2:0 HR

Ready, Set, Code

Initial setup and hands-on projects to bring your Alvik robot to life.

Get to Know Alvik

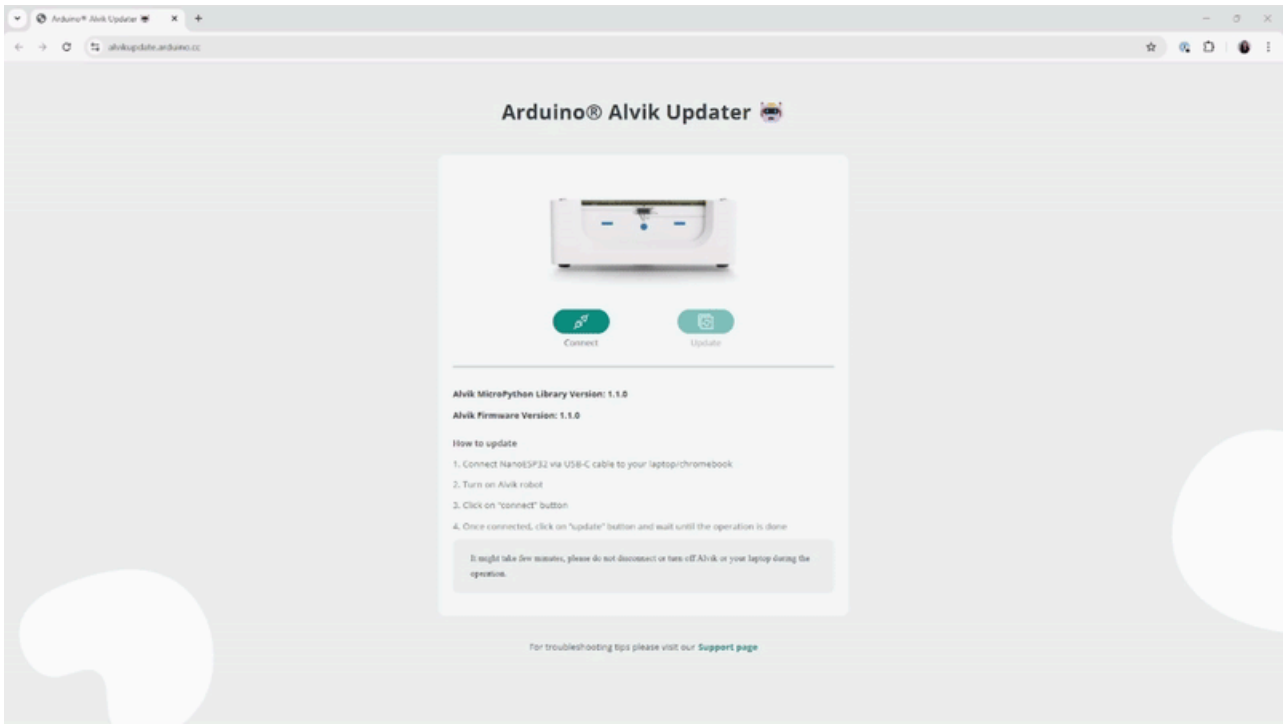
We are excited to introduce you to Alvik, a powerful robot companion at your command. As we explore the exciting world of robotics, you will learn how to communicate and interact with your robot while building real skills along the way. Before our adventure begins, let's learn a little more about the hardware.

Intelligent

Alvik is controlled by an **Arduino Nano ESP32**, a microcontroller that's small in size but big on capability. With integrated **Bluetooth® and Wi-Fi® connectivity**, your robot is not just a machine - it's a member of the **Internet of Things**.

Update Firmware with Alvik Updater

Updating the firmware on your Alvik ensures it has the latest features, bug fixes, and performance improvements. Regular updates help maintain compatibility with the last updates of the software and ensure optimal performance of your robot.

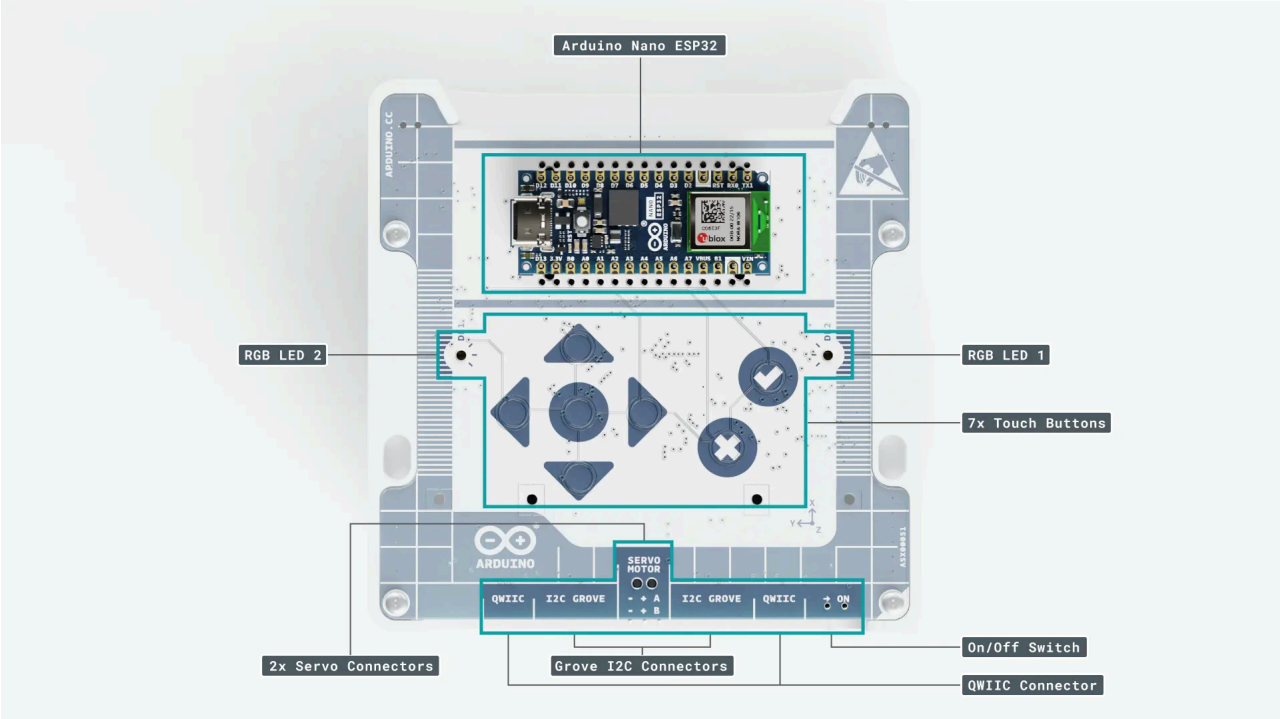


1. Visit https://alvikupdate.arduino.cc.
2. Connect Alvik to your computer.
3. Turn ON your Alvik.
4. Click the "Connect" button.
5. **A pop-up window will appear** prompting you to select the COM port. Choose the correct port and confirm.
6. Once connected, click the "Update" button and wait for the process to complete.

