



WELCOME TO THE
CREATIVE ROBOTICS CLUB

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

HOW DO WE RUN **CREATIVE ROBOTICS CLUB?**

WEEKS 2 - 5: Skill acquisition

We will learn new skills, try new ideas, grow our knowledge each week

WEEKS 7 - 10: Project support

Have the things you've learned in Weeks 2 -5 got you itching to make something? Do you have assignments that need electronics or programming support?
We are here to help.

WE ARE OPEN TO YOUR FEEDBACK!

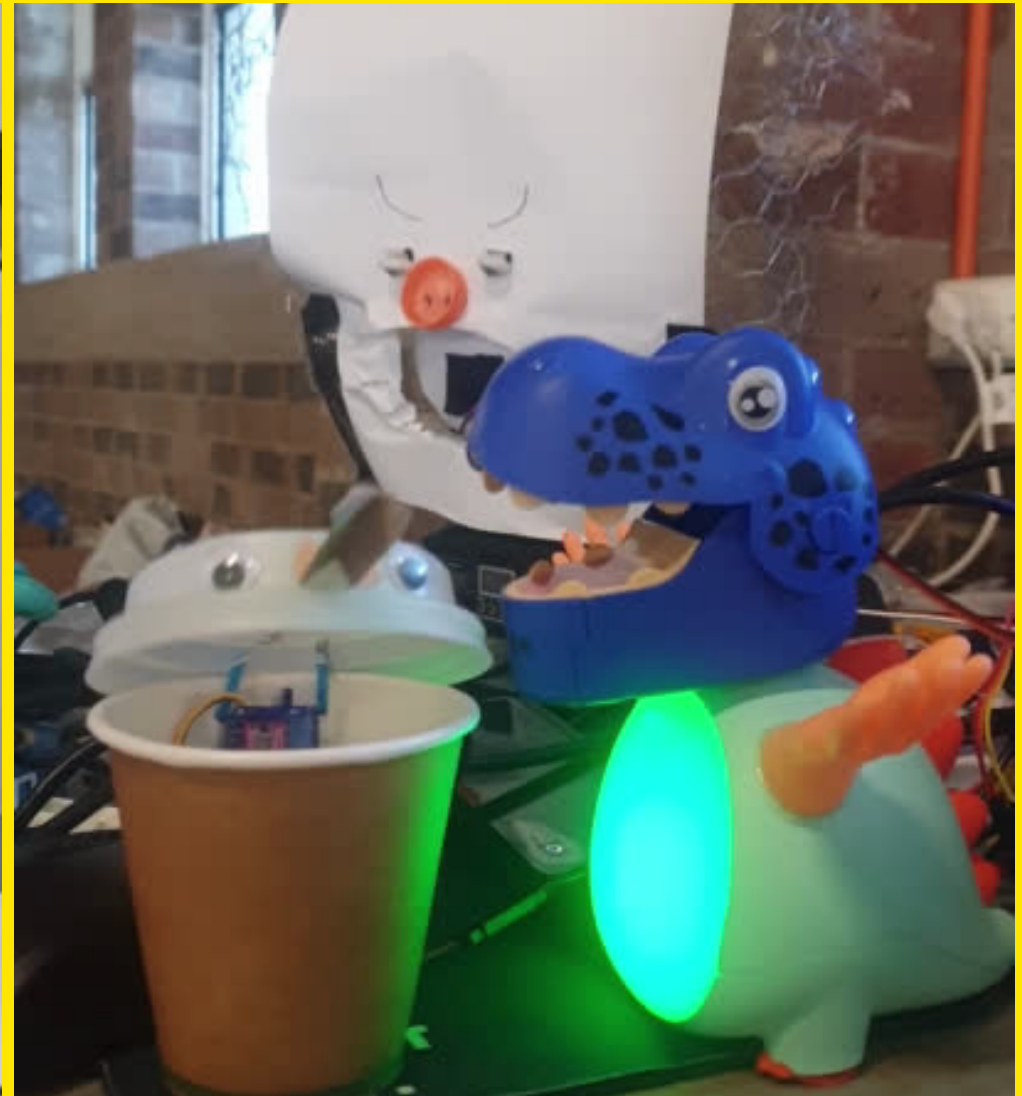
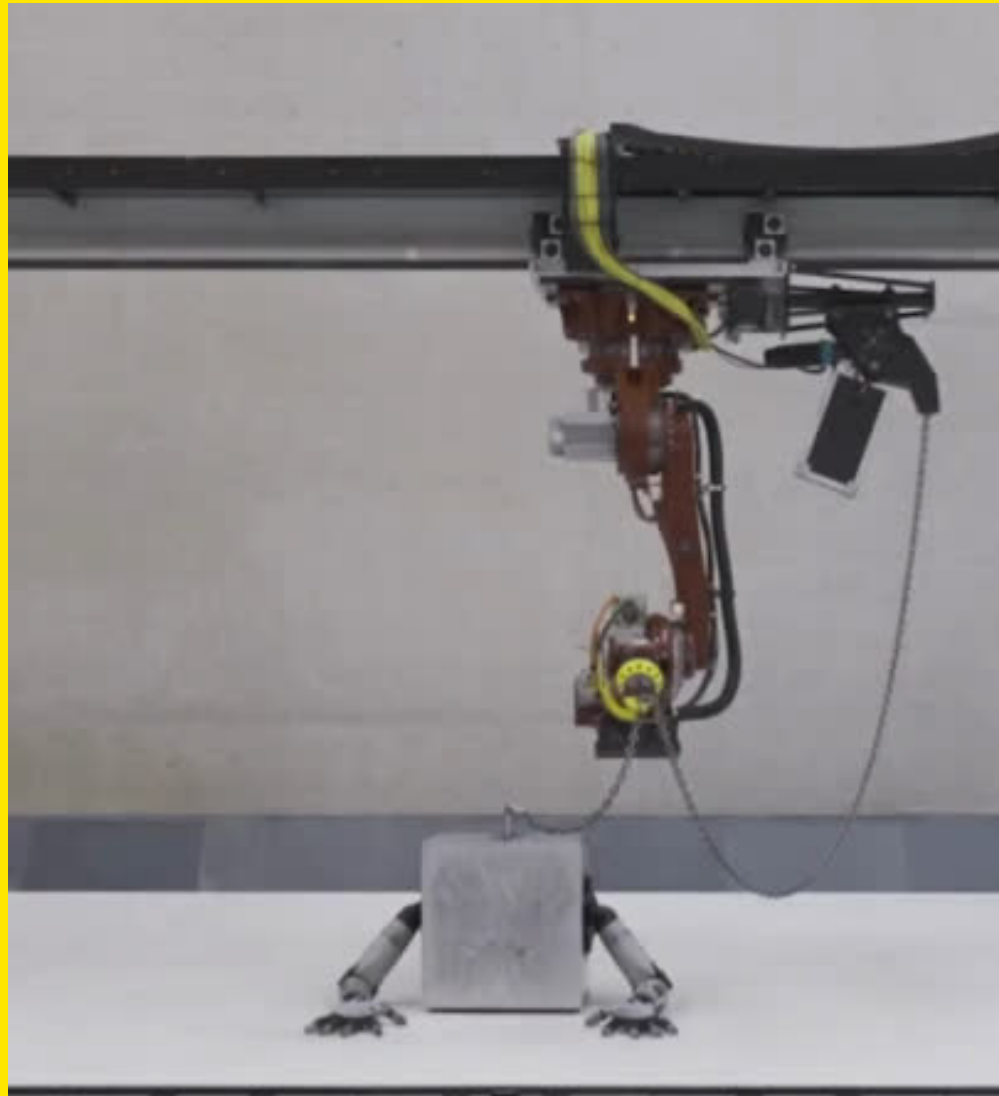
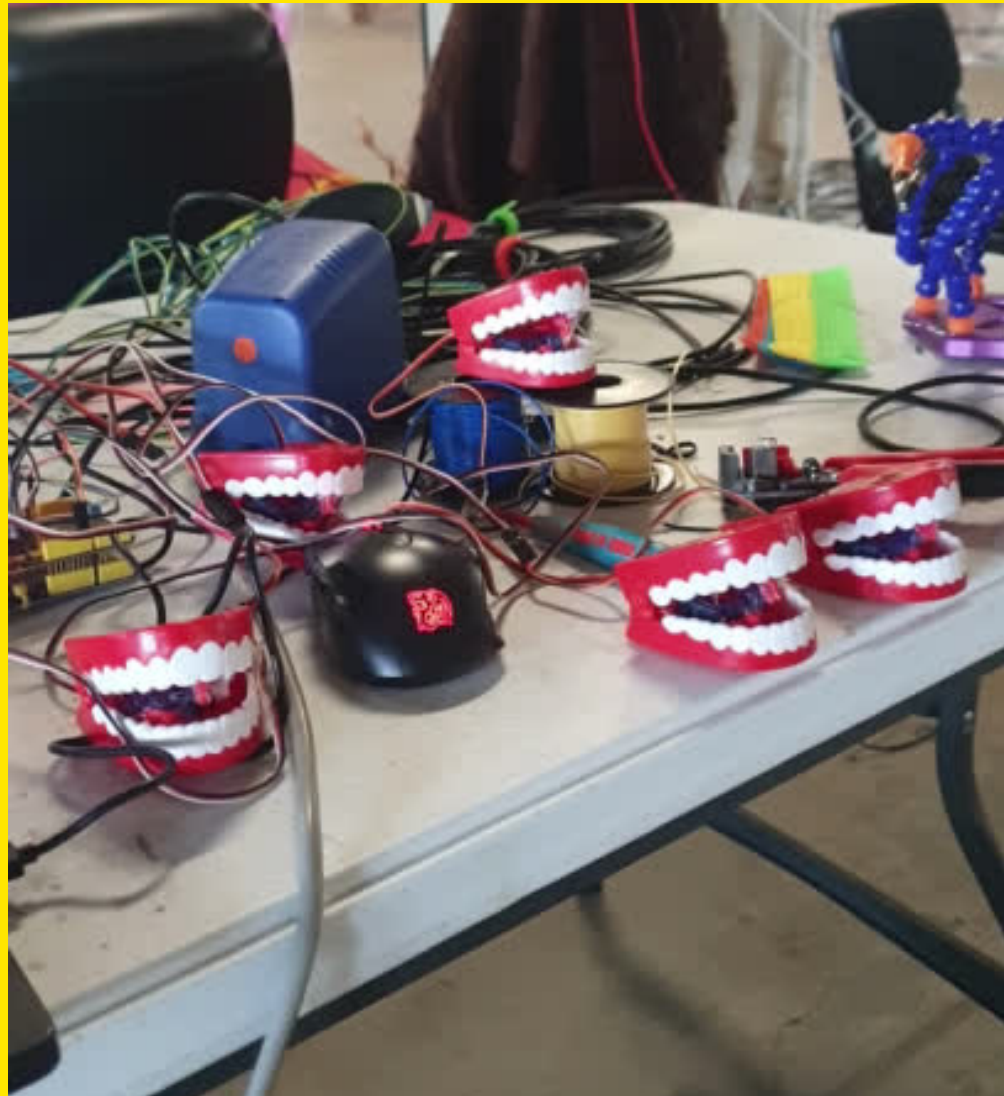
Are there things you want us to talk about?

