ARDUINO LANGUAGES

Peter’s Exploration of Languages

JUNE 19, 2025

NEXT GENERATION ART AND TECHNOLOGY

Canada

Contents

Overview

There are many languages a programmer can use with the Arduino. While C, C++, and Assembly are the primary languages supported directly by the Arduino platform, several other languages can be used through alternative tools and frameworks. These languages include MicroPython, JavaScript (Espruino), Rust, and Lua, depending on the hardware and development environment. Additionally, there are Graphical and Educational Languages like Scratch and Blockly, which are excellent for beginners and are supported by platforms such as mBlock and Tinkercad Circuits.

Each language offers different advantages based on your goals—whether it's **low-level hardware control** with **Assembly** for writing precise timing routines (e.g., bit-banging a custom protocol), **rapid prototyping** with **CircuitPython** for quickly testing sensors using board.D5 and digitalio, or **teaching fundamentals** with **Scratch for Arduino (S4A)** to introduce students to logic and control structures through visual blocks.

The main languages for Arduino are C, C++, and Assembly because they provide direct access to the hardware. These languages are highly efficient, making them ideal for devices with limited memory and processing power. The Arduino's toolchain is built around **avr-gcc**, a C/C++ compiler optimized for AVR microcontrollers.

C++ enables the use of object-oriented programming, which is helpful for organizing complex projects. For example, you can create a Motor class to manage speed and direction, encapsulating all related functions and variables for easier maintenance and code reuse.

Assembly allows for ultra-low-level control, useful in performance-critical or timing-sensitive tasks. For instance, precise timing routines for generating PWM signals or reading a sensor with nanosecond accuracy might benefit from writing a loop in Assembly.

Most official Arduino libraries and the Arduino core itself are written in **C/C++**. For example, the Wire.h library (used for I2C communication) and Servo.h (for controlling servo motors) are both written in C++.

These languages offer full control over memory allocation, timers, interrupts, and I/O. You can directly manipulate hardware registers like PORTB or use ISR() functions to write custom interrupt service routines.

They are well-documented and widely supported by the Arduino community, with thousands of tutorials and examples available. **C and C++** are also portable across many microcontroller platforms (e.g., ARM-based boards like Arduino Due), making your code more reusable.

Together, C, C++ and assembly strike a balance between high performance and programmer productivity. This is crucial in embedded development, especially when programs become longer and more complicated.

Languages for Arduino

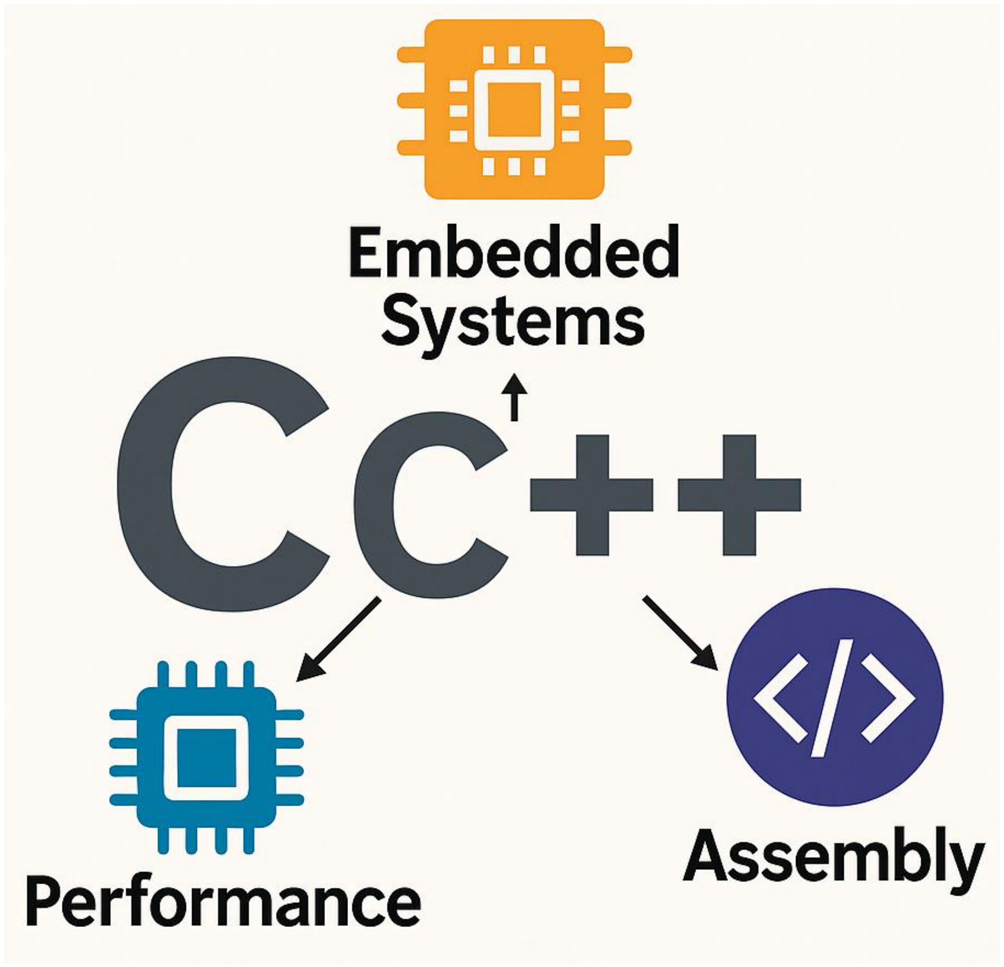
Primary Language

C and C++

C and C++ are the official programming languages for Arduino, with the Arduino language being a simplified version of C++. Arduino sketches are structured using familiar functions like setup() and loop() to define initialization and repeated behavior. Developers can also include standard C libraries and, if necessary, write code entirely in pure C, offering flexibility and control for a wide range of applications.