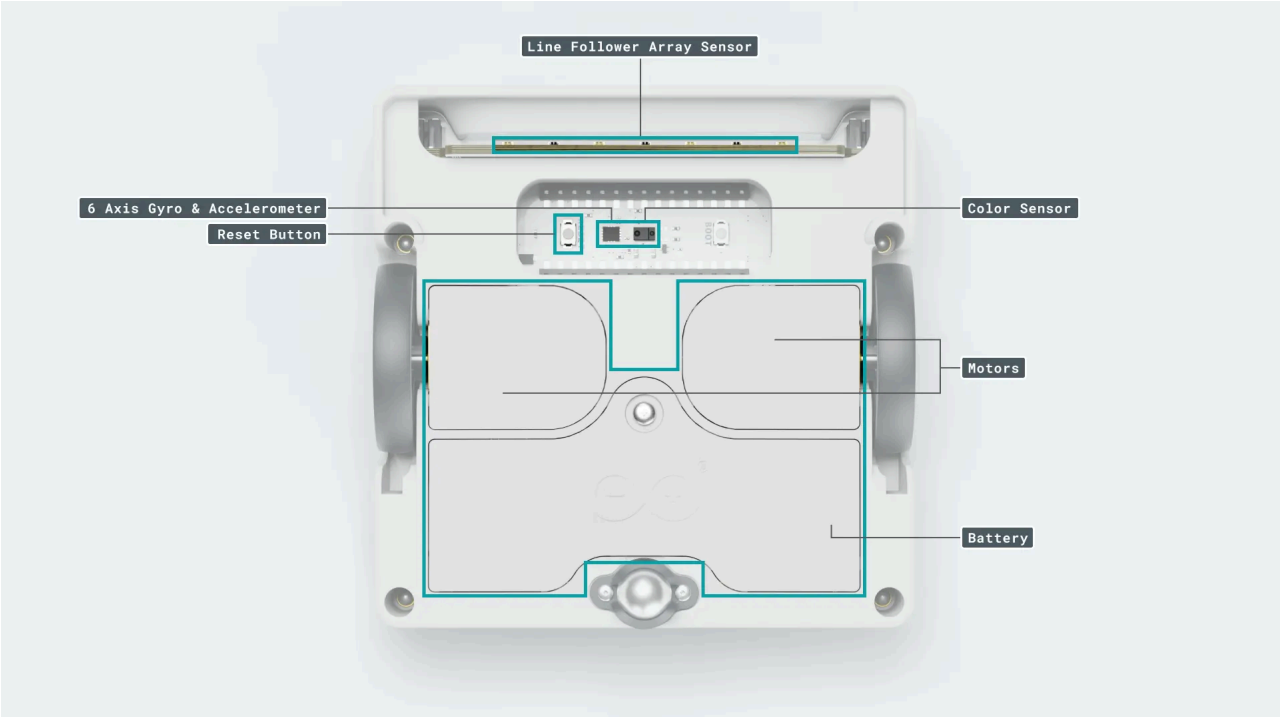
Energized

Your robot comes with a removable **USB-C® Rechargeable 18650 Li-ion Battery**, providing you with hours of cable-free engagement on a single charge. Every time you plug the USB cable to your device, Alvik starts recharging its battery, **so remember to keep it charged!**

Sensational



Equipped with a **broad range of sensors and connectors**, your robot is well-prepared to observe and engage with its surroundings. It features many inputs, such as **RGB Color and Distance Sensors**, providing you with a robust toolkit to explore and overcome challenges. As your skills evolve, you can upgrade the robot with **customized addons** or external hardware connected via **Qwiic, Grove I2C**, and **Servo Motor Ports**.

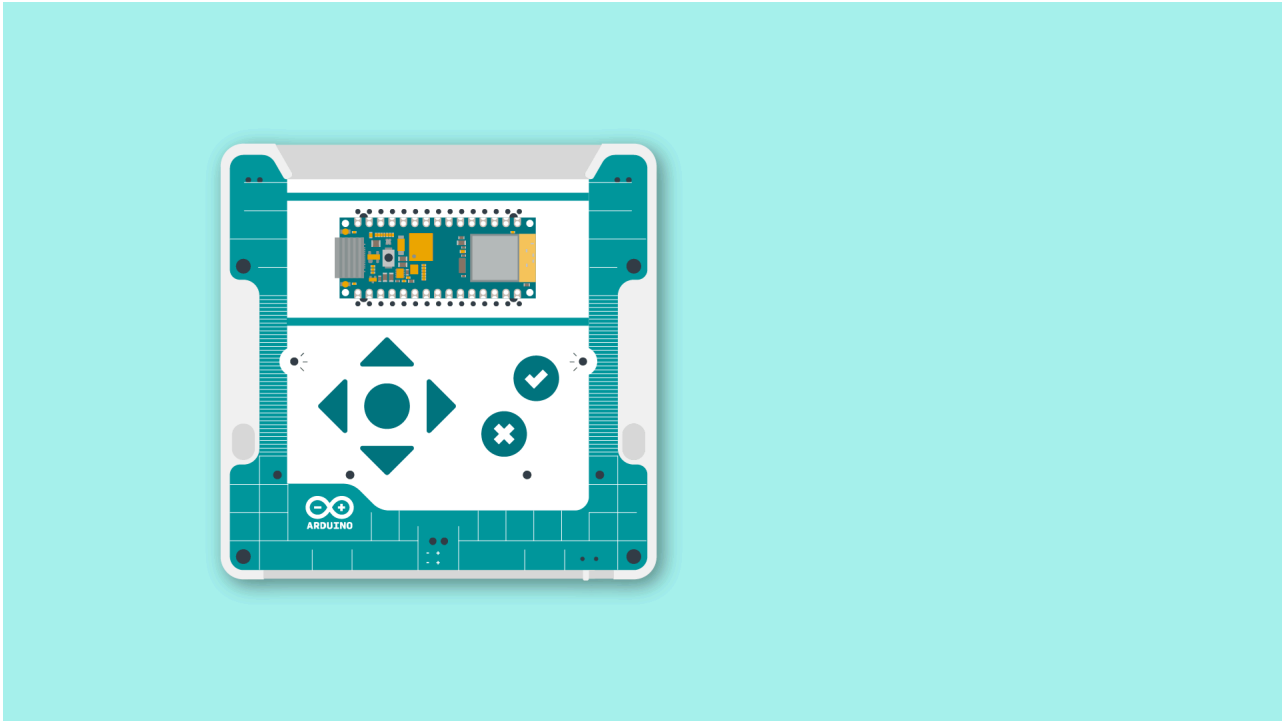


First Contact

Alright, it's time to get started! If you haven't done so already, unbox your Alvik.

Welcome to your new robot adventure! Alvik comes with three ready to go examples, each designed to showcase some of its amazing capabilities. These examples are color-coded Red, Green, and Blue. **Turn on your Alvik**, press the **up and down buttons to pick a color**, and hit the **checkmark button to start**. It's that easy!

Now, let's learn what these colorful programs can do:



Red Program (Touch Mode): Wanna be the boss of your robot? With Touch Mode, you can! Use the arrows to tell your robot what to do: up and down for moving forward and backward by 10 cm, and left and right for turning 90 degrees. The robot will remember the buttons you press and repeat the movements when you push the checkmark.

Green Program (Hand Follower): Get ready for some awesome follow-the-leader action! Your robot will keep a steady 10 cm distance from your hand or any object you put in front of it.

Blue Program (Line Follower): Ever seen a robot follow a line? Get ready for some magic! Your robot will glide along a black line on the floor like a pro. Just grab a white board and thick marker, or place black tape (2 cm wide) on a smooth floor to create the path.

> **NOTE:** If you want select a new program while another is running, restart Alvik by turning it off/on or pressing the white button on the Nano ESP32 board.

We hope you've had fun playing with Alvik! Once you feel familiar with your new robot, move on to the next sections where we learn how to connect and program it to do more cool stuff!