A different way of running you think will work?

Skills you want to share?

We are a club for students, and we welcome your
suggestions and input

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



SERVO MOTORS

BUT FIRST LETS TALK ABOUT...

LAST WEEK

Last week:



DC gear motor



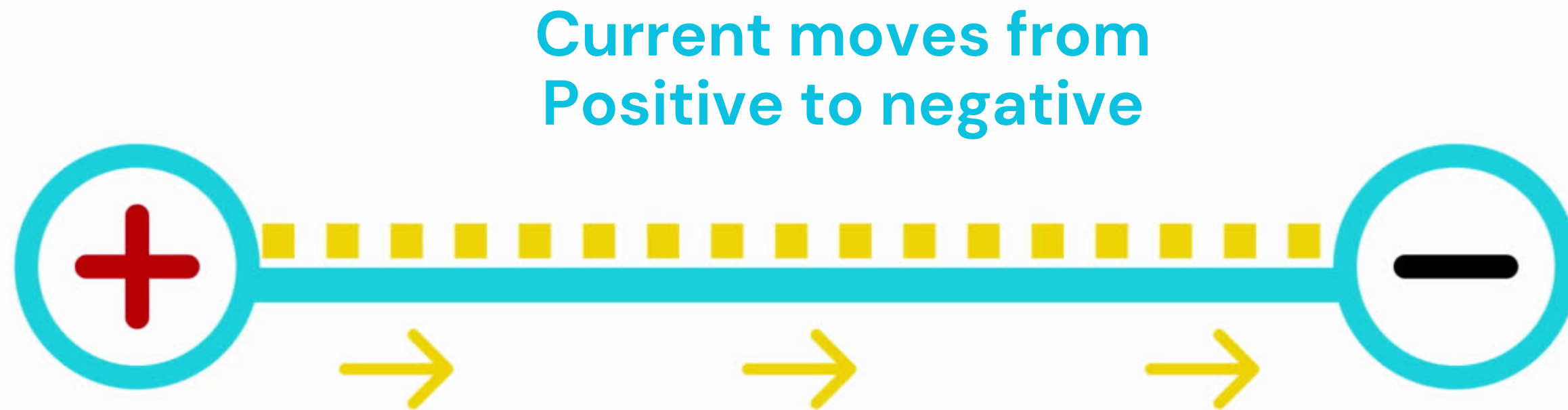
AA Battery [**1.5v**]

AA Battery holder



Switch [optional]

HOW DOES ELECTRICITY WORK?



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

Negative: GND, Ground, -, \perp

Positive



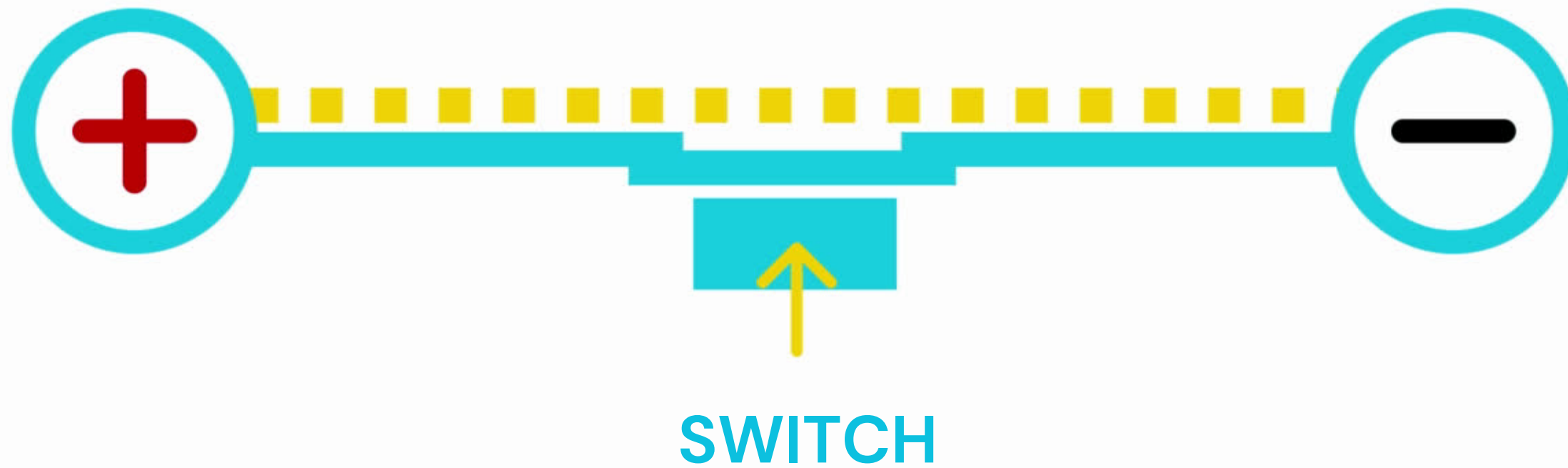
Negative

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

When the connection is
broken nothing will work



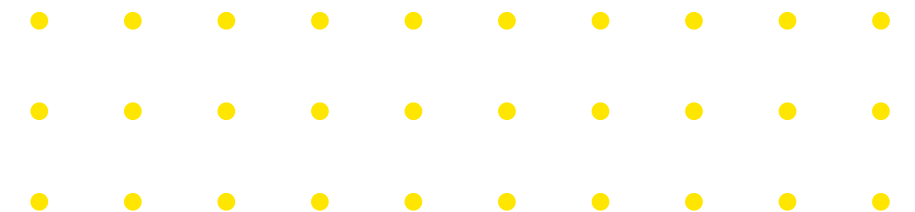
**We can use this to our advantage to
add switches, or know why our project
isn't working**





WHAT IF?

