Lab 1- Introduction to Embedded Systems

1. What Is an Embedded System?

An **embedded system** is a purpose-built computer that lives inside a larger product, giving that product its "smarts." Unlike a general-purpose PC or phone—designed to run countless unrelated programs—an embedded system is engineered to perform one (or a small set of) well-defined tasks, often under real-time constraints. At its heart sits a microcontroller or system-on-chip that combines processor, memory, and peripherals on a single silicon die. Surrounding it are the application-specific inputs (sensors, buttons, communications buses) and outputs (motors, displays, LEDs, actuators) needed to interact with the physical world.

Several traits distinguish embedded systems:

- **Dedicated functionality:** The hardware and firmware are tailored to a specific job—e.g., regulating a thermostat, steering a drone, or managing an airbag deploy sequence.
- **Resource constraints:** They must deliver reliability and determinism with limited CPU speed, memory, and energy budget—often running from batteries or harvesting power.
- **Real-time responsiveness:** Many embedded tasks (closing a motor control loop, timestamping a GPS pulse) require guaranteed reaction within micro- or milliseconds.
- **Tight integration with hardware:** Firmware often manipulates registers directly, interfaces via SPI/I²C/UART, and handles interrupts to meet timing.
- **Cost and size optimization:** Billions ship in consumer goods, so every gram, mill watt, and cent matters.

Put simply, if a computer is invisibly embedded in a product, dedicated to making that product operate safely, efficiently, or "smartly," it earns the name **embedded system**

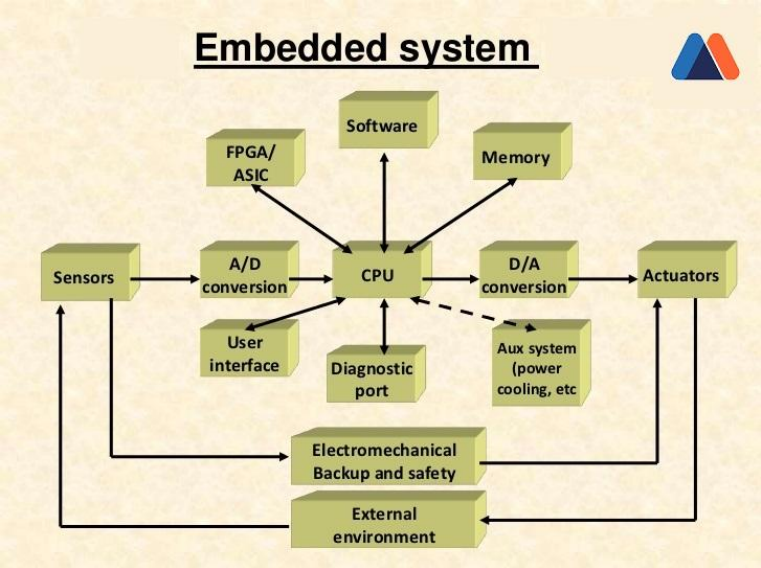

Figure 1: Embedded System https://www.heavy.ai/technical-glossary/embedded-systems

2. Course Platforms Overview

In this lab you'll gain hands-on experience with **two complementary Arduino-based platforms**:

- **Arduino Uno R3** – the classic 5 V, ATmega328P board that sparked the maker movement. With 14 digital I/O pins (6 PWM), six analog inputs and the familiar USB-B programming port, it is still the most documented board in the Arduino family docs.arduino.cc. At home you'll pair the Uno with the **Sidekick Basic Kit for Arduino V2**, a parts box that includes a breadboard, jumper wires, LEDs, push-buttons, a buzzer, resistors and small sensors—everything you need to practice fundamental electronics and "blink-to-sensor" sketches on your own desk wiki.seeedstudio.com.
- **Arduino Alvik** – a modern, dual-MCU educational robot (Nano ESP32 controller + STM32 Arm® co-processor) equipped with encoders, ToF distance matrix, line & colour sensors, inertial unit, RGB LEDs and Grove/Qwiic expansion ports. Alvik ships with MicroPython support out of the box and will be the center-piece of your in-lab experiments in smart robotics and autonomous behavior arduino.cc.

Using the **Uno at home** builds your circuit-wiring confidence and C/C++ coding fluency, while the **Alvik in class** lets you tackle higher-level robotics problems—sensor fusion, closed-loop motion and competition-ready behaviors—without first assembling a drivetrain. Together they create a smooth learning curve from discrete-component prototyping to full-featured embedded-robot design.