Learning Objectives

In this chapter, the goal is to set up your workspace and give simple commands to the robot. Key objectives include:

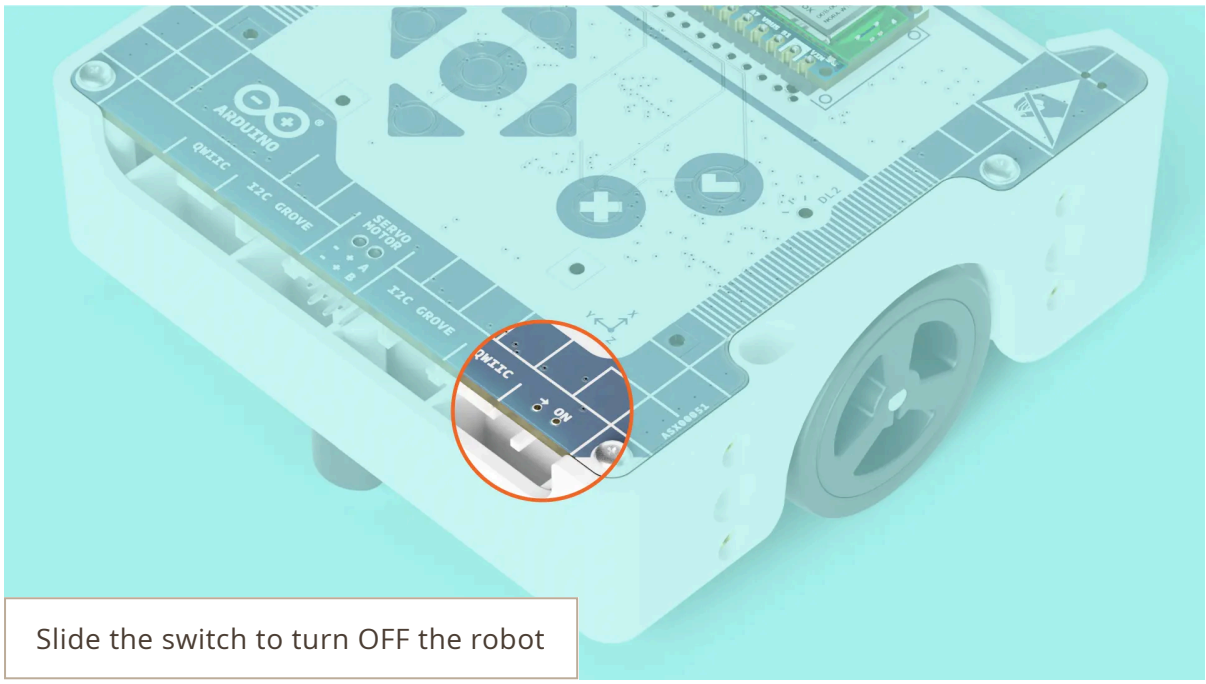
- Connect your robot to the editor, Arduino Lab for MicroPython.
- Navigate the editor and practice managing project files.
- Run scripts and input commands using the on-board terminal (REPL).
- Understand how to import key modules and initialize scripts.
- Experiment with functions to control Alvik's hardware.

Setting Up

Now that you're familiar with Alvik, let's connect it to your computer and install the code editor. Remove the **USB-C® Programming Cable** from the box and follow these steps to get started:

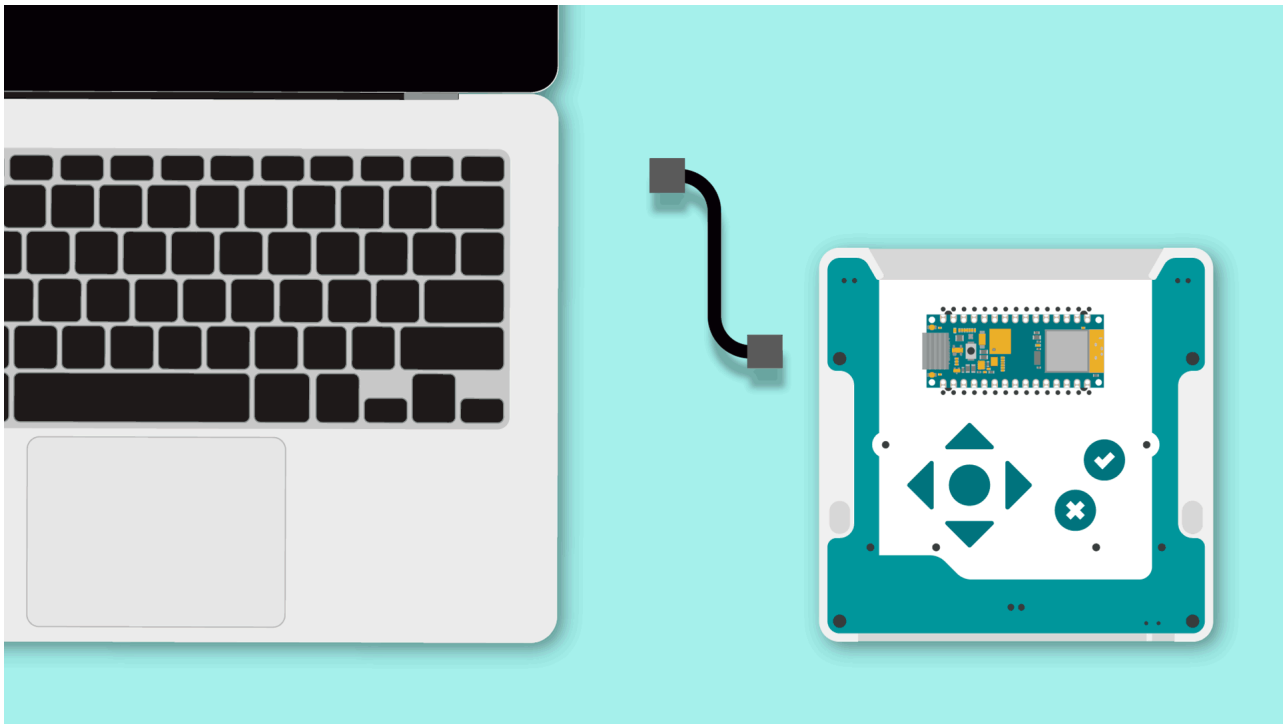
1. Locate the robot's **on/off switch** and slide it to the **OFF** position.

To connect your robot safely to the computer, it's important to **switch off the robot first**. If you try to connect it while it's still on, it may not show up on your computer! This isn't just about connectivity; it's also about taking care of your robot's battery and electronic bits. **So remember, before plugging in, power down!**



