the main sketch content.

```
int LEDPin = 13;
void setup() {
    pinMode(LEDPin, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
    // turn the LED on (HIGH is the voltage level)
    digitalWrite(LEDPin, HIGH);
    // wait for a second
    delay(1000);
    // turn the LED off by making the voltage LOW
    digitalWrite(LEDPin, LOW);
    // wait for a second
    delay(1000);
}
```

“GREAT, NOW  
WHAT ABOUT  
MOVING A SERVO  
MOTOR?”

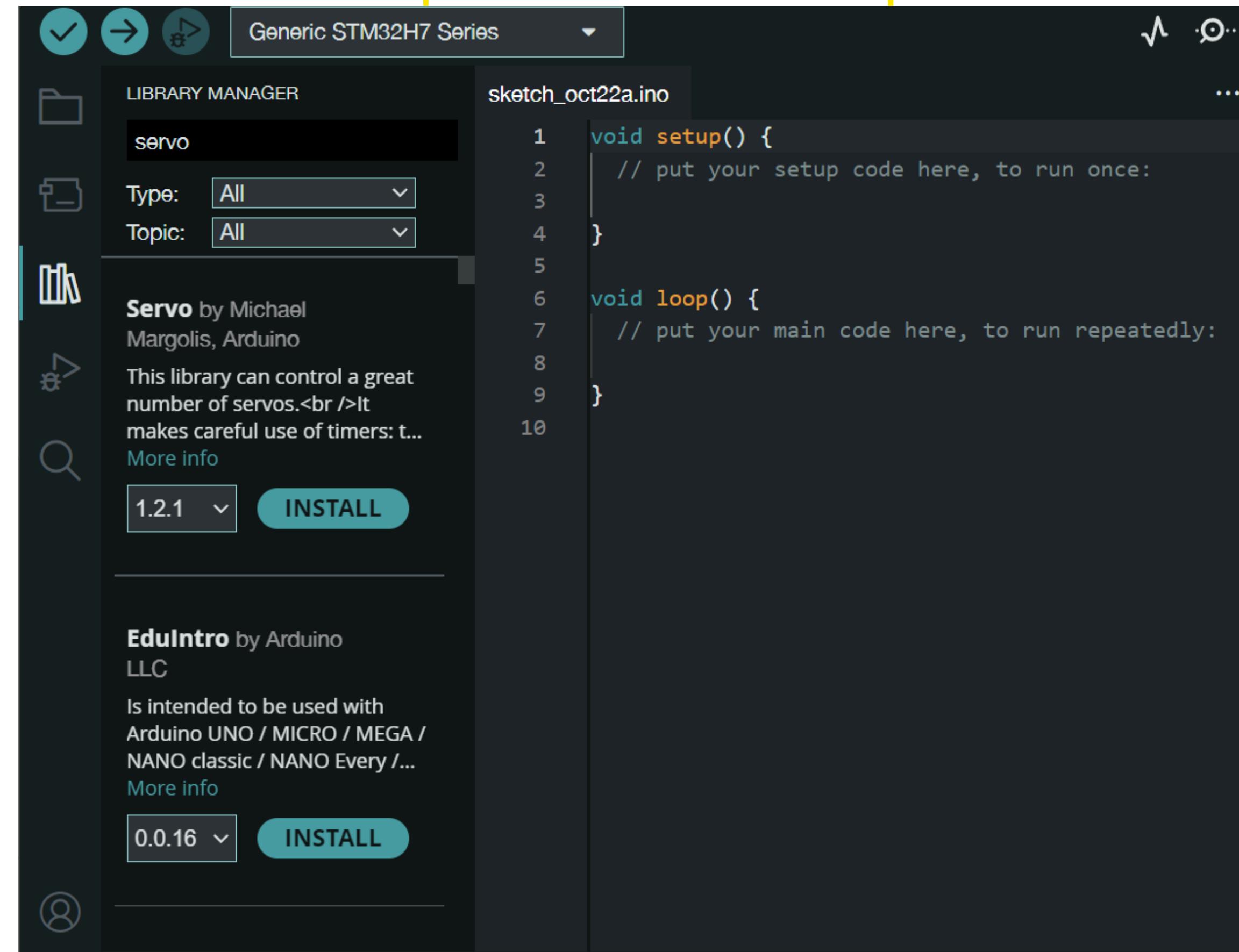


# SERVO MOTORS

Use Tools > Manage Libraries or the books icon to install the servo library

Libraries are blocks of code that are written by people with more experience than us to make complex tasks easier.

Search for the Servo library and make sure it is installed and up to date.



