Embedded Systems

C++ for Developers

lafftale
for developers

Content

- Embedded Systems
- Components
- How does it work?
- Operating Systems
- Challenges

Embedded Systems

C++ for Developers

What is an embedded system?

What type of systems are we used to? General Purpose Systems like PCs.

Embedded Systems are smaller components, part of a larger system, which have a defined purpose.

Example: PC

A personal computer (PC) is General Purpose, therefore not recognized as an embedded system as a whole.

But if we break components related to the PC into smaller parts:

- GPU
- Screen
- Mouse

These are Embedded Systems - components with limited purposes and a part of a larger system.

Where is the line drawn?

Area	Embedded Systems	General-Purpose Systems
Purpose	Fixed or narrowly defined	Open-ended, can be changed at will
User Programmability	Rarely (or never) directly programmable	Designed to be programmed by the user
Resources	Limited (CPU, memory, storage)	Relatively abundant
UI	Minimal, often non-interactive	Rich user interface, OS-level
Integration	Part of a larger device	Stand-alone
Software	Firmware or tightly bound code	OS + arbitrary user apps
Upgradability	Occasional firmware updates, if any	Frequent, flexible software installs

Is it embedded or not?

- Smartwatch with a GUI?
- Wifi router?
- TV?
- ABS braking system?
- Digital Camera?
- Infotainment System in a car?

Where is embedded development used today?

- Automotive
- Aerospace
- IoT
- Medtech
- Networking

Where there are technology - there will be embedded.

Components

C++ for Developers

Microcontrollers

A small single-chip computer, housing all electronic building blocks to perform its purpose.

Often contain a processor, volatile- and non-volatile memory, communication modules and clock.

Microprocessor

A microprocessor is the single CPU. This is often much more powerful than the MCU’s processor, but the other components are not included inside the chip.

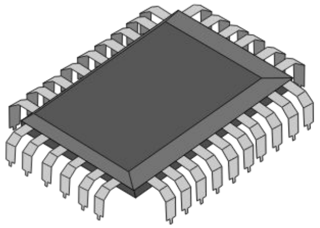
A microprocessor needs to be complemented with more components to run.

What’s the difference?

They both might look the same, but they work differently!

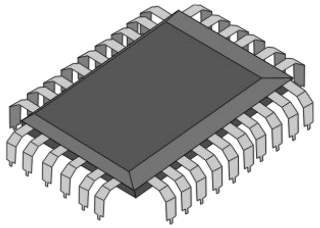
Microcontroller Unit (MCU)

All of the functionality resides within the chip, making it extremely small and cost-efficient.



Microprocessor Unit (MPU)

Microprocessor (a CPU) without any other peripherals. Often more powerful, but also need to be combined with more componenets.



lafftale
for developers

Different types

Microcontrollers (MCU)

- PIC16 / PIC18
- Silicon Labs EFM32
- NXP LPC
- STM32F0/1/4
- ESP32
- STM32H7
- AVR ATmega328P

Microprocessors (MPU)

- Cortex-A
- NXP i.MX6
- TI Sitara
- Broadcom SoCs
- Raspberry Pi SoCs

Embedded system

Microcontroller

Single-chip computer housing all necessary building blocks to control the system.

Memory

Sometimes placed outside depending on system. Often contains the following:

- Flash memory (non-volatile)
- RAM (volatile)
- EEPROM (non-volatile)

Input (Sensors)

Reads physical signals and converts them into electrical signals.

Clock

Dictates the timing of the system. This is needed for synchronization between all the components in the system.

Output (Actuator)

Converts electrical signals into physical movement or effects.

Power Supply

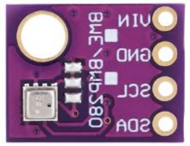
Supplies steady current and voltage to run the system.

Example: Computer Cooling System

Input

Temperature Sensor

Reads temperature inside the computer.

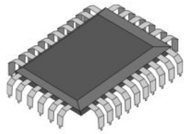


→ Sends the temperature reading to MCU. →

Microcontroller

MCU

Parses the readings from BME280 and executes instructions based on outcome.



→ Sends instruction based on outcome. →

Output

Cooling Fan

Receives instruction from MCU. Can we on / off or fan speed.

How does it work?

C++ for Developers

The “brain”: MCU

The MCU contains the system’s software and the hardware modules needed to perform its internal functions. These modules also enable the MCU to communicate with external components such as sensors and actuators.

Inside the brain: Core Modules

The MCU architecture includes essential hardware required for basic operation. Some of these core modules are:

- Processor
- SRAM
- Flash memory
- Clock system

Inside the brain: Peripherals

Peripherals are hardware modules that extend the functionality of a processor, such as timers, serial interfaces or memory controllers.

