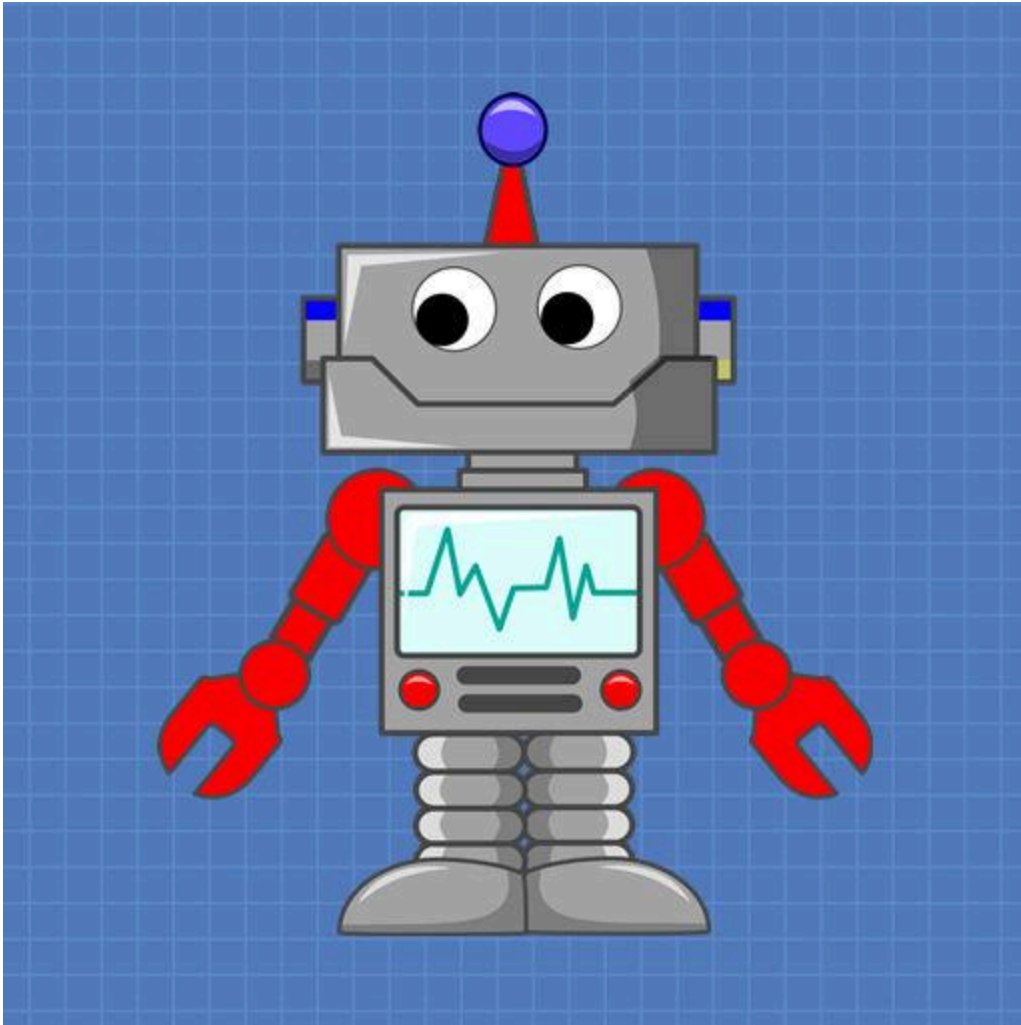


Arduino ALVIK Robot - Makers Guide



DroneBot Workshop Tutorial

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

Today, we will look at Alvik, a new robotic experimenter platform from Arduino. While this product targets the STEAM educational market, we will see how it is also ideal for makers and robotics experimenters of all ages.



Introduction

We have featured many robotic projects in the DroneBot Workshop, ranging from simple object-avoidance bots to more sophisticated designs. Robotics is a fascinating field; getting experience working with and programming robots is valuable for students and electronics experimenters.

However, while building a robot can be a valuable experience, it can also be difficult to source parts. For those with expertise in software development, working with hardware can also be challenging, especially if soldering is involved.

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

Purchasing a pre-built robotics platform makes sense. It lets you focus on software development without any hardware concerns. It also allows you to exchange code with other developers using an identical platform.

Today, we will be working with an Arduino robotics development platform. Although this robot is geared towards the educational market, it is also ideal for makers and electronics experimenters who want a good foundation for their robotics projects.

The Arduino Alvik



The Arduino Alvik packs a lot of robotics hardware in a small package. It's a complete, ready-to-go robot that can be programmed in three different languages:

- MicroPython
- Arduino C++
- Block-based programming

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

The block-based language is excellent for students learning programming concepts; however, we will focus on just MicroPython and C++ today.

Alvik Specifications

Two microcontrollers power the Alvik:

- An Arduino Nano ESP32
- An STM32 Arm Cortex-M4 32-bit processor

The STM32 Arm Cortex-M4 microcontroller directly controls the motors, sensors, and LEDs. It communicates with the Nano ESP32 and can be controlled by the Nano using a set of dedicated APIs, which are available for MicroPython and C++.