The screenshot shows the Arduino IDE's Library Manager interface. At the top, there are icons for saving, running, and settings, followed by a dropdown menu set to "Generic STM32H7 Series". Below this is a toolbar with a folder icon, a search icon, and a refresh icon. The main area is titled "LIBRARY MANAGER" and has a search bar containing "servo". Underneath are two library entries:

- Servo** by Michael Margolis, Arduino  
This library can control a great number of servos.  
It makes careful use of timers: t...  
[More info](#)  
Version: 1.2.1 [INSTALL](#)
- EduIntro** by Arduino LLC  
Is intended to be used with Arduino UNO / MICRO / MEGA / NANO classic / NANO Every /...  
[More info](#)  
Version: 0.0.16 [INSTALL](#)

To the right of the library list, a code editor window titled "sketch\_oct22a.ino" displays the following Arduino sketch:

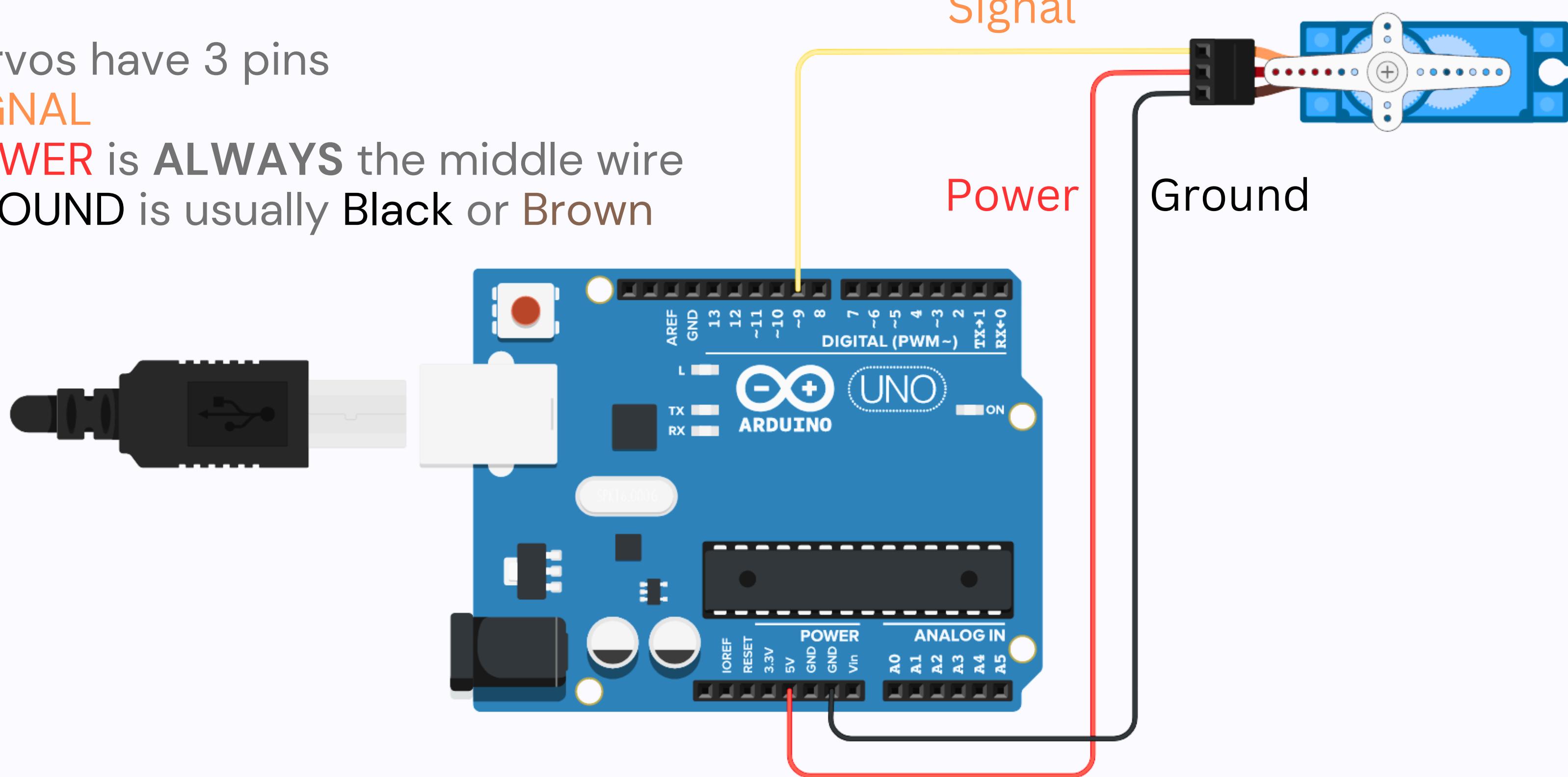
```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }  
4  
5 void loop() {  
6     // put your main code here, to run repeatedly:  
7 }  
8  
9 }  
10
```

# SERVO MOTORS

Servos have 3 pins

**SIGNAL**

**POWER** is **ALWAYS** the middle wire  
**GROUND** is usually Black or Brown



# SERVO MOTORS

Add the line

#include <Servo.h>

BEFORE void setup() to use it

Define your servo object using Servo  
<name>; names are case sensitive!