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

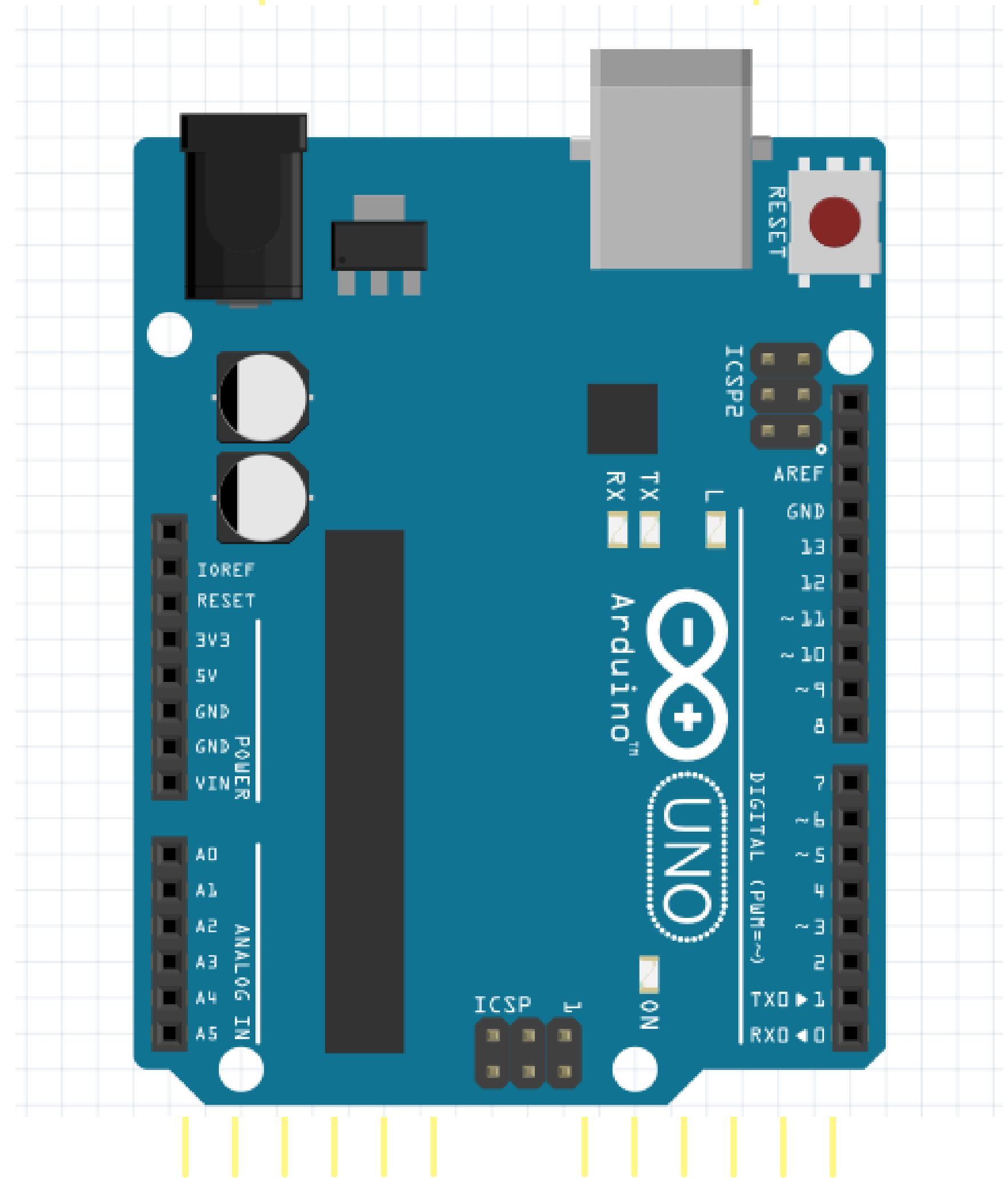


ARDUINO

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

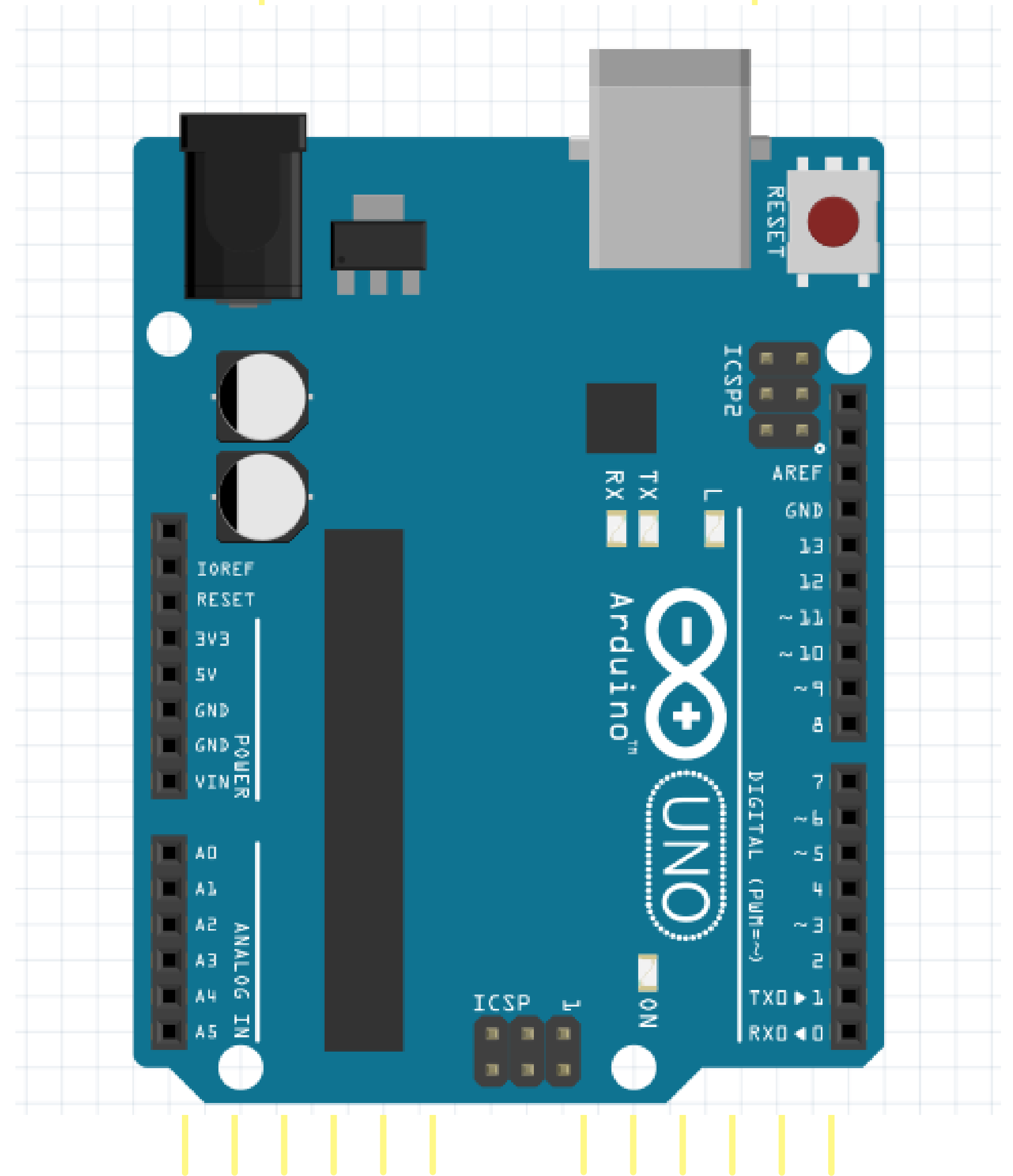
It has 4 main sections:

Power

Analog in (Read sensors)

Digital I/O (Input / Output)

PWM~ (Fake analog out)