The Alvik has several integrated sensors for sampling its environment:

- **Color Detector** – [Broadcom APDS 9660](#) – This is an I2C-compatible interface that provides red, green, blue, and clear (RGBC) sensing.
- **Inertial Measurement Unit (IMU)** – [STMicroelectronics LSM6DSOX](#) – A 6-axis IMU (inertial measurement unit) with embedded AI.
- **Time of Flight (TOF) Distance Sensor** – [STMicroelectronics VL53L7CXV0GC](#) – Time-of-Flight 8×8 multizone ranging sensor with 90° Field-of-View.
- **Line Follower Array** – 3 custom-made IR sources & sensors.
- **Capacitive Touch Switches** – 7 switches.

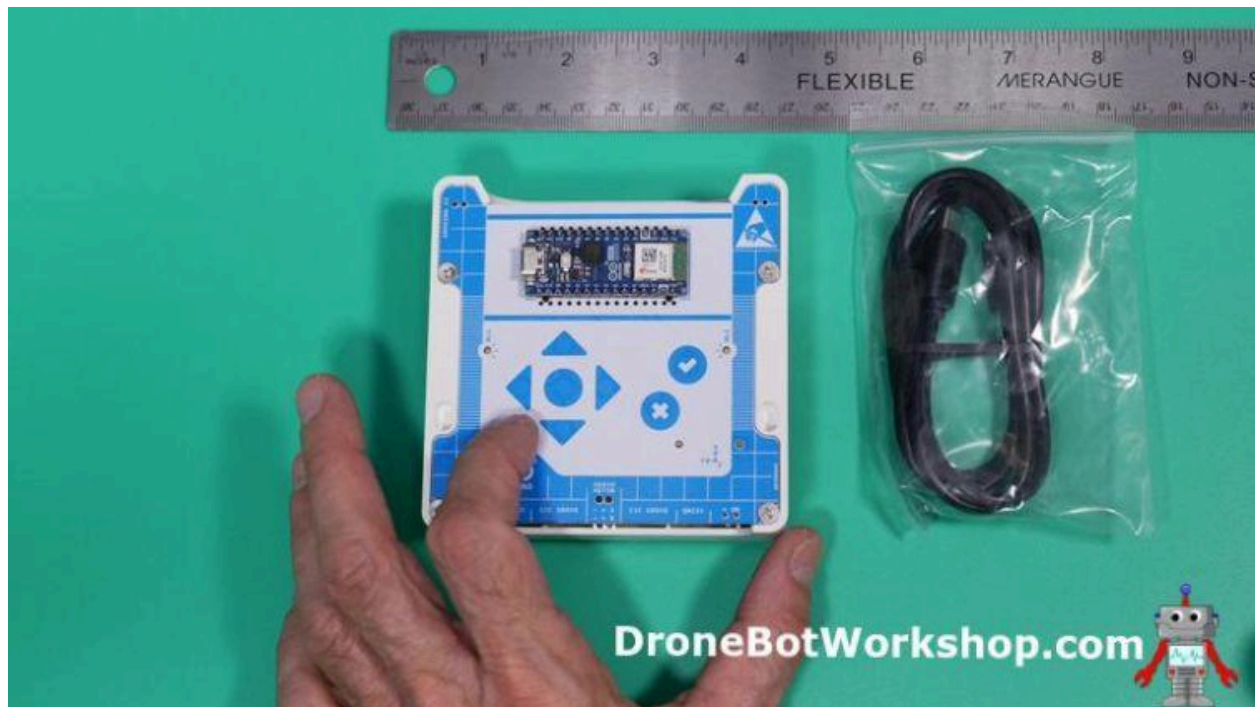
There is also a connector for two servo motors and two pairs of QWIIC and Grove connectors for the Nano I2C bus.

The Alvik is a two-wheeled or “differential” drive robot. Each 34mm rubber wheel is driven by its own GM12-N20VA-08255-150-EN DC motor, with a magnetic (hall effect) relative encoder to provide positional feedback. The motors operate at 6 volts and have a maximum speed of 96 rpm. A ball castor at the robot’s rear maintains balance.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

The two motors are driven with a [Maxim Electronics MAX22211](#) dual FET H-Bridge motor driver.

There are also two RGB LEDs on the top of the robot. Both of these are fully programmable.



The Alvik is powered by a single 18650 Li-Ion battery, which can be charged onboard.

Battery management is performed by a [Maxim Electronics MAX17332X22 BMS](#).

Internally, both a boost and step-down converter are used to provide three voltages:

- **3.7 volts (unregulated)** – This voltage is directly from the 18650 battery.
- **5 volts** – The output of the Boost Converter.
- **3.3 volts** – The output of the Step-Down converter.