Slide the switch to turn OFF the robot

2. Connect the programming cable to your computer and the Nano ESP32.

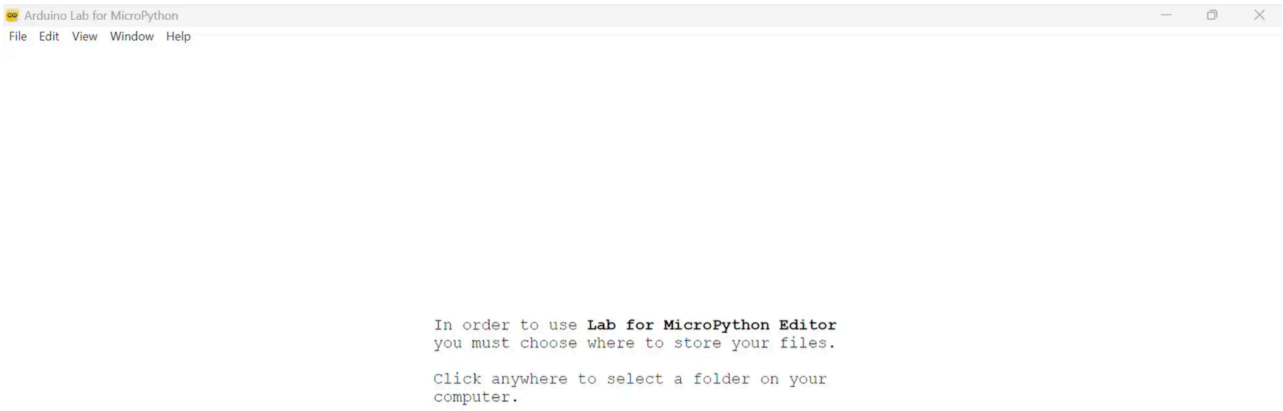


Arduino Lab for MicroPython

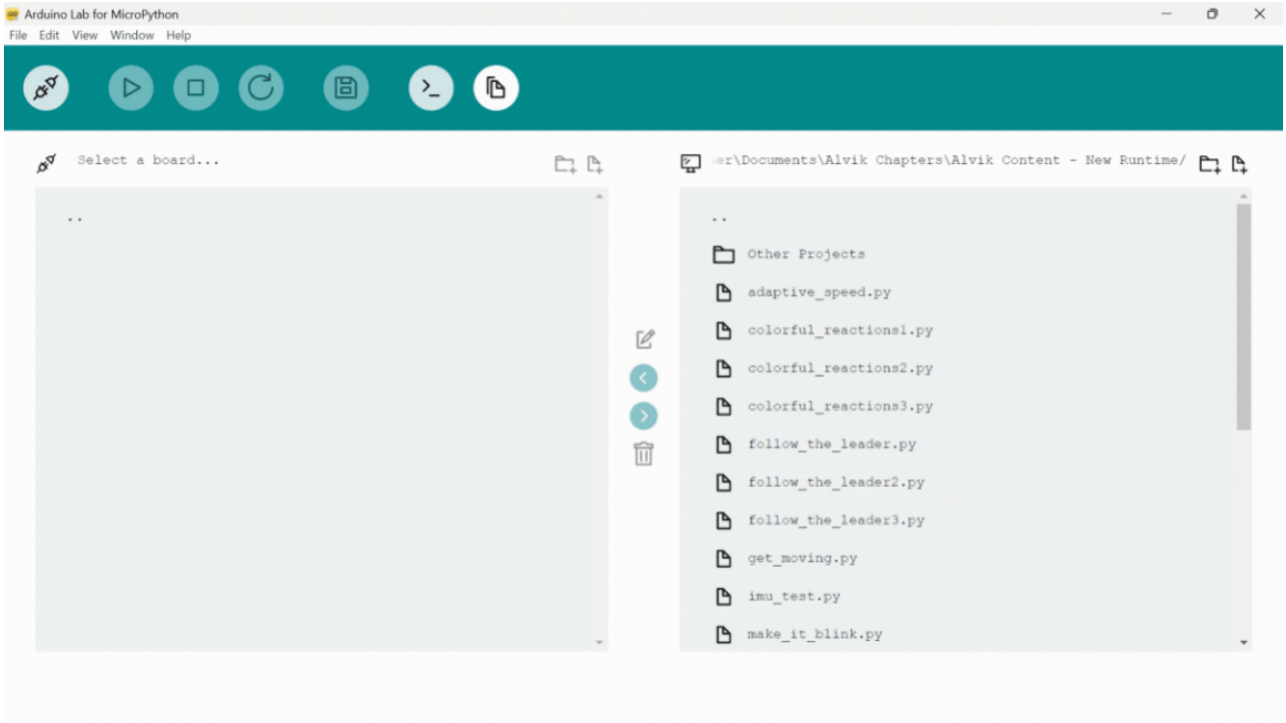
We are now ready to launch Arduino Lab for MicroPython, a light editor designed to write and run code in MicroPython on boards like Arduino Nano ESP32. If this is your first time using MicroPython, we recommend you check out our introductory course **MicroPython 101** to learn more.

NOTE: If not yet installed on your device, your teacher may guide you through the installation of Arduino Lab for MicroPython using the following link: **Arduino Lab Download Page** or use the **Online version** from a supported browser (Chrome, Edge, Opera). You'll need to login with an Arduino account to access it.

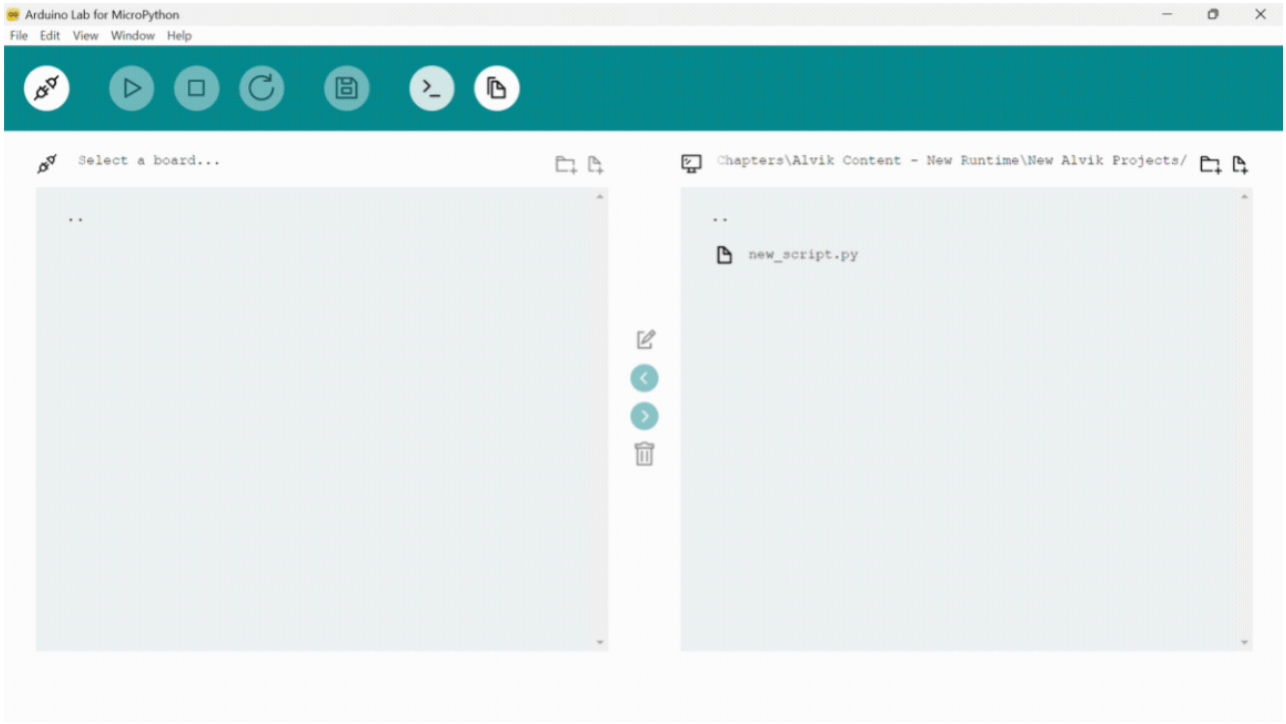
1. When you first launch Arduino Lab, you are asked to choose a folder for your files. **Create a Projects Folder on your device, then follow the editor's prompted instructions.**