Lowercase **servo** is different from  
uppercase **Servo**.

Servos can be attached to any Digital  
I/O pin using the code  
**servo.attach(#);** in the **void setup**  
function

Then we use **servo.write(#);**  
to move them. # is between 0-180

sketch\_sep25a.ino

```
1  #include <Servo.h>
2
3  Servo servo;
4
5  void setup() {
6      //Tell Arduino which pin the Servo is connected to
7      servo.attach(9);
8      //Move the Servo to a starting position (0-180);
9      servo.write(90);
10     //Wait a second, so the Servo can move
11     delay(1000);
12 }
13
14 void loop() {
15     //Move the servo
16     servo.write(0);
17     //Give the servo time to move
18     delay(1000);
19     //Move the servo
20     servo.write(180);
21     //Give the servo time to move
22     delay(1000);
23
24 }
25
```

# SERVO MOTORS

Servo will:

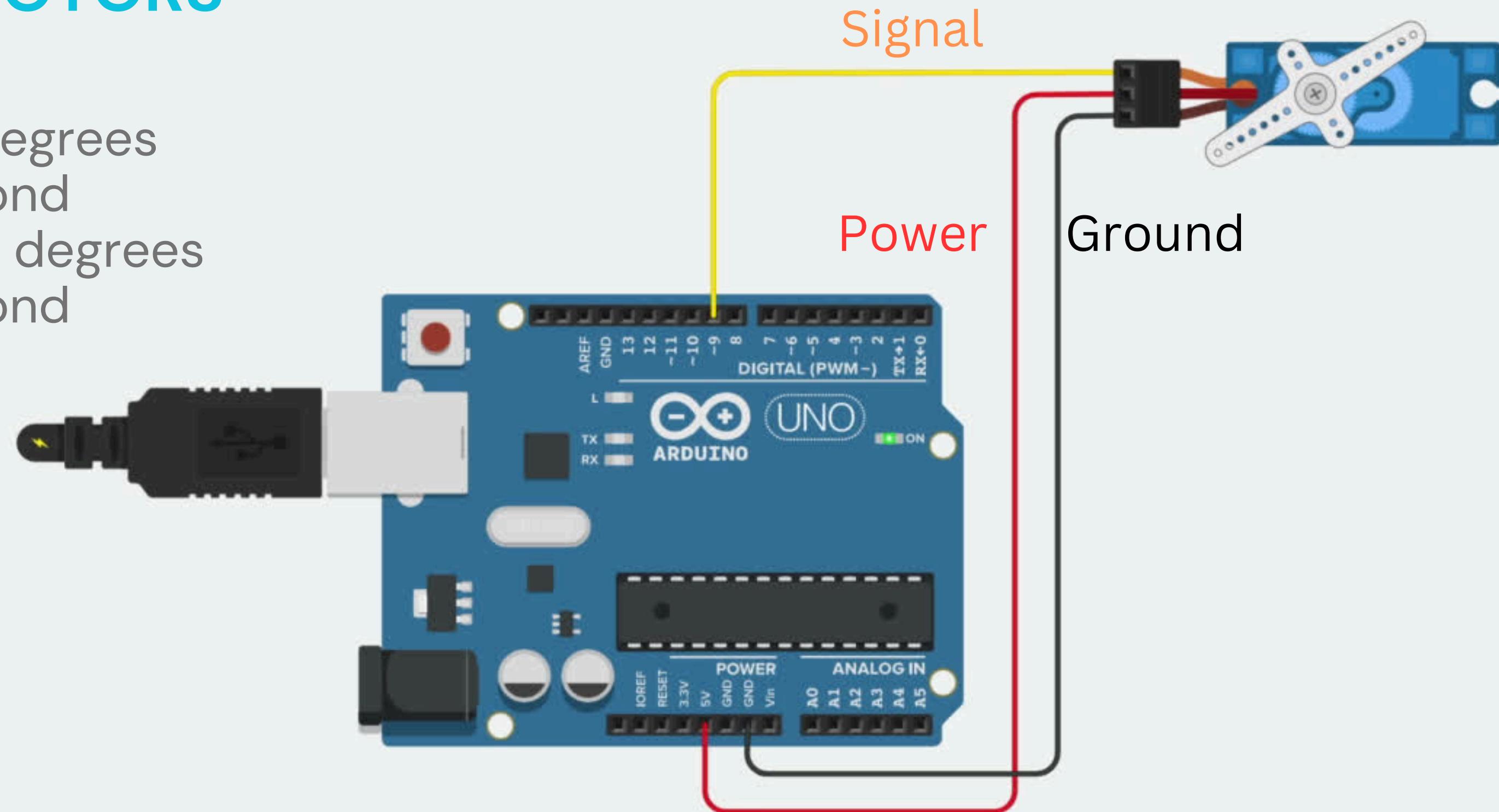
Move to 0 degrees

Pause 1 second

Move to 180 degrees

Pause 1 second

Repeat

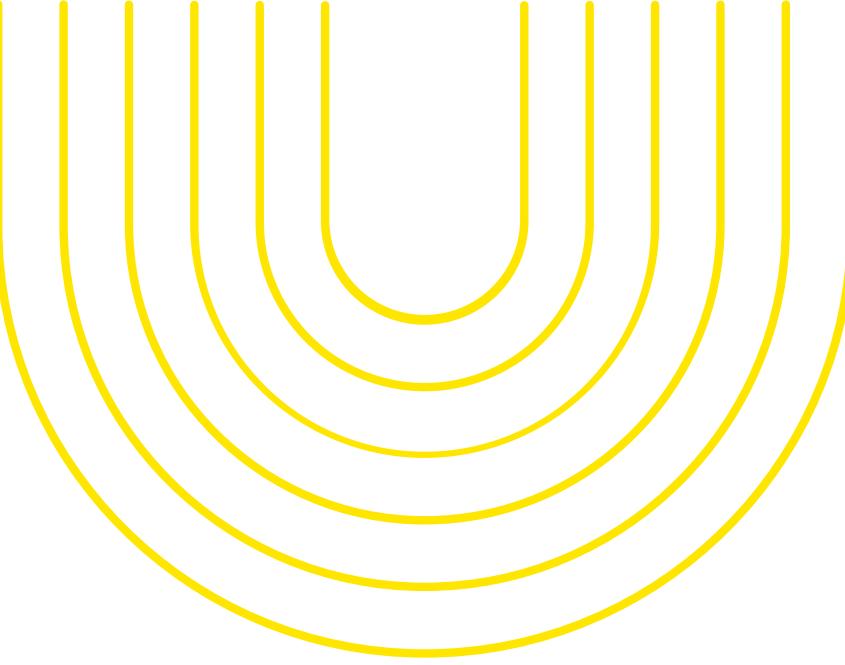


# SERVO MOTORS



sketch\_sep25a.ino

```
1 #include <Servo.h>
2
3 Servo servo;
4
5 void setup() {
6     //Tell Arduino which pin the Servo is connected to
7     servo.attach(9);
8     //Move the Servo to a starting position (0-180);