There are also a pair of voltage level shifters that provide the 6 volts for the motors.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

The user can remove and replace the battery. An [Analog Devices LTC4360CSC8-2](#) reverse-voltage protector guards against an incorrectly inserted battery.

The Arduino Nano ESP32 on the Alvik has a USB-C connector for data transfer. A USB-C cable is included with the robot.

Physical Expansion

While the Alvik is a self-contained unit, it can also be expanded with user-designed attachments and peripherals.

The robot can attach to Lego Technics pieces, allowing for various custom designs. There are two Lego-compatible peg holes on each side of the robot.

You can also attach additional components using four M3 screws on each side of the robot. To avoid damaging the Alvik, the maximum screw length should be 10mm (M3x10).

MicroPython Programming

[MicroPython](#) is a scaled-down implementation of the Python 3 language specifically developed for microcontrollers and embedded devices. It is ideal for students and developers with [Python](#) coding experience.

The Arduino Nano ESP32 used in the Alvik can be programmed in MicroPython. Arduino has provided a library, some sample code to get you started, and an IDE to work with MicroPython programming.

Alvik MicroPython Library

The Arduino Alvik MicroPython Library is a powerful tool designed to simplify programming and control of the Arduino Alvik robot using MicroPython.

The primary purpose of the Arduino Alvik MicroPython Library is to abstract the lower-level hardware interactions, enabling users to control the robot and explore various programming concepts easily.

The library supports functions for motor control, sensor interaction, LED management, and more, all within an easy-to-use Python syntax. It lets users focus on developing code, algorithms, and robot behaviors without getting bogged down by hardware-specific details.

The Arduino Alvik MicroPython Library includes several code examples that you can try out. The *demo.py* example is running on the Alvik when you first unpack it.

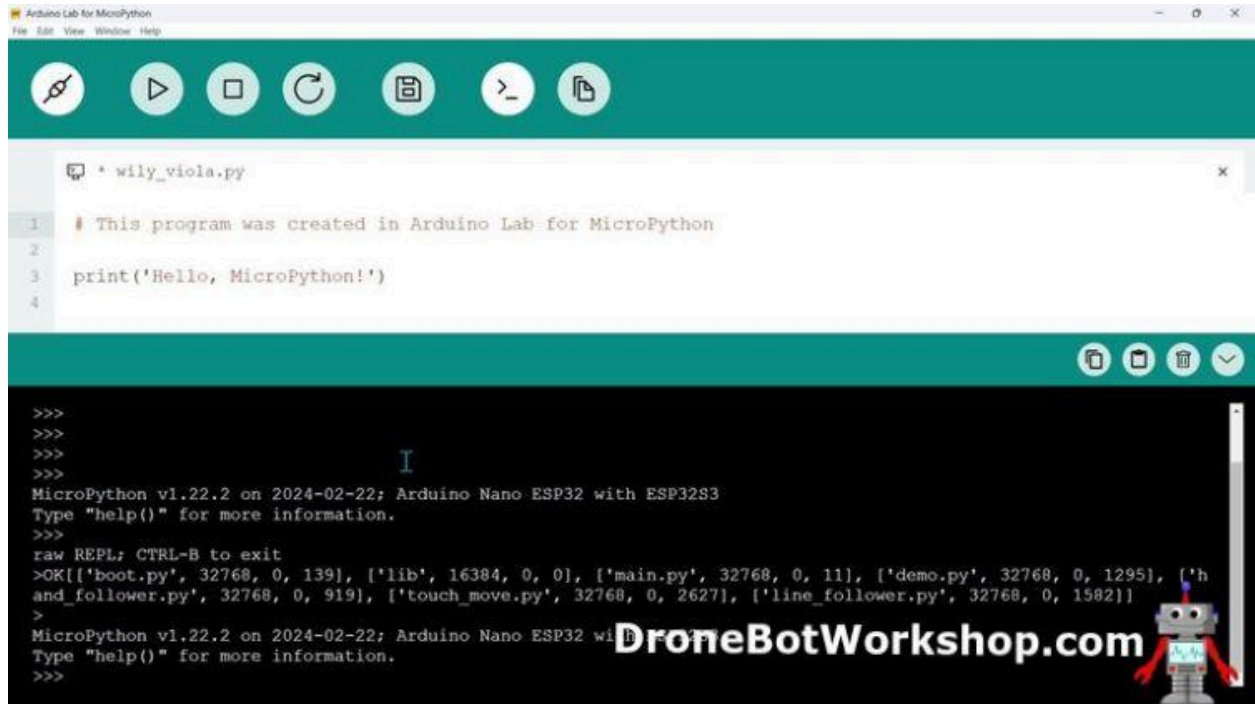
The library's API is well-documented on [Arduino's Documentation Website](#).

Arduino Lab for MicroPython

The [Arduino Lab for MicroPython](#) is a lightweight editor that facilitates interaction between your computer and MicroPython-compatible Arduino boards. It is available for Linux, MacOS (Intel & Silicon versions), and Microsoft Windows.