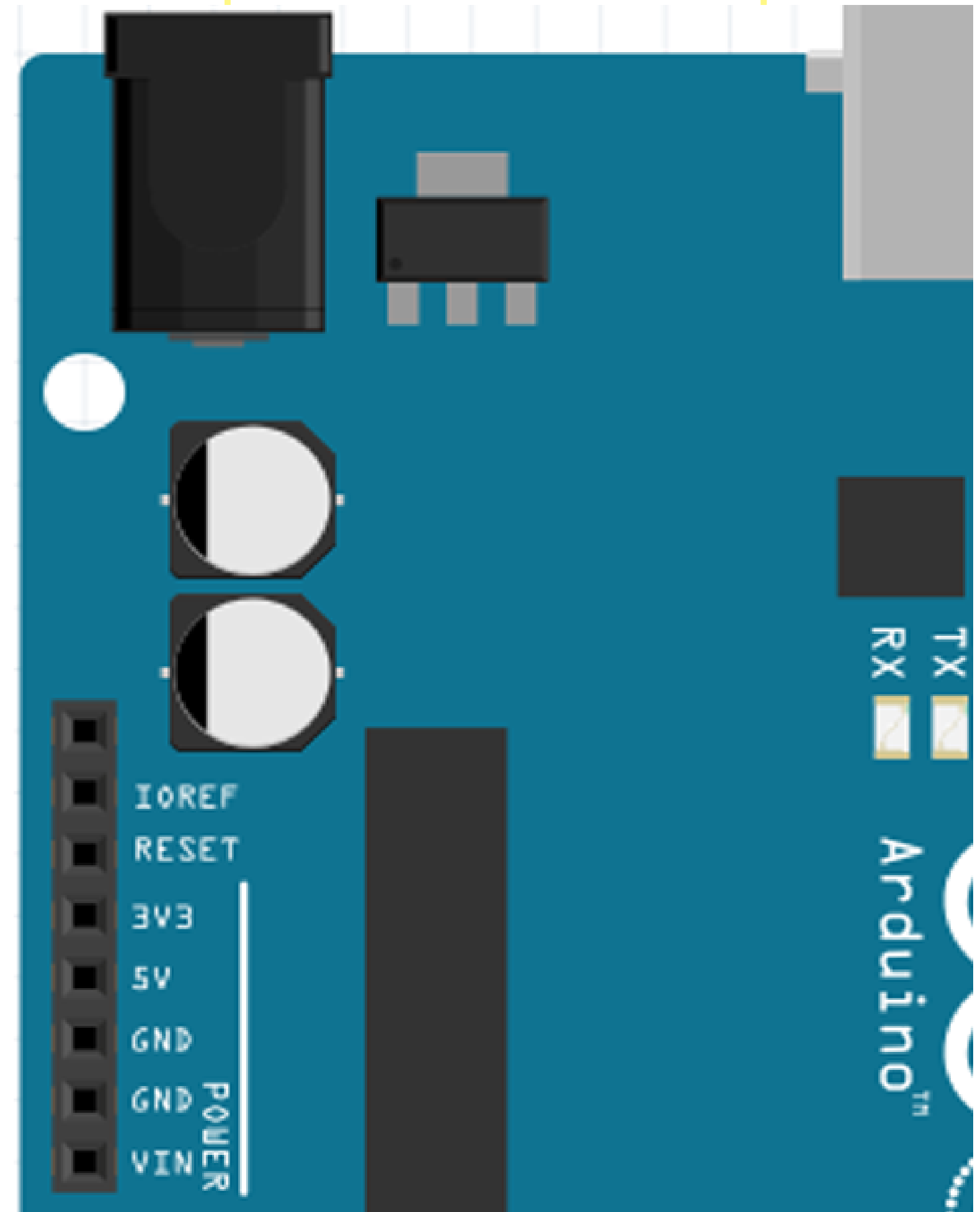
ARDUINO

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

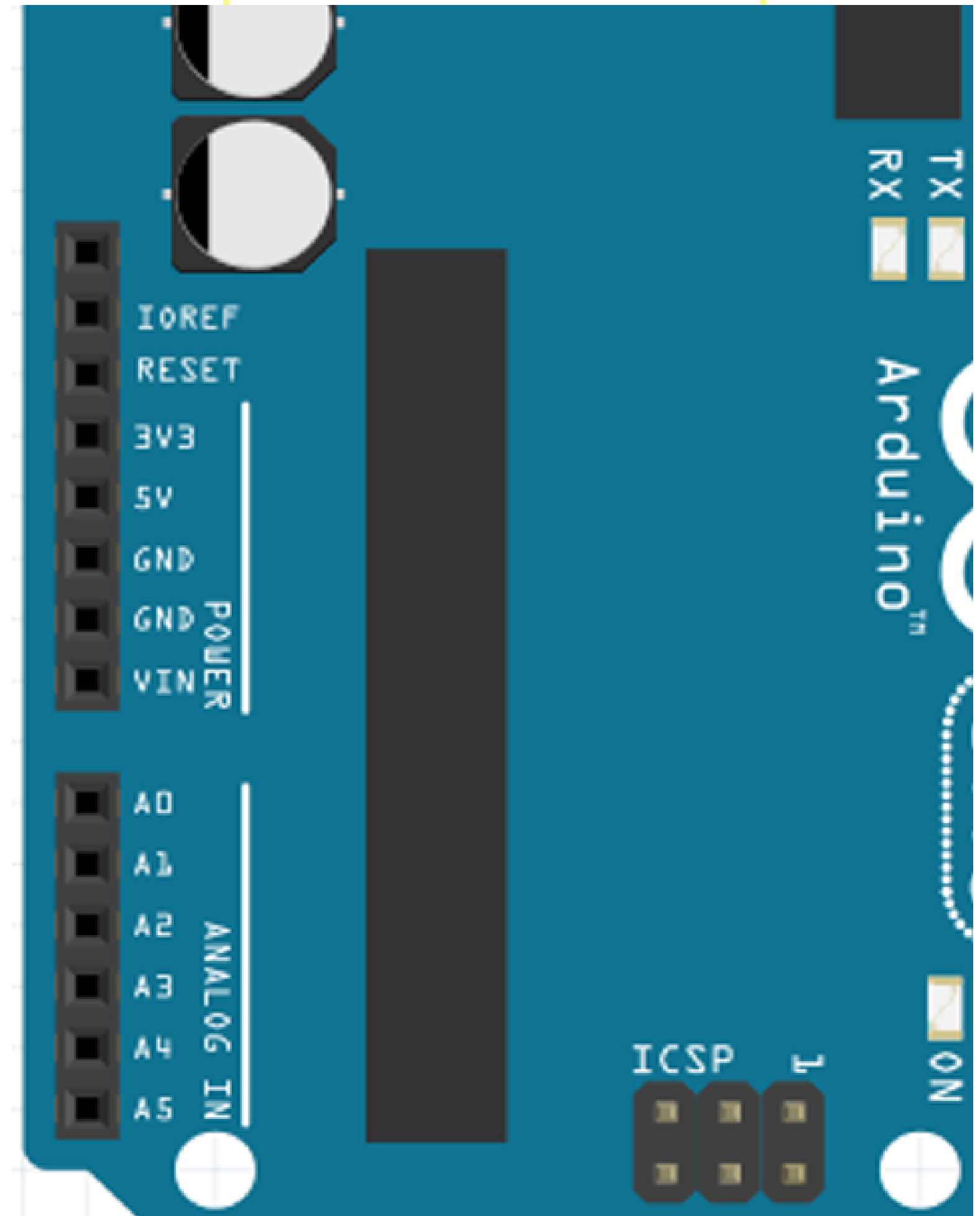
Analog In pins are located in the bottom left

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**)