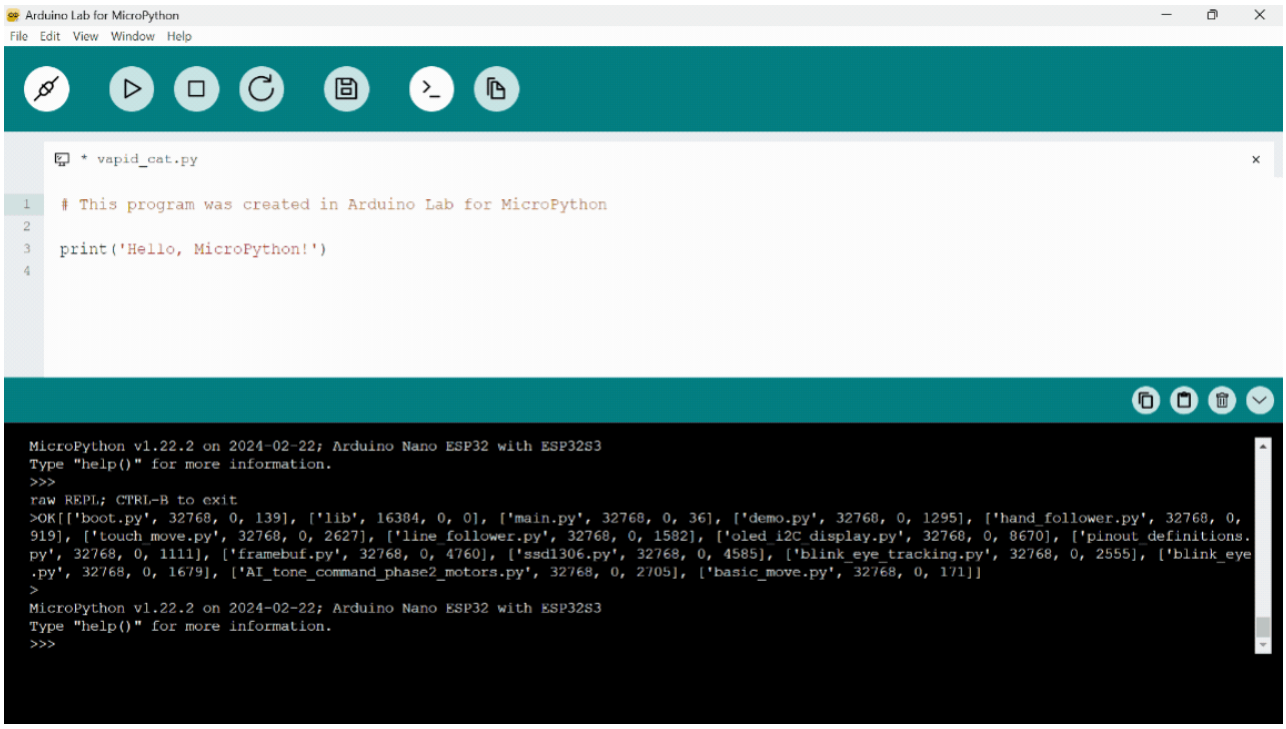
2. Click the File Manager button to access folder contents or change to a new folder.



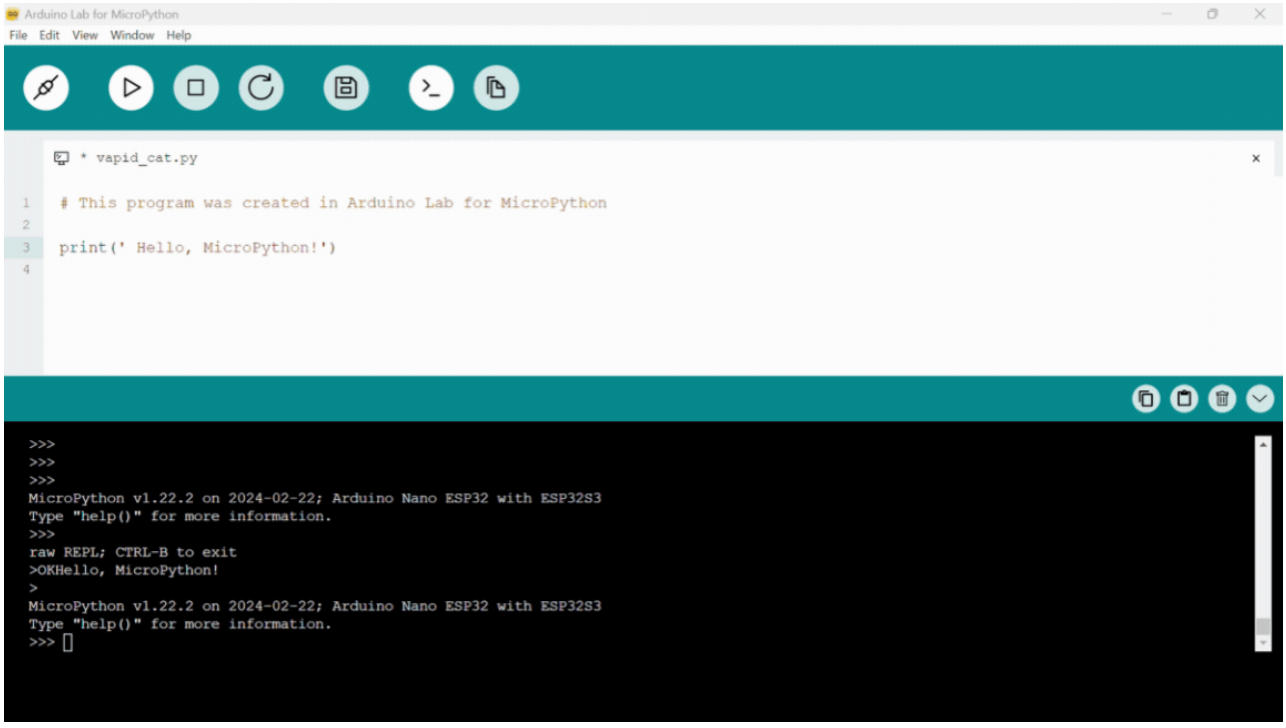
3. Click the Connect button and **choose the correct COM port**. In the File Manager, you should now see a list on the left that shows all your board's files.



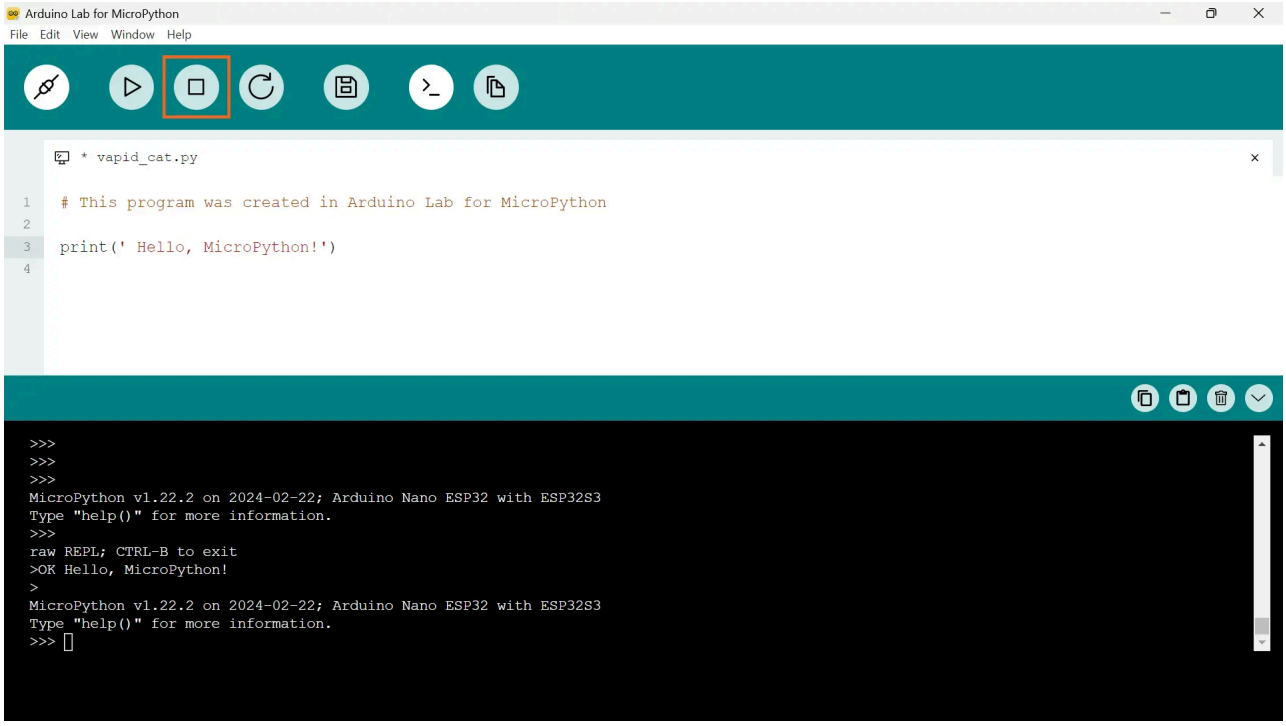
4. Click the Editor and REPL button. The top half is the editor where we write scripts and the lower half is the terminal.