This is a simple MicroPython editor sponsored by Arduino and based on [Murilo Polese's](#) original work. It looks similar to the older Arduino IDE 1.8 or the Arduino online editor.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>



As of this writing, the Arduino Lab for MicroPython is beta software, but it is stable and suitable for use with the Arduino Alvik.

The Arduino Lab for MicroPython code editor is simple to use, making it ideal for beginners. It can be used to both edit code and to manage files.

Using the Arduino Lab for MicroPython

The top of the editor interface is a toolbar with the following buttons:

- **Connect / Disconnect** – This lets you connect and disconnect an external microcontroller. The icon changes shape according to the connection status.
- **Run** – Run the program in the editor.
- **Stop** – Stop a running program.
- **Reset** – Reset the connected microcontroller (Alvik must be on for this to work).
- **Save** – Save the current editor document. The file will be saved in the default location selected when installing the Arduino Lab for MicroPython.
- **Editor and REPL**—Set the interface to Editor and REPL mode. In this mode, the top pane is used to edit code, and the bottom pane is used for REPL (Read,

<https://dronebotworkshop.com>

Evaluate, Print & Loop). This is the mode in which the Arduino Lab for Micropython boots up. It can also function as a serial monitor.

- **File Manager** – In this mode, the interface displays two panes. One pane displays the contents of the microcontroller file system, whereas the other pane shows the files in the default working directory on the local computer.

The editor is designed to be user-friendly, allowing students and experimenters to focus on learning and creating rather than struggling with complex development environments.

Let's use the editor to edit a Micropython program and run it on the Alvik.

Obstacle Avoidance Code

I assume that you have already installed the Arduino Lab for Micropython on your workstation and set up a default folder to save your programs locally. Open the program and connect a USB-C cable to the Arduino Nano ESP32 on the Alvik (a cable is included with the robot). **Make sure that the Alvik is OFF before connecting the cable!**

Use the *Connect* button to select the robot's USB port. When the connection is made, a status display message should appear in the REPL window.

Now click on the File Manager icon. This will change the interface to file management mode, and you will see the files in your local directory displayed on the right pane.

We will need to create a new file to hold our program. Click the New File icon (top left) and name the file "*obstacle_avoider.py*." Now, double-click the file to open it in editor mode.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

You will see that the Arduino Lab for Micropython has already added a few statements to the file. Highlight and delete all the text so you are working with a clean slate.

Now enter the following (Arduino-supplied) code:

```
1 from arduino_alvik import ArduinoAlvik
2 import time
3
4 # Initialize Alvik
5 alvik = ArduinoAlvik()
6 alvik.begin()
7
8 while True:
9     # Moves servo 0
10    alvik.set_servo_positions(0,0)
11    time.sleep(2)
12    alvik.set_servo_positions(90,0)
13    time.sleep(2)
14    alvik.set_servo_positions(180,0)
15    time.sleep(2)
16    alvik.set_servo_positions(90,0)
17    time.sleep(2)
18    # Moves servo 1
19    alvik.set_servo_positions(0,1)
20    time.sleep(2)
21    alvik.set_servo_positions(90,1)
22    time.sleep(2)
23    alvik.set_servo_positions(180,1)
24    time.sleep(2)
```

```
25     alvik.set_servo_positions(90,1)
26     time.sleep(2)
```

Click the Save button to save the code. Before we do anything else, let's examine the code to see how it works.

Obstacle Avoider Code

We start by importing the relevant libraries, including the ArduinoAlvik library. We then define an ArduinoAlvik object, start it, and wait five seconds for the robot to complete its warm-up sequence.

Next, we define a few parameters:

- ***distance*** – The minimum allowable distance between the robot and any obstacle(s).
- ***degrees*** – The number of degrees to turn when an obstacle is encountered.
- ***speed*** – How fast to move the robot when driving straight ahead.

Of course, you can play around with these values to see how they affect the robot's behavior.

Now, we run a loop, which starts by measuring the distance from each of the robot's sensors. Once the distances are measured, we print these values to the console.

Then, we check the sensor values and compare them to our *distance* variable. If any of them are less than the defined *distance*, we move the robot by the amount specified by the *degrees* variable. It's pretty simple, but it gets the job done. Note how the Alvik library is used to position the robot easily.

If none of the sensors show an obstacle within the defined distance, we drive straight ahead at the specified speed.