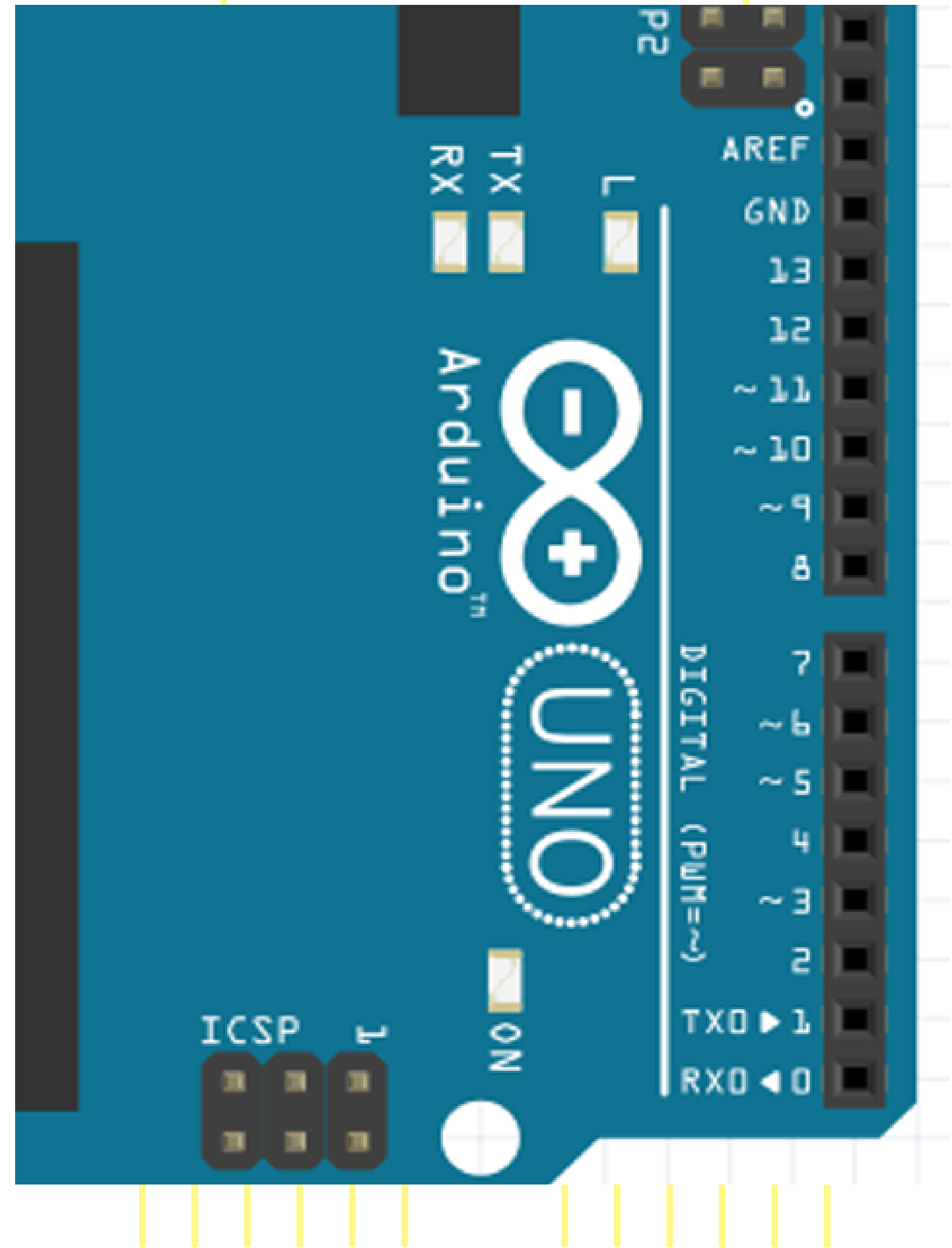


ARDUINO

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



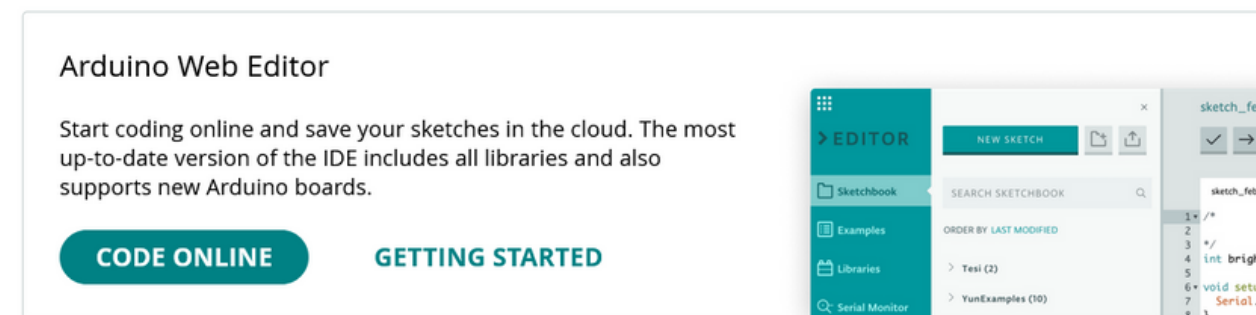
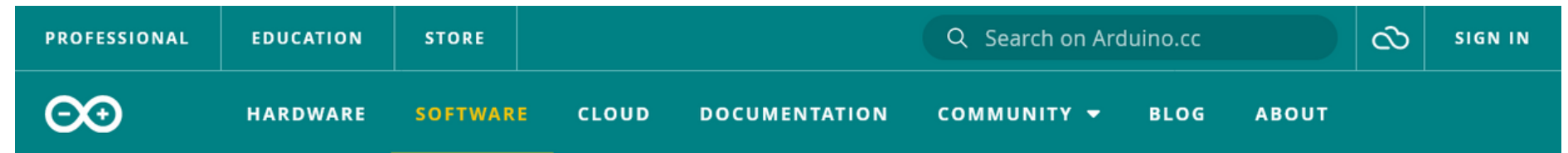
“Okay, but how do I use it??”

ARDUINO


First download the
Arduino IDE

This is the easiest way to
program the Arduino

You can get it from:
arduino.cc/en/software



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](#)

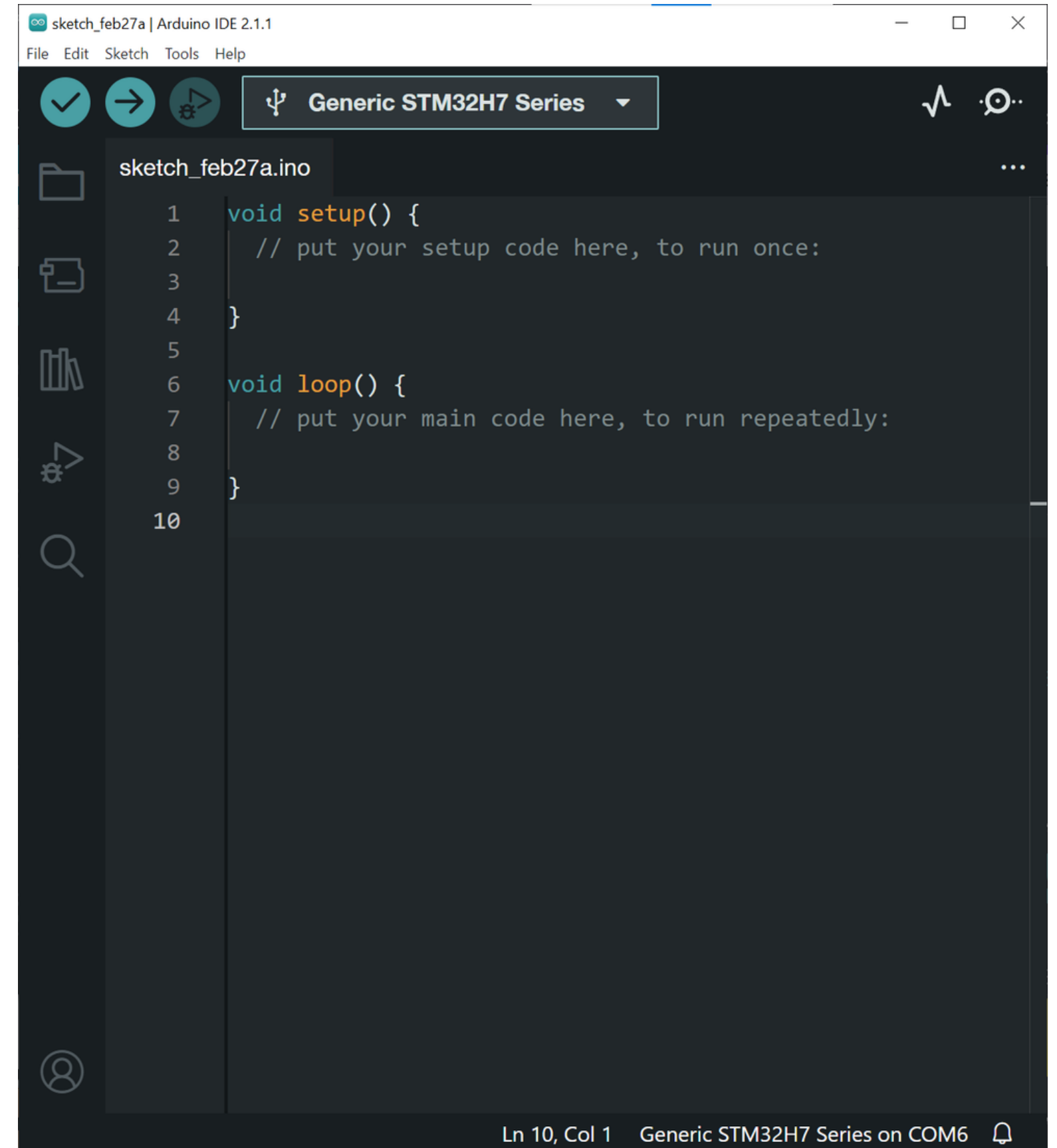
[Help](#)

ARDUINO

Open the
Arduino IDE

You will see something that
looks a lot like this.

Don't be scared!





**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?

Directions

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

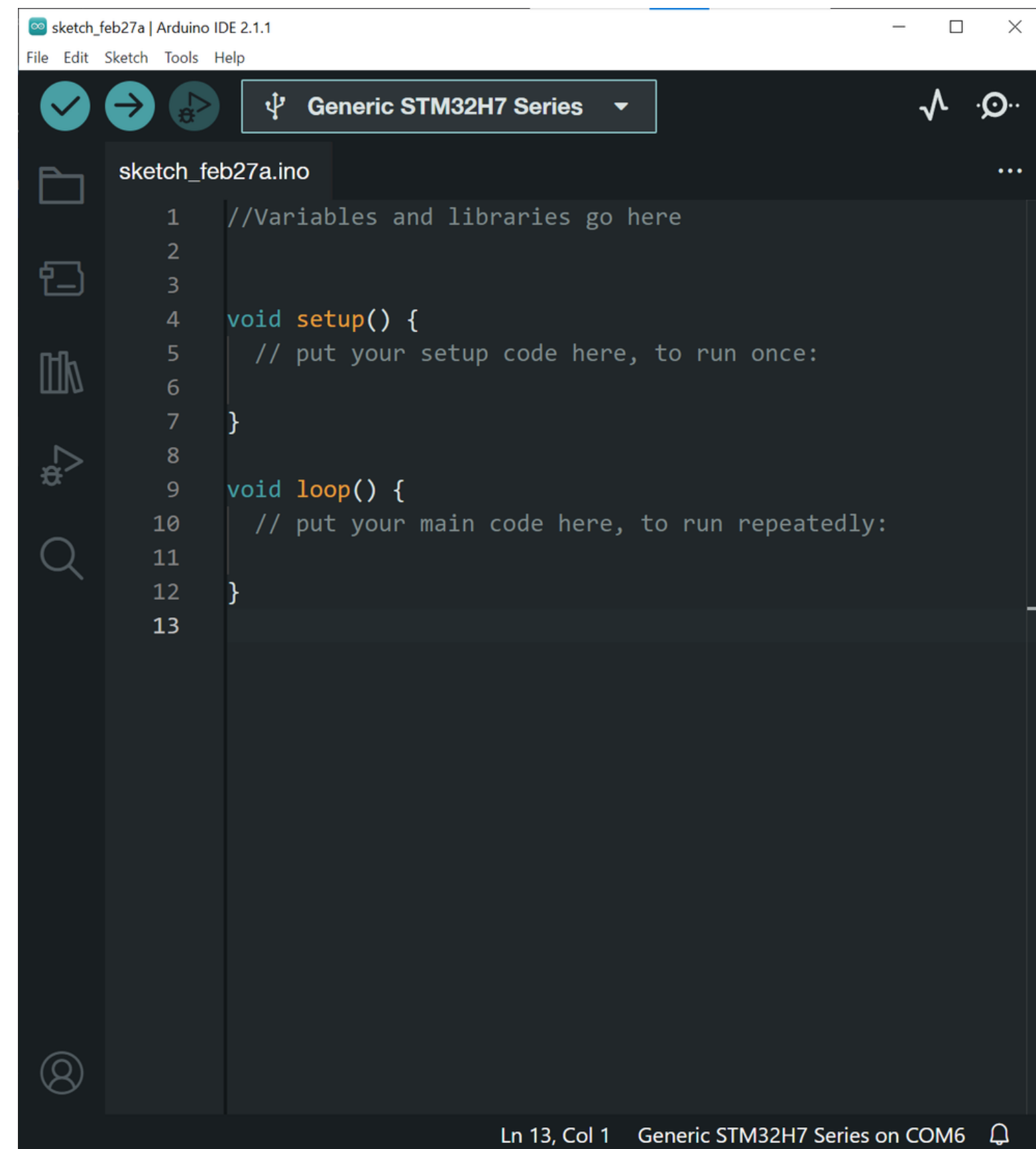
PROGRAMMING IS THE SAME

THINK OF IT LIKE A RECIPE

At the top we tell the program what ingredients we need. We call this declaring our **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.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
```

Ln 13, Col 1 Generic STM32H7 Series on COM6

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.

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