9     servo.write(90);
10    //Wait a second, so the Servo can move
11    delay(1000);
12 }
13
14 void loop() {
15     //Move the servo
16     servo.write(0);
17     //Give the servo time to move
18     delay(1000);
19     //Move the servo
20     servo.write(180);
21     //Give the servo time to move
22     delay(1000);
23
24 }
```



**NEED  
MORE  
ROBOTS?**



# Robot Royale



UNSW FOUNDERS



# Workshops!

Don't know how to build a robot? We will teach you!

Week 1 & 2 – Intro + kit handover

Week 3 – Sketching & cardboard prototyping

Week 4 – CAD, 3D printer slicing

Week 5 – Soldering & robot assembly

Week 7 – Radio Setup & control practice

Week 8 – Spinner / lifter design, brushless motors & servos

Week 9 – Mini practice tournament

Week 10 – Extras & Social event

MCIC MAKERSPACE

# MAKER NIGHT

UNSW FOUNDERS

5-8 PM EVERY  
TUESDAY

If you like to make things this is your chance to access the makerspace afterhours, join a community of makers, and work on all things creative!



PERSONAL  
PROJECTS



AFTER  
HOURS



SKILL  
SHARING      TOOLS  
                 AVAILABLE



NO SIGN-UP  
REQUIRED

# **Event page + workshop sign up**





**Last minute build day**

**December 7th**

**Tournament**

**December 8th**