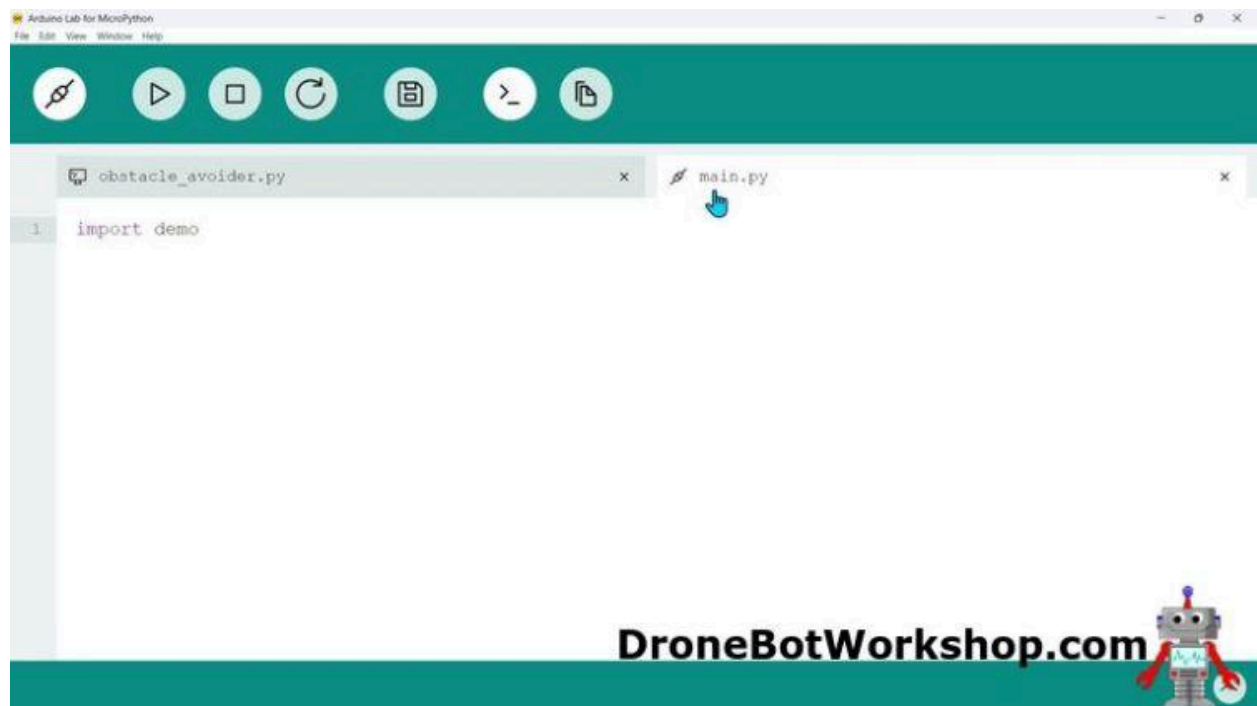
Running the Code on the Alvik

Now that we know how our program works, it's time to load it onto the Alvik and test it.

Click the icon on the toolbar to place the Arduino Lab for Micropython back in File Manager mode. If you have followed these instructions, the files on Alvik's Arduino Nano ESP32 should be displayed in the right panel.

We need to copy the file from our local computer onto the Alvik. Highlight the *obstacle_avoider.py* file on the right pane and click the left-arrow icon to copy it to the Nano ESP32.

Next, we need to modify Alvik's *main.py* file so that our new code will run once Alvik is powered up. *main.py* is a special file that automatically runs when a Micropython microcontroller boots up.



For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

Find the *main.py* file in the left (Alvik) file manager pane and double-click on it. The file will open in the editor in a new tab (the *obstacle_avoider.py* file will remain open).

On a new Alvik, *main.py* has only one line of code—*import demo*. This statement causes Alvik to import and run a file called *demo.py*. You can examine *demo.py* to learn more about the demo program.

We need to edit that line to import our new obstacle avoidance program. Edit *main.py* to be like this:

```
1 import obstacle_avoider
```

Once you have edited and saved *main.py*, we will be ready to go. Disconnect the USB cable from the Alvik and turn on the robot with its tiny little power switch.

Now place it on the floor and watch it go! It should react to any obstacles in its path.



For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

You can experiment with the three code variables if you wish. You can also use the technique of calling your Micropython programs from *main.py* for other projects that you develop on your Alvik.