```
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
```

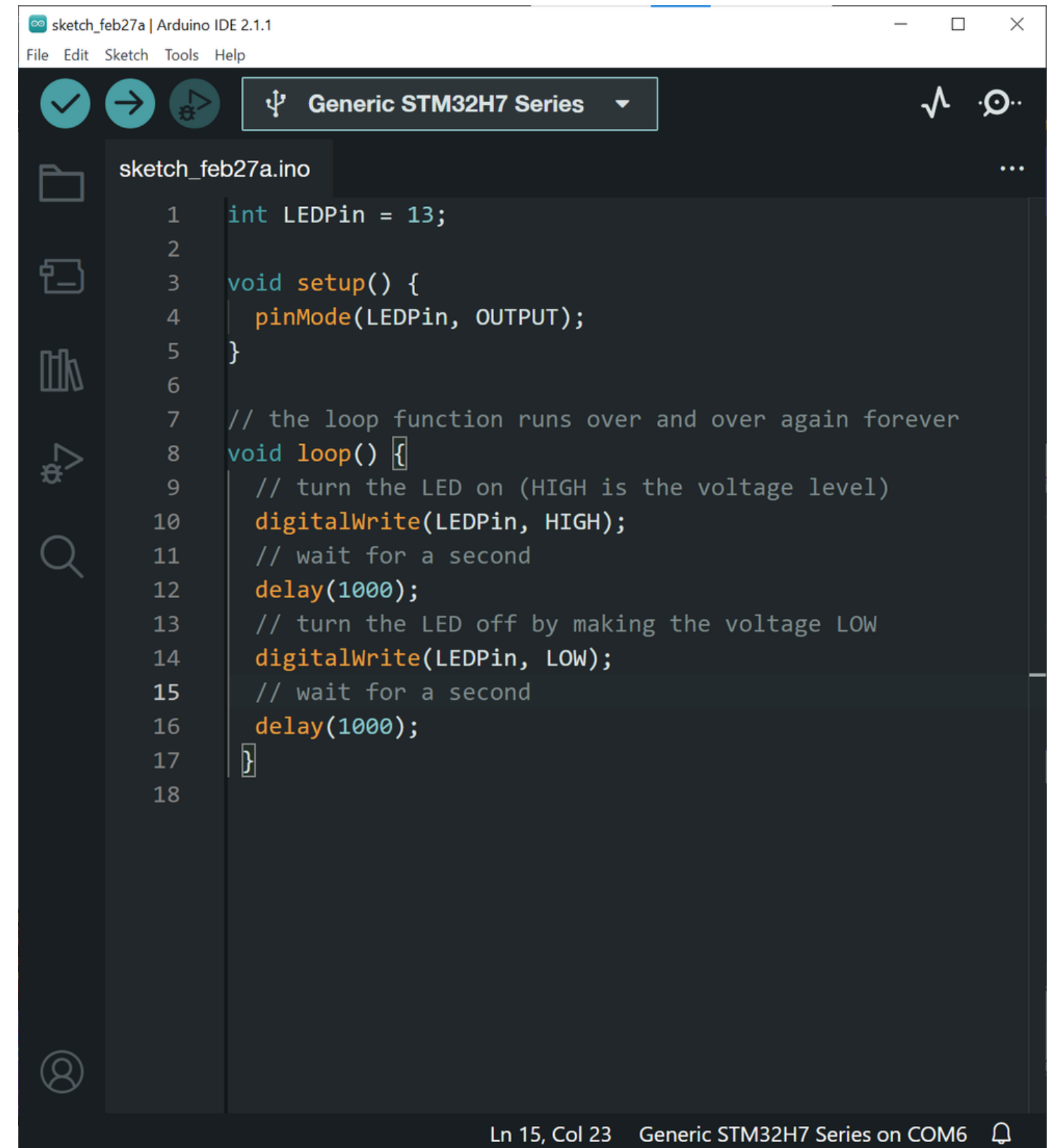
Ln 13, Col 1 Generic STM32H7 Series on COM6

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)



```
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
```

Ln 15, Col 23 Generic STM32H7 Series on COM6

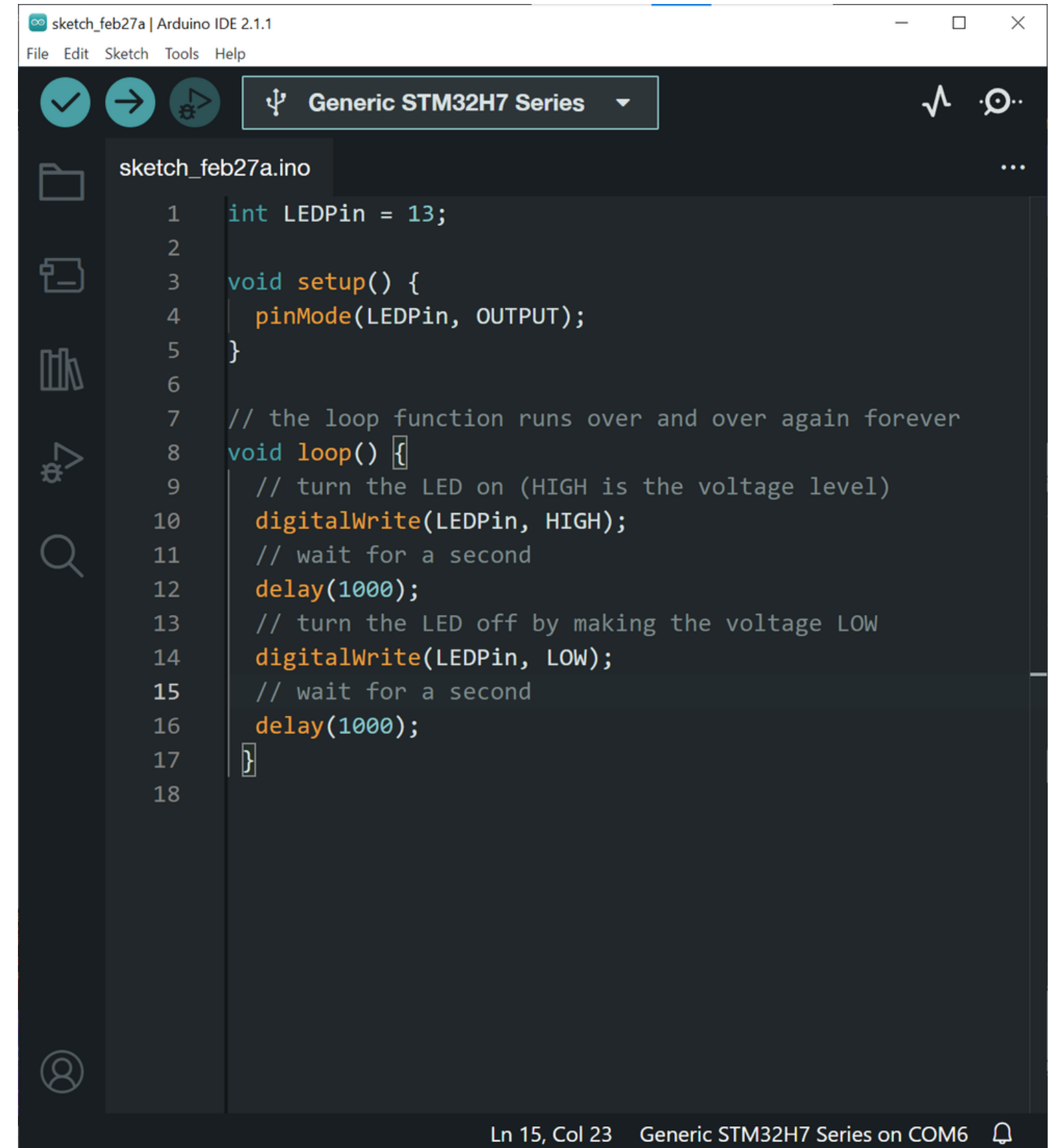
HERE'S AN EXAMPLE

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.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
```