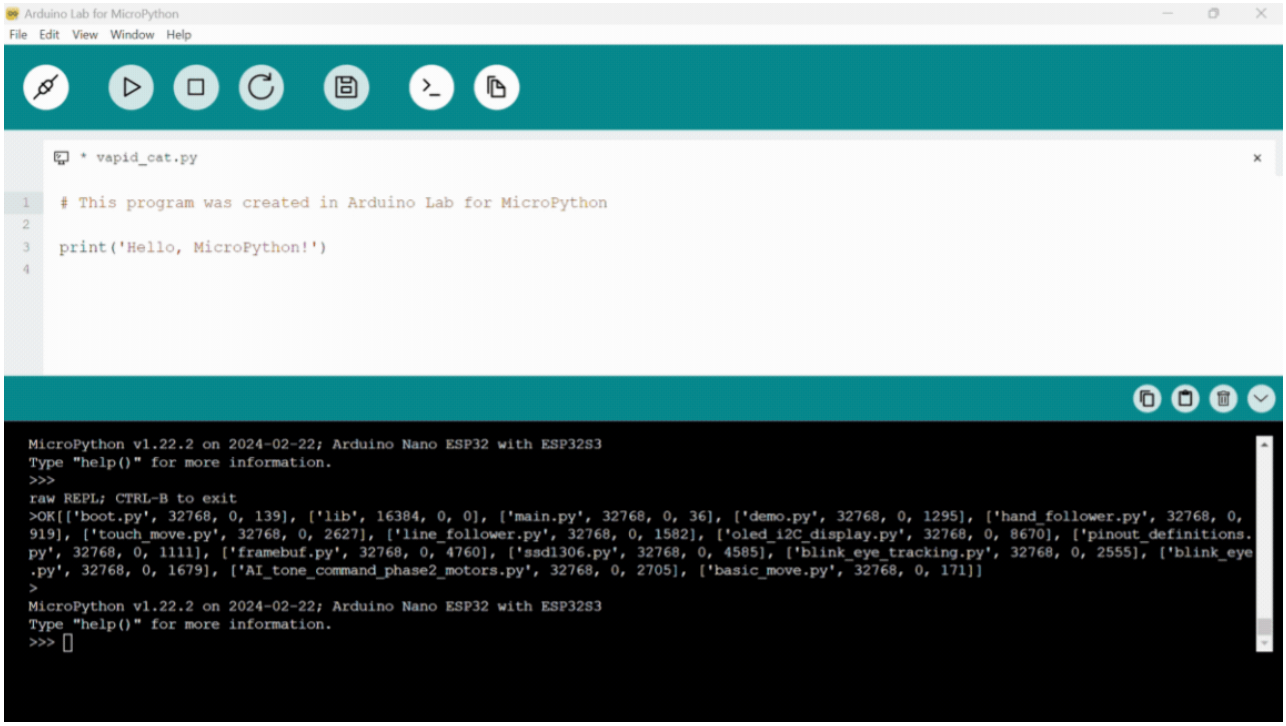
5. Try running the default sample script by clicking the Run button. You should see the text "Hello, MicroPython!" printed to the terminal.



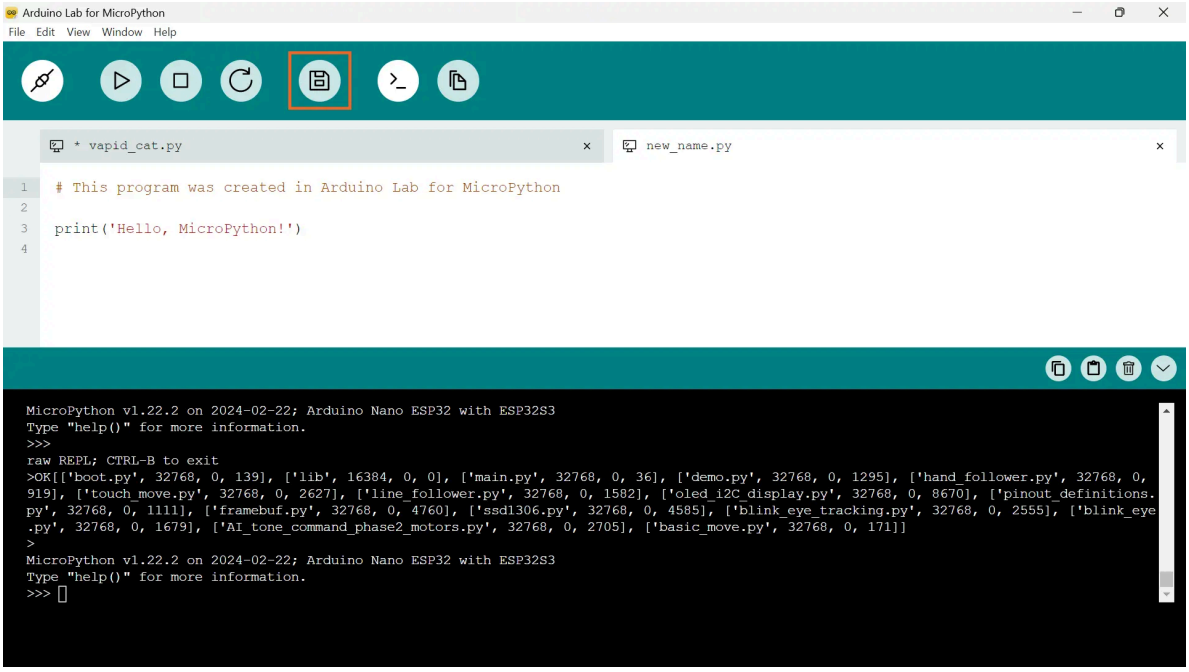
6. Click the Stop button to finish running your script.