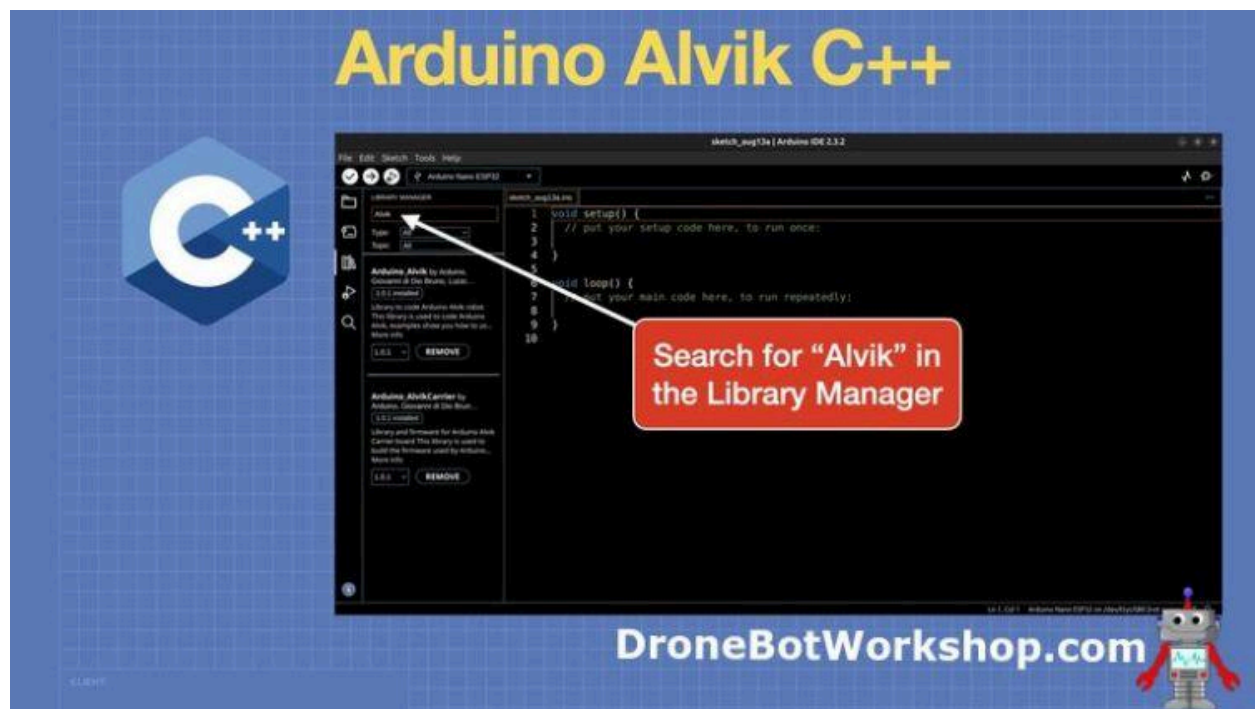
For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

Arduino C++ Programming

Long-time Arduino enthusiasts will be familiar with the C++ implementation used with the Arduino Uno and other microcontrollers. The Alvik is based on an Arduino Nano ESP32 board, which easily supports C++. Arduino has created a couple of libraries to simplify programming the Alvik using the popular Arduino IDE. Of course, you can combine these with other Arduino libraries to program just about anything.

Alvik C++ Libraries

Arduino has provided two libraries for the Alvik. One library has functions for working with the Nano ESP32; the other is for working with the “carrier” STM32 microcontroller.



Arduino_Alvik Library

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

The Arduino_Alvik library provides a set of functions and classes to interact with Alvik's hardware components. Key functions and uses of the library include:

- **Motor Control** – Manage speed, direction, and braking of the robot's motors.
- **Sensor Integration** – Read data from sensors (e.g., distance, color, touch).
- **LED Control** – Manipulate onboard LEDs for visual feedback.
- **Sound Control** – Play sounds and melodies.
- **Movement Control** – Implement movements (e.g., forward, backward, turn).
- **Battery Management** – Monitor battery levels.

The library is packaged with some sample sketches to get you started:

- **bridge_firmware_updater** – Convert your Arduino Nano ESP32 to a USB-to-serial adapter to flash firmware into the Arduino Alvik Carrier board.
- **color_calibration** – This sketch calibrates the color sensors on the Arduino Alvik.
- **drive** – This sketch demonstrates the basic movement control of the Alvik Robot. It includes commands to move the robot forward, backward, and turn in different directions
- **hand_follower** – This sketch is a practical example of how to use Alvik's proximity sensors.
- **line_follower** – Demonstrates the use of Alvik's line following sensors.
- **remote_control** – This example shows how to interface 2 Alvik robots using ESP-Now.

Arduino_AlvikCarrier Library

The Arduino_AlvikCarrier C++ library is a specialized software library designed for interacting with the Alvik Carrier board and its STM32 Arm Cortex-M4 32-bit processor.

Key functions include:

- **Motor Position** – Get feedback on the position of the motors and control their speed precisely.
- **Sensor Data** – Read data directly from sensors.