3. What exactly is an Arduino? — Quick Summary

- **Open-source electronics platform** – Both the hardware designs and the core software (bootloaders, IDE, libraries) are published under permissive licenses, so anyone can study, modify and build their own boards.
- **Easy-to-use hardware + software** – A family of inexpensive microcontroller boards plus the cross-platform Arduino IDE (or Web Editor/CLI). Sketches are written in a C/C++ dialect that hides low-level register work and comes with hundreds of community libraries.
- **Seamless I/O** – Arduino boards read real-world inputs (potentiometers, light sensors, IMUs, serial data) and convert them into actionable outputs (motor pulses, LED patterns, network packets) with a few lines of code.
- **Huge community ecosystem** – Since its launch in 2005 the platform has grown to millions of users, thousands of tutorials, shields and example projects, making problem-solving and idea sharing remarkably quick.
- **From prototypes to products** – While ideal for beginners, Arduino's open hardware/schematics let advanced users spin custom PCBs or migrate the firmware to production-grade microcontrollers with minimal re-work.

In short, **Arduino lowers the threshold for embedded-system creativity**, letting artists, students and engineers focus on *what* a device should do instead of wrestling with *how* to make a microcontroller talk to the real world.

4. Getting Started – Hands-On References

Activity	Quick Link
Upload *Blink* to Uno	https://docs.arduino.cc/tutorials/uno-rev3/getting-started/
Sidekick Kit tutorial index	https://wiki.seeedstudio.com/Sidekick_Basic_Kit_for_Arduino_V2/
Alvik product page	https://docs.arduino.cc/hardware/alvik/
"Ready, Set, Code" MicroPython intro	https://courses.arduino.cc/explore-robotics-micropython/lessons/getting-started/

5. Uno "Hello World" — Blink & Sidekick Primer

Objective: verify your tool-chain, observe the built-in LED, and practise editing timing parameters.

Hands-on steps

1. *No wiring required* — the Uno has an on-board LED wired to **digital pin 13**.
2. Open **File ▸ Examples ▸ 01.Basics ▸ Blink** in the Arduino IDE.
3. Verify → Upload. The amber LED labelled **"L"** should flash **1 s ON / 1 s OFF**.
4. Edit both occurrences of `delay(1000);` to `delay(250);` → upload and watch a 2 Hz blink.
5. Make the LED stay **100 ms ON / 900 ms OFF**—what human-perception effect do you notice?

Meet the Sidekick Basic Kit V2

Unbox the kit and locate: breadboard, jumper wires, RGB LED, push-button, 10 kΩ potentiometer, photo-resistor, piezo buzzer, and micro-servo. These parts underpin the home exercises (traffic-light, tone generator, servo sweep). Keep everything organised in the labelled tray.

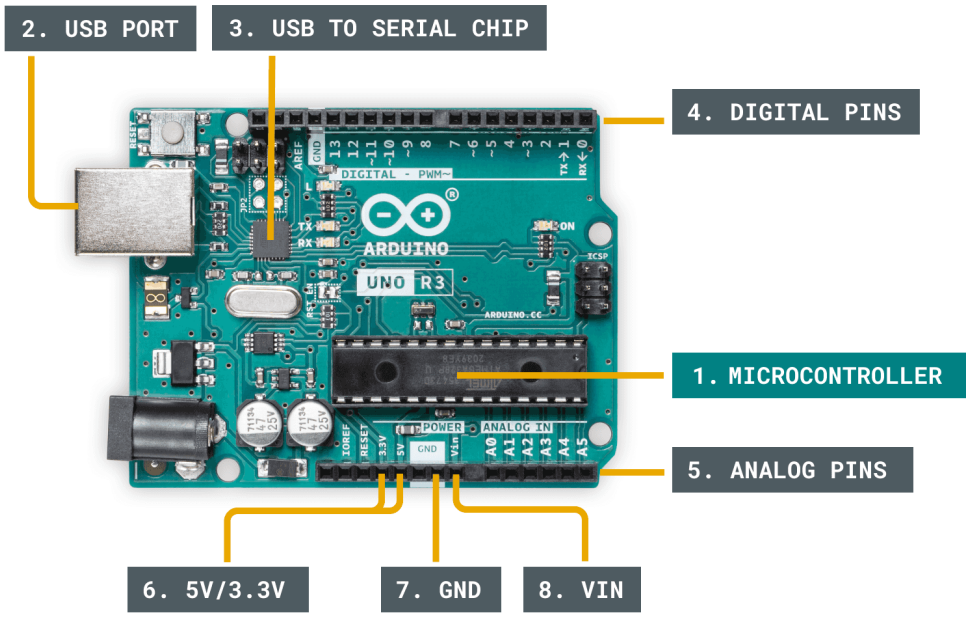

Figure 2: Arduino UNO R3 https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/

6. Getting Started with Alvik Platform

Arduino Alvik is a palm-sized educational robot built for rapid exploration of robotics, IoT, and AI concepts. A **Nano ESP32** module (Wi-Fi / BLE, 16 MB flash) acts as the high-level brain, while an on-board **STM32F411 Arm-Cortex-M4** manages the low-level sensors and actuators. This dual-controller layout keeps real-time motor and sensor tasks deterministic yet leaves plenty of head-room for user code in C++, MicroPython, or block-based environments. docs.arduino.cc

The chassis integrates an **8 × 8 VL53L7CX ToF distance matrix** for 90 ° ranging, a **color / proximity sensor**, a **6-axis IMU**, a **3-zone IR line-follower**, wheel encoders, and two addressable RGB LEDs. Twin DC gear-motors (with hall encoders) provide differential drive at up to ~13 cm s^{-1}. Power comes from a swappable **18650 Li-ion** cell, and expansion is straightforward thanks to dual Grove + Qwiic I^2C ports, two servo headers, and LEGO-compatible mounting points. docs.arduino.ccdocs.arduino.cc

Because the firmware stack and hardware design are fully documented—and paired with downloadable STEP files, example sketches, and an online MicroPython course—Alvik lets learners progress from blinking LEDs to autonomous navigation without first building a drivetrain from scratch.