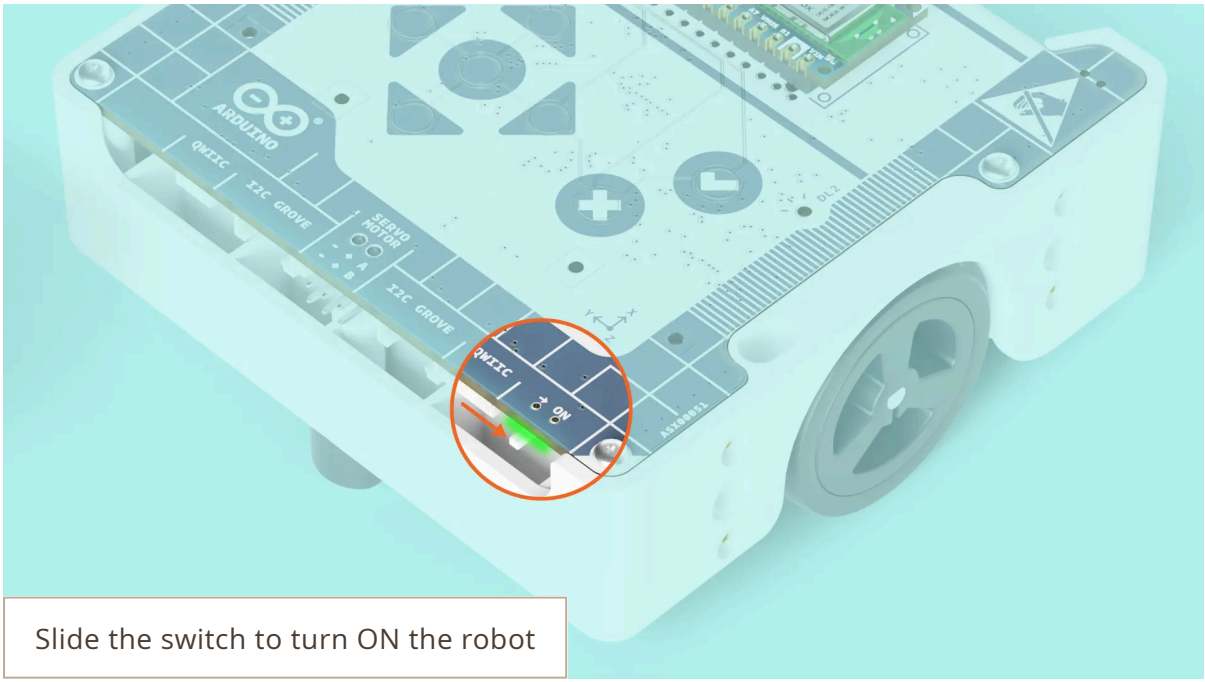
7. Open the File Manager and create a new file. Once named, **double-click on the file to open it in editor**.



`WARNING!` Remember to **save your changes often** by **clicking the Save button**.



Now slide the robot's **on/off switch** to the **ON** position.



Slide the switch to turn ON the robot

Woohoo, we are ready to start programming Alvik - great work! Stretch those coding fingers, do a few pushups, and let's go!

All on Board

MicroPython is a version of Python designed for microcontrollers, making it easy and intuitive to use. With MicroPython, you can write, test, and save code directly on devices like the Arduino Nano ESP32 using an interactive REPL (Read-Eval-Print Loop), all happening on board. This allows you to control hardware like LEDs and sensors quickly, without having to constantly recompile and upload your code. Essentially, it's the classic Python experience we love, all packed onto a tiny device. How awesome is that!

Let's do a quick warm-up example to show you what we are talking about. Check out the code below:

```
from time import sleep

print("Hello! I'm a talkative robot. What's your name?")
student_name = input("Your name: ")
print("Great meeting you, " + student_name + "! Would you like to name me?"
robot_name = input("Robot's name: ")

print(f"{robot_name} is a fantastic name! I feel more human already.")

sleep(2) # Use sleep() to make interaction feel more natural
print(f"Okay, {student_name}, time for a quick laugh:")
sleep(2)
print("Have you heard of the robot that tried to swim?")
sleep(4)
print("It shocked everyone. :D")
sleep(5)
```

Copy this code into the last practice script or create a totally new file. You can change the names of these files anytime.

UPDATE FILE NAMES

NOTE: Remember to save changes often throughout your coding journey!

When ready, **click the Run button**. You should see some activity inside the REPL. Go ahead, **click on the last line in the terminal to input a response.**

Now that you have completed your first coding interaction, let's break everything down to make sure we understand what's happening.

Bring Your Toolbox

Python is loaded with incredibly useful resources like the `time` module. Think of **modules** as toolboxes that are filled with specialized tools. Whenever you need to complete a specific task, you simply grab the tool you need to get the job done.

```
1   from time import sleep
```

In coding one of our most powerful tool types is called a **function**, or in other words, a specially coded set of instructions that complete a very specific task. All you need to do is insert a function in your code and it will do its job - for example, we are using `sleep()` to make our program wait a few seconds before moving to the next line of code. Simply import `sleep` from the `time` module, and you can use it anywhere in your script.

Robot Greeting

```
1   print("Hello! I'm a talkative robot. What's your name?")
2   student_name = input("Your name: ")
```

Notice how we use `print()` to display text on the terminal. We will use a lot of different data types throughout this course, but if you see data inside single or double quotation marks, we are talking about a **string** of text.

We also have a special function called `input()` which prompts the user to enter information directly into the terminal. Whatever information the user provides will be stored in the variable `student_name`. Keep in mind that the program will wait for an input and not continue until one is received.

Concate-what?

In our next print statement, we are combining strings and variables to make the interaction more dynamic. Such a combination of data is called **concatenation**.

```
print("Great meeting you, " + student_name + "! Would you like to name me?"
```

Pay attention to how we use the `+` symbol to combine our strings with the `student_name` variable.

- **Challenge:** Take a look at the full-code. Can you identify an alternative way to combine variables with a string? Do you think one way is better than the other?

Interaction Pacing

```
print(f"{robot_name} is a fantastic name! I feel more human already.")
sleep(2) # Using sleep to make interaction feel more natural
print(f"Okay, {student_name}, time for a quick laugh:")
sleep(2)
print("Have you heard of the robot that tried to swim?")
sleep(4)
print("It shocked everyone. :D")
sleep(5)
```

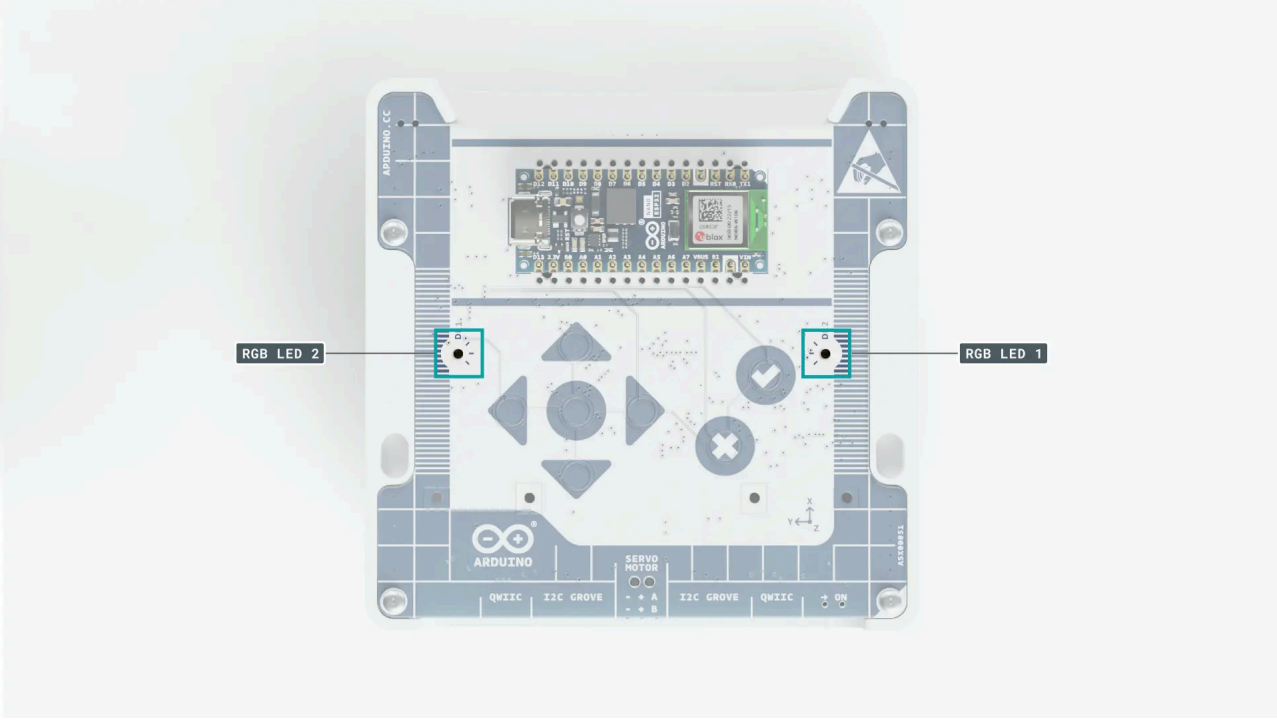
If you need to leave **comments** in your code, use the `#` at the start of your comment. Comments should be as short as possible and explain key information to help the reader understand the code. Explanations are **not what the code does, but generally how or why we do something**.