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

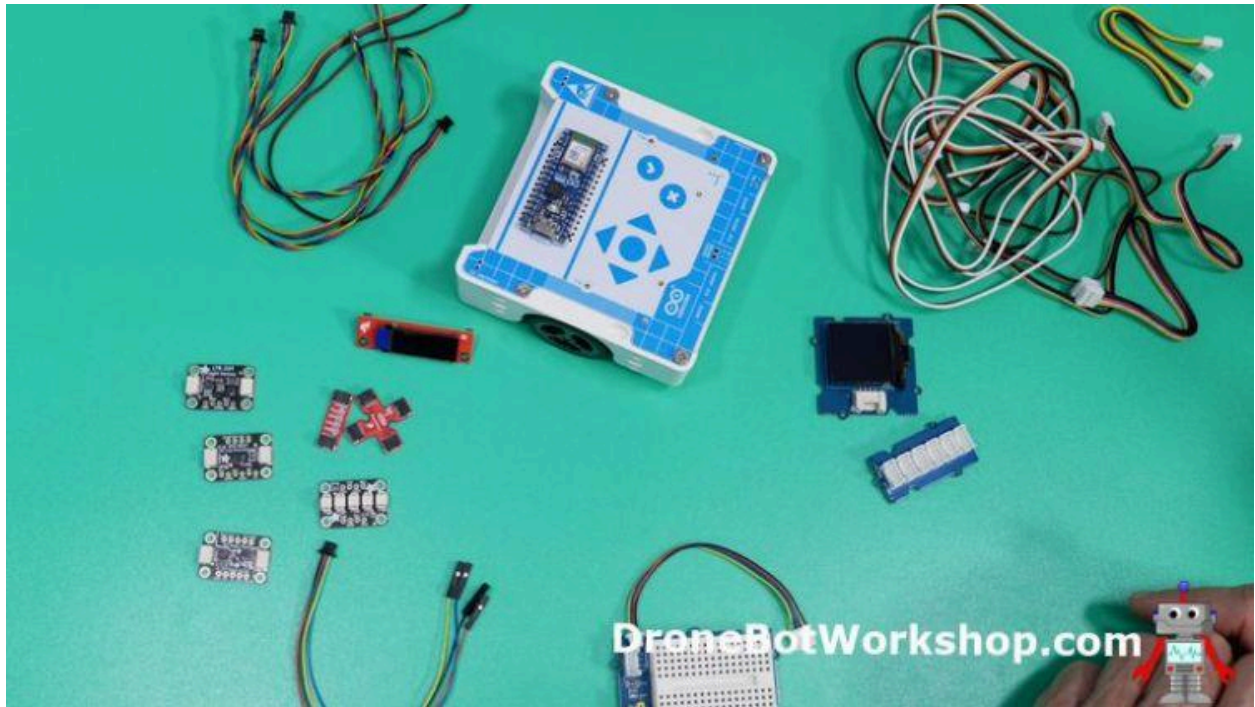
- **Firmware Reload** – Reload the Alvik Carrier board firmware.

The library includes the following example sketches:

- **blink** – The classic “blink” sketch, rewritten to control an onboard LED.
- **firmware** – Update and manage the firmware on the Alvik Carrier Board.
- **imu** – An example for obtaining IMU data.
- **kinematics** – Control the motors and retrieve odometry data.
- **motor** – A sketch that shows how wheel velocity controls can be used.
- **position** – This example shows how left wheel position control can be used.

Expanding the Alvik

One key feature of the Alvik is its ease of expansion. Arduino has provided several data expansion connectors, all using the I2C bus. There is also a connector that supports two external Servo motors.