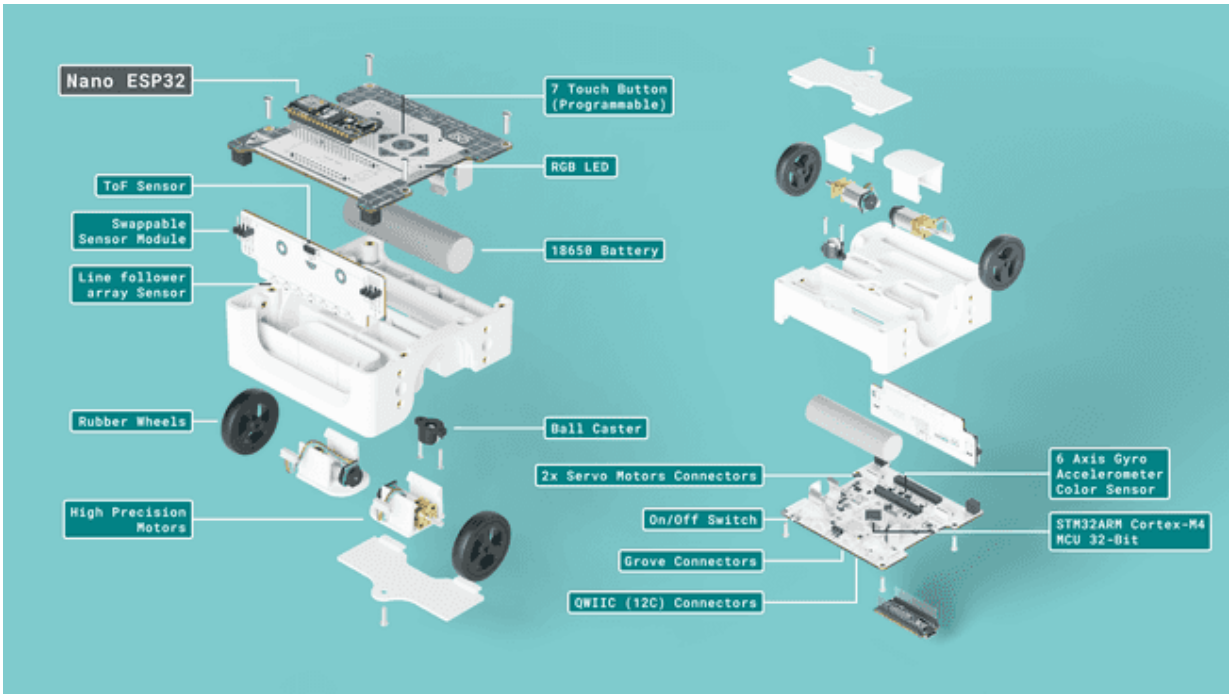

Figure 3: Alvik exploded view https://docs.arduino.cc/tutorials/alvik/user-manual/

Getting Started with Alvik tutorial:

- 1-Ready, Set, Code Explore Robotics in MicroPyton.pdf

Or use the online material available at: https://courses.arduino.cc/explore-robotics-micropython/

- Lessons→Getting Started → Ready, Set, Code https://courses.arduino.cc/explore-robotics-micropython/lessons/getting-started/

For the Lab report:

1. Check-Your-Understanding Questions

a. **Definition**: List three attributes that distinguish an embedded system from a laptop.
b. **Hardware**: How many PWM-capable pins does the Uno R3 expose, and why are PWM pins useful?
c. **Microcontrollers**: What roles do the Nano ESP32 and RP2040 play inside Alvik?
d. **Programming**: In the Uno Blink sketch, which two lines would you edit to double the blink frequency?
e. **MicroPython**: Write a one-line MicroPython command that sets Alvik's wheel speeds to 25 rpm each.
f. **Reflection**: Describe one advantage and one trade-off of using MicroPython versus C++ for embedded development.

2. Post-Lab Reflection Tasks

a. **Experience summary (≤150 words)** – What surprised you most when programming the two platforms?
b. **Block diagram** – Draw a functional block diagram for *make_it_move.py* or *make_it_blink.py* (inputs, processing, outputs).
c. **C++ vs MicroPython essay (½ page)** – Compare workflow, performance, and ecosystem for embedded work on Uno vs Alvik.