As you can see above, we have a comment explaining why we are using the `sleep()` function to pace the interaction. This creates a more natural conversational flow, and in the case of `sleep()` positioned before the final `print()` function, an extra-long delay of four seconds is used to improve comedic timing when delivering the punchline.

- **Challenge:** What happens if we shorten or remove all the `sleep()` functions? Does the conversation feel the same?

Make It Blink

Alvik comes with two built-in **RGB LEDs (Red-Green-Blue Light Emitting Diodes)** which can generate a wide range of colors by adjusting the combined intensity of red, green, and blue light emitted. These are located on top of the robot. In our next activity, we will program the lights to blink and change colors.



Let's start with a simple blink program. **Create a new file called *make_it_blink.py* and copy the code below into your script**.

```
from arduino import *
from arduino_alvik import ArduinoAlvik

```

```
alvik = ArduinoAlvik()

def setup():
  alvik.begin()
  delay(1000)

def loop():
  alvik.left_led.set_color(1, 0, 0)
  alvik.right_led.set_color(1, 0, 0)
  delay(1000)
  alvik.left_led.set_color(0, 0, 0)
  alvik.right_led.set_color(0, 0, 0)
  delay(1000)

def cleanup():
  alvik.stop()
```

Arduino Runtime for MicroPython

Before we go much further, this is a good time to talk about the **Arduino Runtime for MicroPython**, a special system that resembles the traditional Arduino coding experience with all the benefits of MicroPython.

RUNTIME TROUBLESHOOTING

Let's do a quick breakdown before taking the blink program to the next level, starting with **script initialization:**

```
from arduino import *
from arduino_alvik import ArduinoAlvik

alvik = ArduinoAlvik()

def setup():
  alvik.begin()
  delay(1000)
```

We start by importing two modules with important functions and classes for later use in our program. The first module `arduino` contains all the magic that makes the runtime work. Then we have `arduino_alvik` which provides access to hardware-specific features of the Alvik robot. We need to create an `alvik` **object** from the `ArduinoAlvik` **class** to control all the hardware. We will learn more

about classes and objects in the future, so don't get hung up on the terminology at this point.

Everything that comes after starts to feel a bit like "classic Arduino" with a special function `def setup()` where we include any **initialization code that only runs once** and `def loop()` **for all the code that repeats forever**.

Note that MicroPython normally uses `sleep()` or `sleep_ms()` from the `time` module to control delays, but this runtime lets you use `delay()` with milliseconds just like Arduino C.

At last, we have `def cleanup()` **to decide what happens when the Stop button is clicked in Arduino Lab.** The method `alvik.stop()` is used in this case to stop all hardware when you end the script. And to wrap up, all these functions are passed as arguments to the `start()` function which executes them.

> **NOTE:** Keep a careful eye on spacing, as it may affect how your code executes. Remember that Python and MicroPython use indentation to represent code blocks, or in other words, chunks of code that run together.

Blinking Light

Okay, let's get back to RGB lights. Pay attention how we control the **state** of the LEDs, or in other words, if they are on or off.

```
1   alvik.left_led.set_color(1, 0, 0)
2   alvik.right_led.set_color(1, 0, 0)
3   delay(1000)
4   alvik.left_led.set_color(0, 0, 0)
5   alvik.right_led.set_color(0, 0, 0)
6   delay(1000)
```

Observe the first line of code `alvik.left_led.set_color(1, 0, 0)` . Earlier in our script, we created an **object** called `alvik` using `alvik = ArduinoAlvik()` , which gives us access to all the **methods** inside of the `ArduinoAlvik` **class**. Think of a class as an empty template that contains all the

functions your robot can perform, and any time we use these functions, we call them **methods**. `left_led.set_color()` is a class method we use to control the values of the left RGB LED. This method belongs to the `alvik` instance of the class, signified by a `.` that connects methods to the object.

Let's talk about the values, or **arguments**, inside the method `left_led.set_color()`. Notice that the RGB LED has three values that can be turned on (1) or off (0). The colors Red, Green, and Blue each have matching placeholders inside of the method, following the order of R, G, B. In other words, `(1,0,0)` translates to (red ON, green OFF, blue OFF). If you wanted the color blue, you would change the values to `(0,0,1)` which is the same as saying (red OFF, green OFF, blue ON).

Looking at the complete set of code, we should now understand that `left_led.set_color(1,0,0)` **turns on the left LED color red** and `left_led.set_color(0,0,0)` **turns it off**. The same goes for the right LED as you can see in the code. Between each on/off state change, we wait for one second using `delay(1000)` before executing the next line of code.

Test Your Skills

How did it go? Do you see two red LEDs blinking on the top of Alvik when you run the code?

If you were successful, you managed to make both RGB LEDs slowly blink on/off in the color red, but let's see how clever you are! Try out these challenges to test your skills:

- **Can you program an alternating blinking pattern?**
- **Can you create new colors like Blue or Green?**
- **Can you mix colors to create a new color such as Yellow?**
- **Can you create a unique multi-color blinking pattern?**

Make It Move

This is the moment you have been waiting for - let's code Alvik to move!

Create a new file called *make_it_move.py* in your projects folder and copy the code below into your script.

```
from arduino import *
from arduino_alvik import ArduinoAlvik

alvik = ArduinoAlvik()

def setup():
  alvik.begin()
  delay(1000)

def loop():
  # Drive forward
  alvik.set_wheels_speed(10,10)
  delay(2000)
  # Turn left
  alvik.set_wheels_speed(0,20)
  delay(2000)
  # Turn right
  alvik.set_wheels_speed(20,0)
  delay(2000)
  # Drive backwards
```

If you look carefully at the code, you will find many similarities to our previous examples with LEDs. We initialize the robot, create an infinite loop, use a special method to control hardware, and include delays to control timing. Really, as you will soon discover, that's the cool thing about using MicroPython with Alvik - it is intuitive and easy to learn, meaning you will be a pro in no time!

The method we use here is `alvik.set_wheels_speed()` to control the individual speeds of the left and right wheels, measured in **Revolutions Per Minute (RPM)**. The arguments of our first method are `(10, 10)` which translate to **(Left Wheel = 10 RPM, Right Wheel = 10 RPM)**. In this way, it is easy to remember which wheel motor you're controlling: left of the comma is the left wheel and right of the comma is the right wheel. In the example, you will find four movements - observe the comments above each method and try running the program to see it in action.

> **NOTE:** We have learned how to make Alvik perform different tasks while connected to the PC, but it can do much more by itself. **In the next lesson,**

you will learn how to run programs freely without a cable.

Dance Challenge

Okay, you can make the robot move... but have you got the moves? Try out the challenge below and have fun with it!

Think about the different ways you can control the movements and independent RPM values of each wheel. How could you combine different movements to make Alvik dance? Think about other methods and hardware you have tested - could you use these to make the dance more vibrant?

Try competing against your friends in a mechanical dance competition. Humans may also participate, but only if they dance "the robot"!