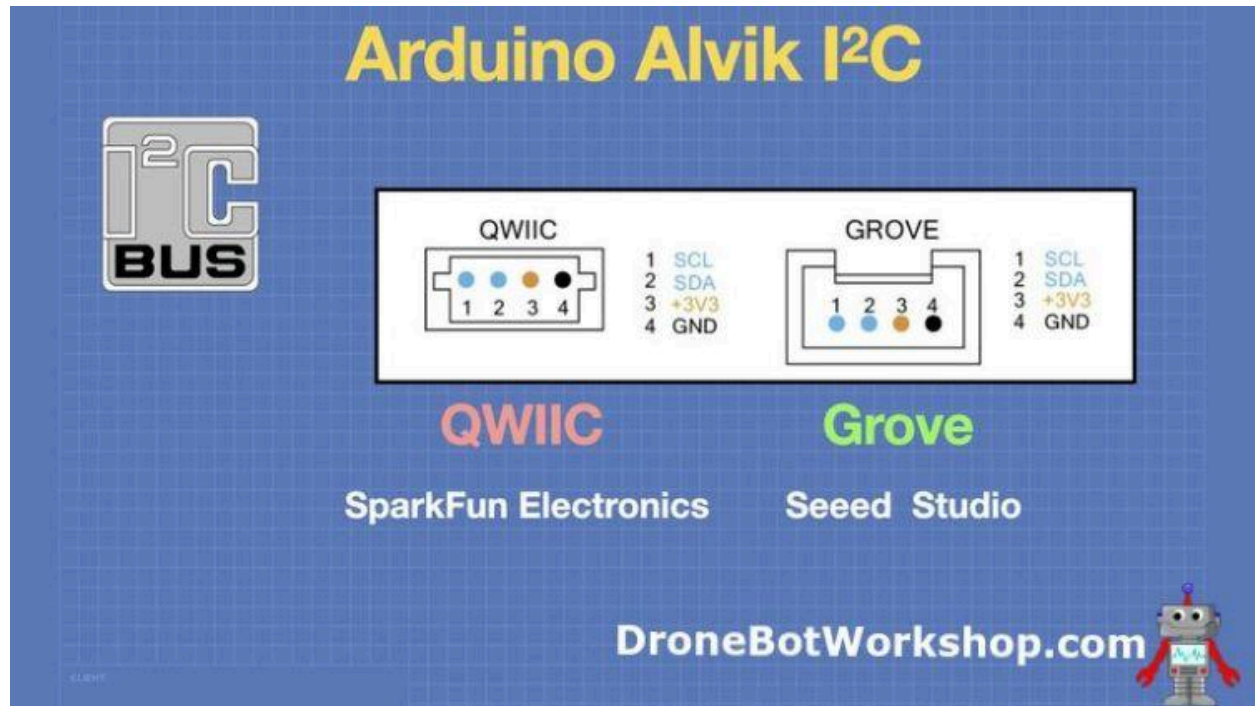


All of the connectors use the same I2C bus, wired in parallel. Here is a MicroPython program that will scan the I2C bus to identify the address(es) of any attached devices.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

```
1 from machine import I2C
2 from machine import Pin
3
4 i2c = I2C(0, scl=Pin(12, Pin.OUT), sda=Pin(11, Pin.OUT))
5
6 print()
7 print('Scan i2c bus...')
8 print()
9
10 devices = i2c.scan()
11
12 if len(devices) == 0:
13     print("No i2c device !")
14 else:
15     print('i2c devices found:',len(devices))
16 print()
17
18 for device in devices:
19     print("Decimal address: ",device," | Hexa address: ",hex(device))
20
21 print()
```

You can run this in the Arduino Lab for MicroPython and use REPL to observe the results.



QWIIC Connector

The QWIIC Connect System by SparkFun Electronics is an interconnect system that connects sensors, actuators, and other devices to microcontrollers. It uses 4-pin JST connectors to establish quick and reliable connections.

The Alvik has two QWIIC connectors.

Grove Connector

Grove connectors are a modular, standardized connector system for connecting various electronic modules, sensors, actuators, and displays to microcontrollers. Developed by Seeed Studio, Grove connectors feature a simple, color-coded, and intuitive design, making it easy to connect and swap modules without the need for soldering or complex wiring.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

The Alvik has two Grove connectors.

Servo Connector

The Alvik also has a pair of 3-pin connectors for powering and driving external servo motors. The pinouts for each servo are as follows:

- Servo
- + 5V Servo Power
- Ground

This pinout is compatible with most 5-volt hobby servos, such as the SG90.



The [*set_servo_positions function*](#) in the Arduino Alvik Library can be used to control the two servo motors. This MicroPython code is an example of moving two servo motors attached to the Alvik.

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

```
1 from arduino_alvik import ArduinoAlvik
2 import time
3
4 # Initialize Alvik
5 alvik = ArduinoAlvik()
6 alvik.begin()
7
8 while True:
9     # Moves servo A
10    alvik.set_servo_positions(0,0)
11    time.sleep(2)
12    alvik.set_servo_positions(90,0)
13    time.sleep(2)
14    alvik.set_servo_positions(180,0)
15    time.sleep(2)
16    alvik.set_servo_positions(90,0)
17    time.sleep(2)
18    # Moves servo B
19    alvik.set_servo_positions(0,0)
20    time.sleep(2)
21    alvik.set_servo_positions(0,90)
22    time.sleep(2)
23    alvik.set_servo_positions(0,180)
24    time.sleep(2)
25    alvik.set_servo_positions(0,90)
26    time.sleep(2)
27    # Moves servos A & B
28    alvik.set_servo_positions(0,0)
```

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

```
29     time.sleep(2)
30     alvik.set_servo_positions(90,90)
31     time.sleep(2)
32     alvik.set_servo_positions(180,180)
33     time.sleep(2)
34     alvik.set_servo_positions(90,90)
35     time.sleep(2)
```

You can make use of the M3 screw threads to build a mount for your servo motors. You can even use them to design a (very) simple arm for Alvik!

Conclusion

Robotics is a fusion of many disciplines—electronics, mechanics, and software must work together to create an intelligent machine. Having the electronics and mechanics done for you can allow a developer to focus on writing innovative software to bring their mechanical creature to life.

The Arduino Alvik can be used for such a purpose. While its appeal to the education market is obvious, it can also be a valuable tool for makers and developers who just want a well-documented platform to build upon and write code. With its exposed I2C bus and software code examples in MicroPython and C++, Alvik can be easily expanded to be whatever you need.

If you're looking for a robotics platform to build upon, I'd recommend considering the Arduino Alvik.

And if you're wondering about the name, it's a bit of a mystery. However, I can't help but note that "Alvik" is the Swedish word for "elf," and it is a little elf of a robot!

<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

Parts List

Here are some components you might need to complete the experiments in this article. Please note that some of these links may be affiliate links, and the DroneBot Workshop may receive a commission on your purchases. This does not increase the cost to you and is a method of supporting this ad-free website.

Arduino Alvik – [Arduino Store](#)

Arduino Nano ESP32 – [Arduino Store](#)

Resources

[Arduino Alvik User Manual](#) – Official user guide from Arduino.

[Getting Started with Alvik](#) – Arduinos guide for beginners.