Ln 15, Col 23 Generic STM32H7 Series on COM6

SOME MORE TALK ABOUT VARIABLES

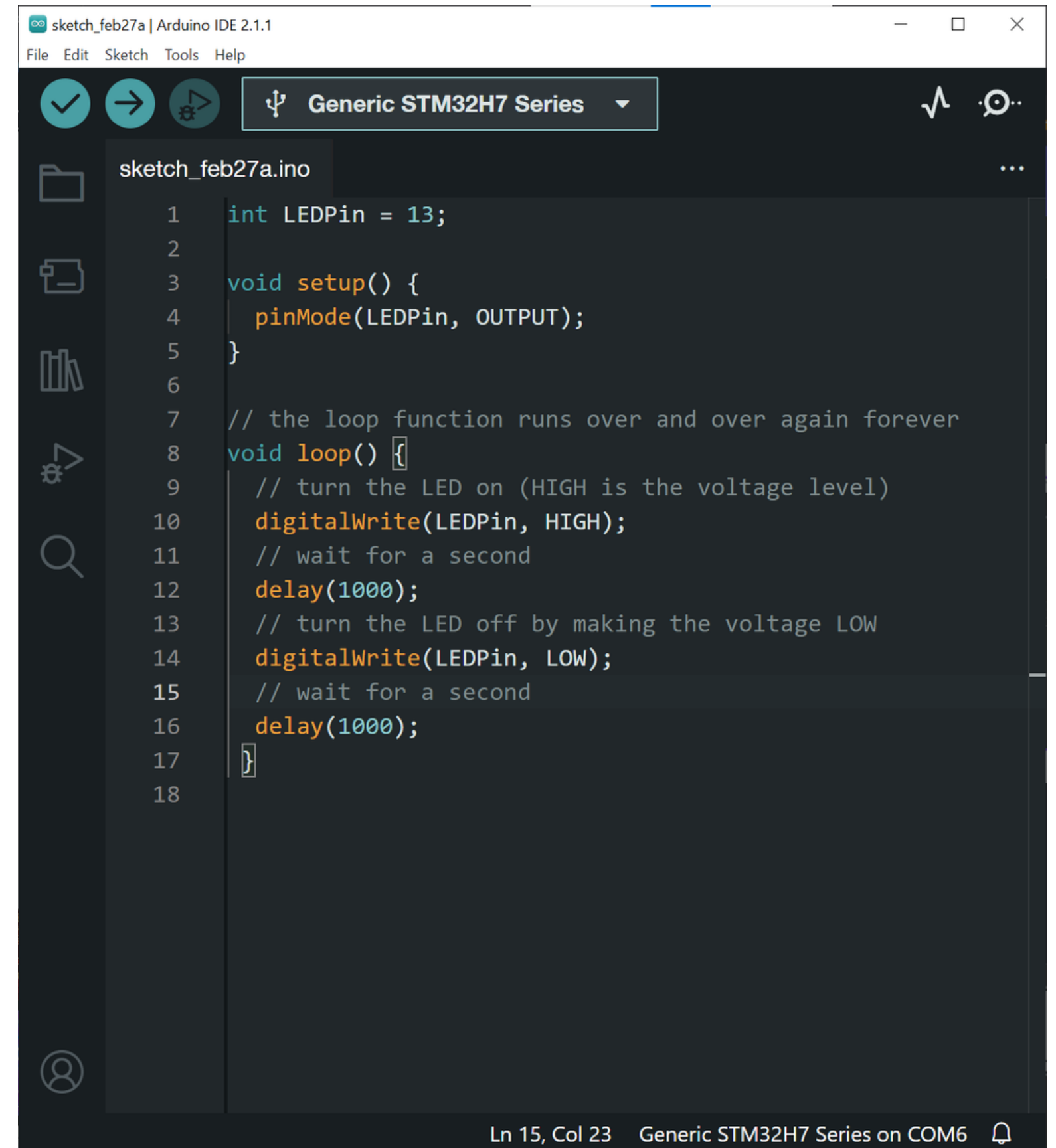
Variables tell the Arduino what to expect. Here are the two most common:

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



```
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

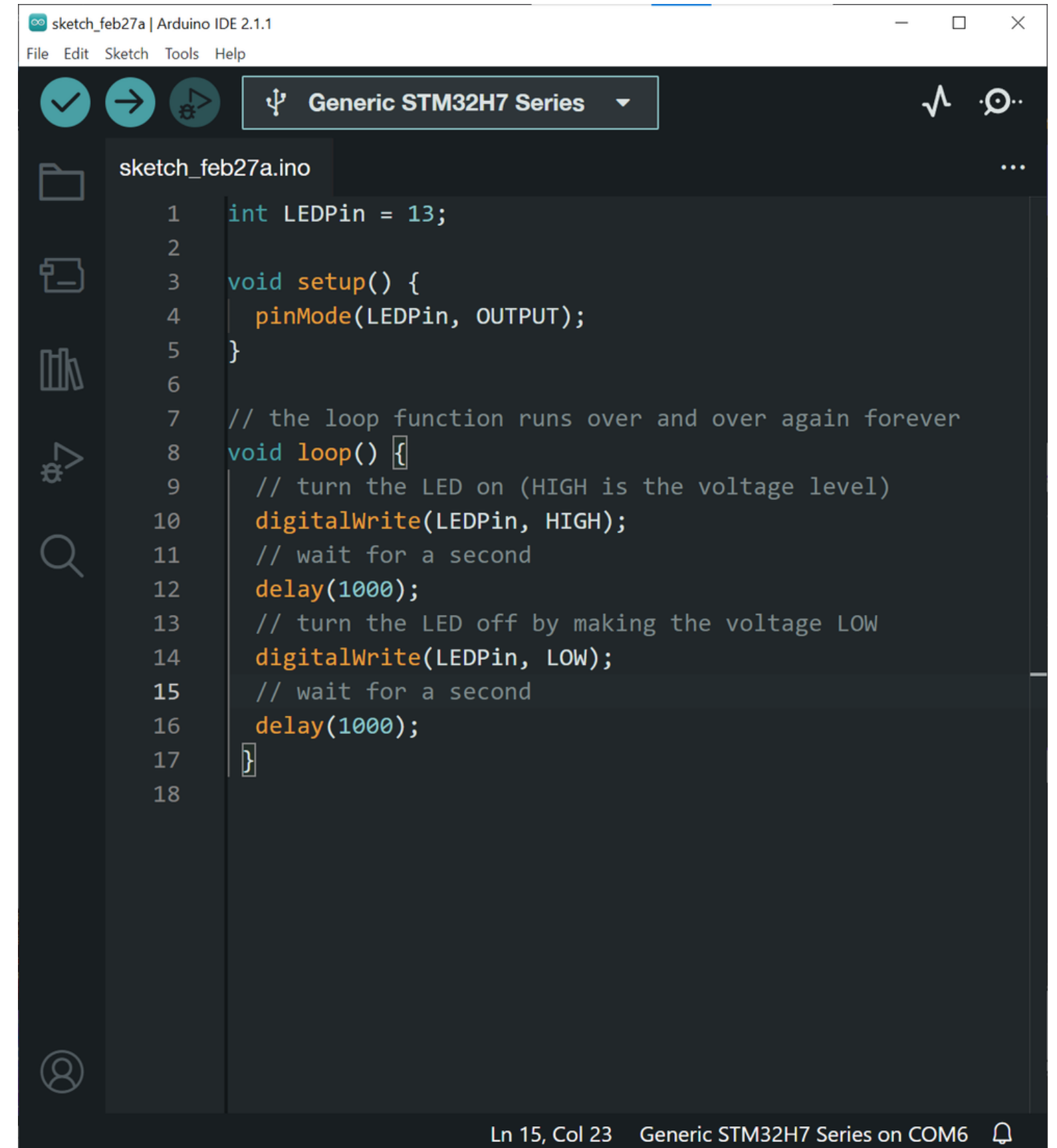
Ln 15, Col 23 Generic STM32H7 Series on COM6
```

LET'S TRY IT OUT!

Type in the code and get it to upload.

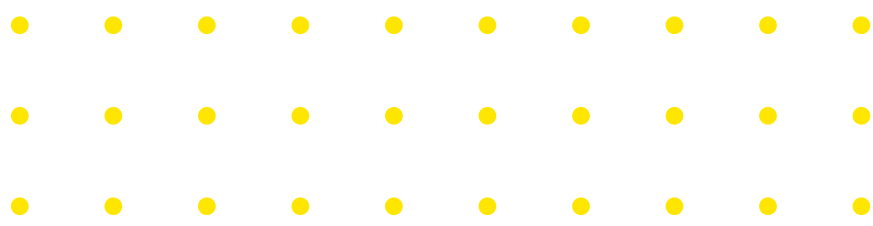
Try changing the delay times.

Try adding more lines of code to change the pattern.