Assembly

A contrasting language to C and C++ is the assembly language. Assembly is supported on AVR-based Arduino boards, such as the Arduino Uno, either through inline assembly within C/C++ code or via separate .S files. It is particularly useful for low-level hardware control and performance-critical optimizations. Assembly provides precise timing and direct access to registers, making it ideal for tasks that require tight control over the microcontroller's operations.



Other Languages (via alternate tools/frameworks)

MicroPython / CircuitPython

MicroPython and CircuitPython are supported on select ARM-based boards such as the Arduino Nano 33 BLE, ESP32, and various Adafruit boards. These languages are not officially supported by the classic Arduino IDE but can be used through alternative platforms like Thonny or Adafruit's own development tools. MicroPython is a lightweight implementation of Python 3 tailored for microcontrollers, while CircuitPython is a beginner-friendly fork created by Adafruit, offering additional libraries and simplified hardware support specifically designed for their products.

JavaScript (Espruino)

JavaScript can be used for embedded development on Arduino-like boards through the Espruino firmware, which allows developers to write code similar to Arduino syntax using JavaScript. This approach works only on boards that specifically support Espruino and is not compatible with classic AVR-based Arduino boards like the Uno. Espruino is designed for simplicity and low memory usage, making it suitable for rapid prototyping and IoT applications. It also supports interactive programming via a console, allowing code to be tested and updated on the fly without needing to recompile. This makes it especially appealing for beginners or developers coming from a web development background.

Rust

The Rust programming language can be used to target embedded systems, including Arduino boards, although it requires a more advanced and manual setup using tools like avr-rust for AVR chips or embedded-hal for Cortex-M devices. Rust differs significantly from more traditional programming languages such as C, C++, or Java, which are often introduced to beginners. Its design eliminates common issues like null pointer dereferencing and buffer overflows, and it includes strong compile-time checks that help catch logic errors early. While these features enhance reliability and safety, they may introduce complexity that could be confusing for beginners, depending on their learning goals.

Go, Python, Lua, etc.

Languages like Go, Python, Lua, and others can be used with Arduino indirectly through specialized frameworks, firmware, or transpilers. For example, the NodeMCU firmware allows Lua programming on ESP8266-based boards, enabling developers to write simple scripts for IoT tasks. These alternative languages are generally not supported by the standard Arduino IDE, but they provide flexible options for specific use cases, particularly when working with networked or web-connected devices. Although they lack the deep hardware access that C/C++ provides, they offer faster development cycles and simpler syntax for certain tasks. These languages can be particularly useful for developers transitioning from other environments or for rapid prototyping where low-level control isn't a primary concern.

Graphical & Educational Languages

Blockly / Scratch

Blockly and Scratch are visual programming languages that use a drag-and-drop interface, making them ideal for beginners, educators, and younger students learning to program with Arduino. Tools like Tinkercad Circuits, Ardublock, and mBlock incorporate these block-based languages to simplify coding by allowing users to create logic visually without writing traditional code. These platforms translate visual blocks into actual Arduino C/C++ code in the background, helping users gradually transition to text-based programming. They are commonly used in classrooms and STEM education programs to teach fundamental programming concepts such as loops, conditionals, and variables in a fun and approachable way. Additionally, mBlock offers compatibility with more advanced boards and sensors, bridging the gap between entry-level learning and real-world Arduino applications.

Tinkercad Link: https://www.tinkercad.com/
mBlock Link: https://ide.mblock.cc/

when flag
move 10 steps
turn right 15 degrees
repeat 10
if touching mouse-pointer ?
say Hello!'
go to random position

Summary Table

Language	**Platform/Boards**	**Notes**
C/C++	All Arduino boards	Official language
Assembly (ASM)	AVR boards (e.g., Uno, Nano)	For performance, low-level access
MicroPython	ESP32, RP2040, Nano 33 BLE	Not for classic Uno
JavaScript	Espruino boards	Via Espruino firmware
Rust	Uno, Cortex-M boards	Advanced, via avr-rust or embedded-hal
Lua	ESP8266 (NodeMCU)	Through Lua firmware
Blockly/Scratch	mBlock, Tinkercad, Arduino EDU	Beginner-friendly, visual coding

Resources

If you want to dive deeper into Arduino and its languages, feel free to explore these resources. My GitHub and YouTube explores the primary language of the Arduino. If you are curious or need any guidance, do not hesitate to contact me—I will be happy to help you get started!

YouTube: https://www.youtube.com/channel/UCOv_iZCx4CW5Y9S8YzXDdgg

GitHub: https://github.com/Vaughan-Peter?tab=repositories

LinkedIn: https://www.linkedin.com/in/peter-vaughan-997478239/

E-mail: otherhalifaxprojects@gmail.com

Arduino Website: https://www.arduino.cc/