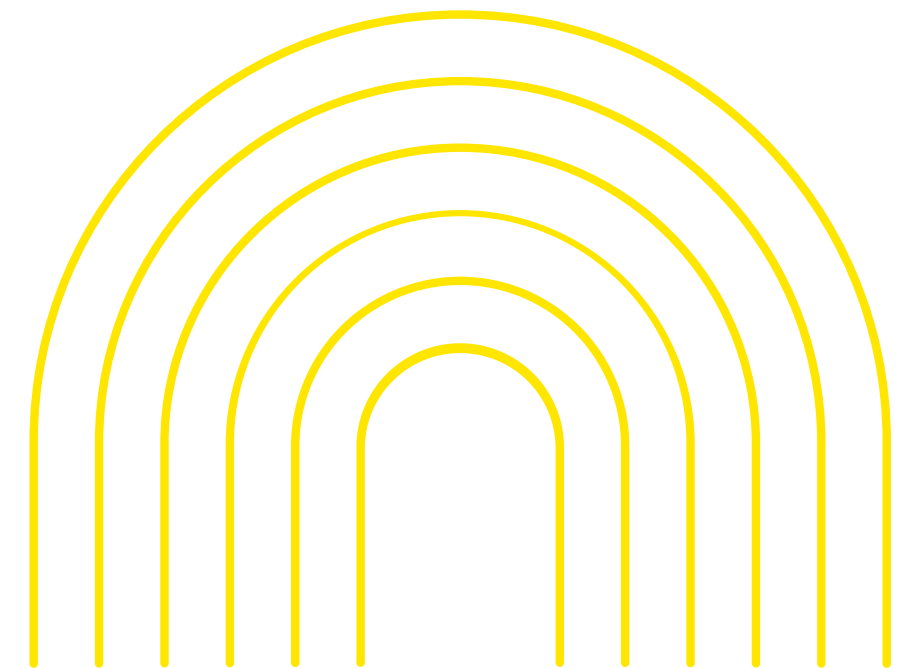


```
sketch_feb27a.ino
1  int LEDPin = 13;
2
3  void setup() {
4      pinMode(LEDPin, OUTPUT);
5  }
6
7  // the loop function runs over and over again forever
8  void loop() {
9      // turn the LED on (HIGH is the voltage level)
10     digitalWrite(LEDPin, HIGH);
11     // wait for a second
12     delay(1000);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

Ln 15, Col 23 Generic STM32H7 Series on COM6



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

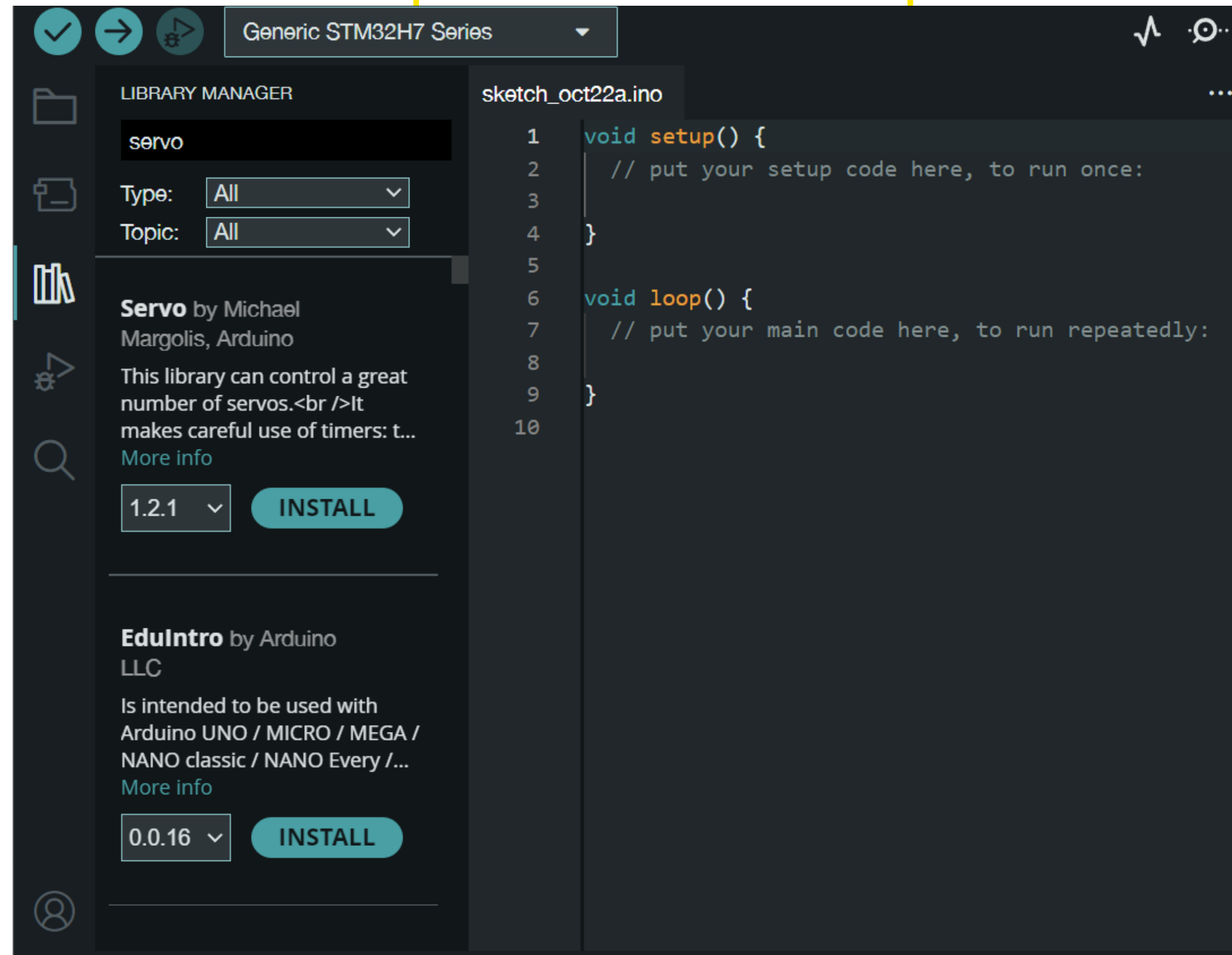


SERVO MOTORS

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.



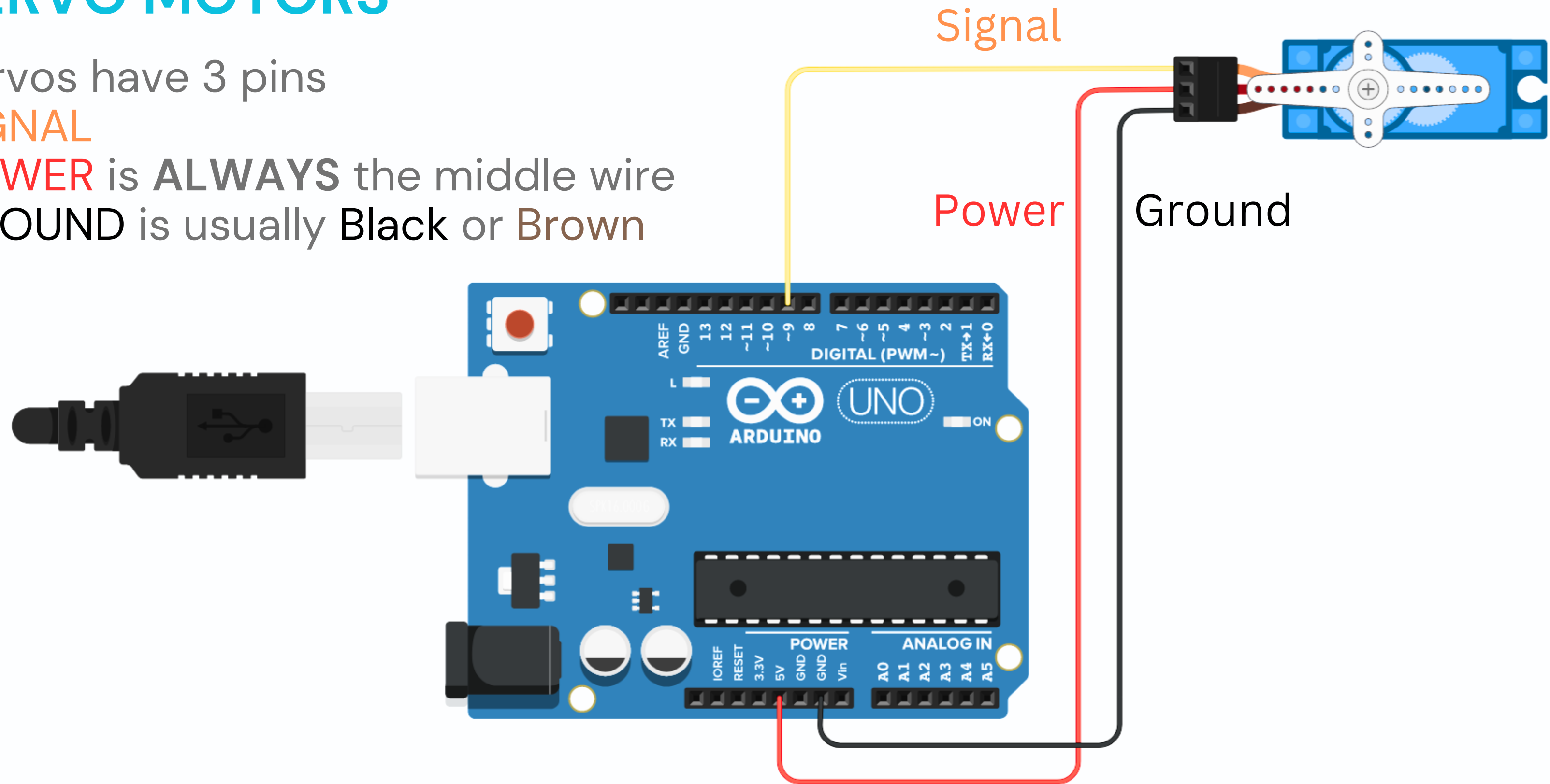
SERVO MOTORS

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

•

A_Servo.ino

```
1  #include <Servo.h>
2
3  Servo servo;
4
5  int pos = 0;
6
7  void setup() {
8      servo.attach(9);
9      servo.write(90);
10 }
11
12 void loop() {
13     servo.write(0);
14     delay(1000);
15     servo.write(180);
16     delay(1000);
17 }
18
```


SERVO MOTORS

Servo will:

Move to 0 degrees

Pause 1 second

Move to 180 degrees

Pause 1 second

Repeat