What’s the difference?
It’s largely a design choice which components are included as part of the MCU’s core architecture. For example:

The flash memory is typically part of the MCU’s core architecture, but it requires a memory controller to operate. The memory controller is still considered a peripheral — even though it resides within the MCU.

Serial communication

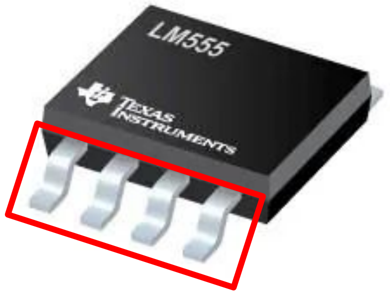
Serial interfaces enable a microcontroller to communicate using serial protocols. This is a lightweight and efficient way for hardware components to exchange data. Some common types include:

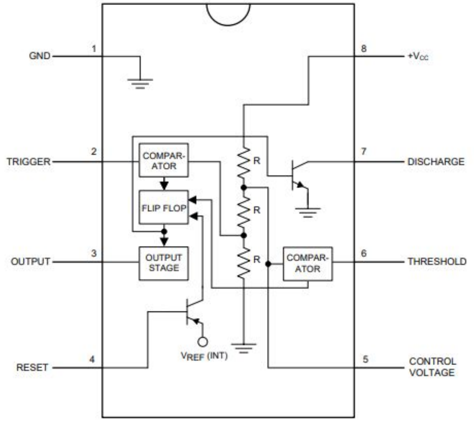
- SPI
- I²C
- UART

These interfaces are used for communication between internal MCU peripherals as well as with external devices such as sensors and actuators.

Pins

Pins serve as connection points to external components like sensors and actuators. Each pin can have a specific function, such as communication, input/output, or providing power and ground.





Registers

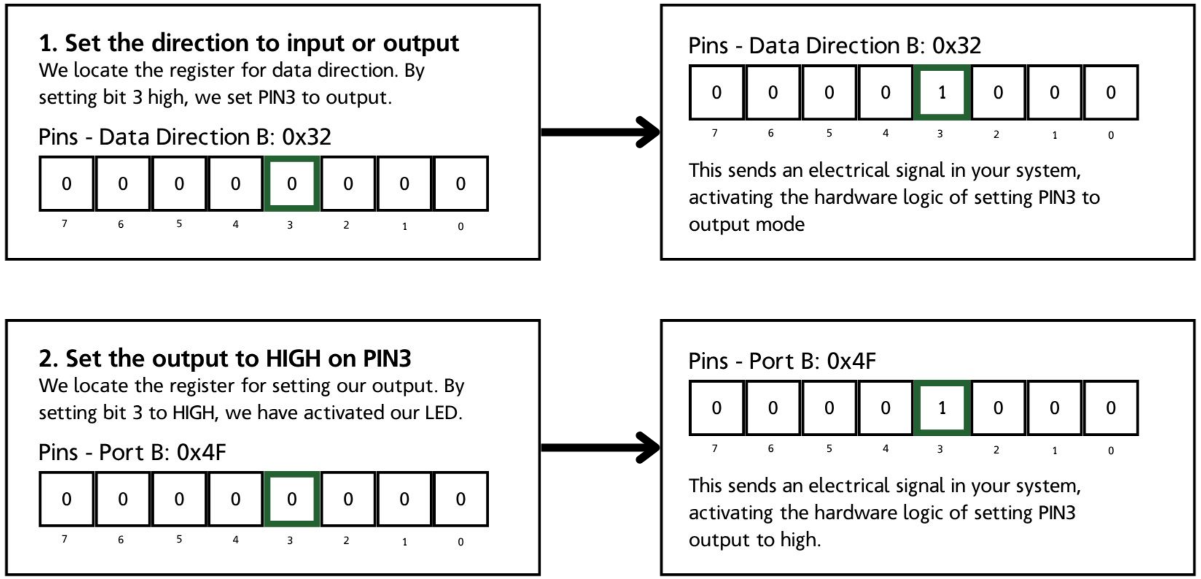
Registers form the interface between software and the MCU's hardware. By setting or clearing bits within a register, the software can enable, disable or configure specific hardware functions inside the MCU.

- **Address:** The memory location where the register resides and used by software to access it.

- **Bits:** Individual binary flags within the register that control or indicate specific hardware functionality.

Example: Turning on a LED

Goal: Turn on an LED Connected to PIN3



1. Set the direction to input or output
We locate the register for data direction. By setting bit 3 high, we set PIN3 to output.

Pins - Data Direction B: 0x32

0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0

Pins - Data Direction B: 0x32

0	0	0	0	1	0	0	0
7	6	5	4	3	2	1	0

This sends an electrical signal in your system, activating the hardware logic of setting PIN3 to output mode

2. Set the output to HIGH on PIN3
We locate the register for setting our output. By setting bit 3 to HIGH, we have activated our LED.

Pins - Port B: 0x4F

0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0

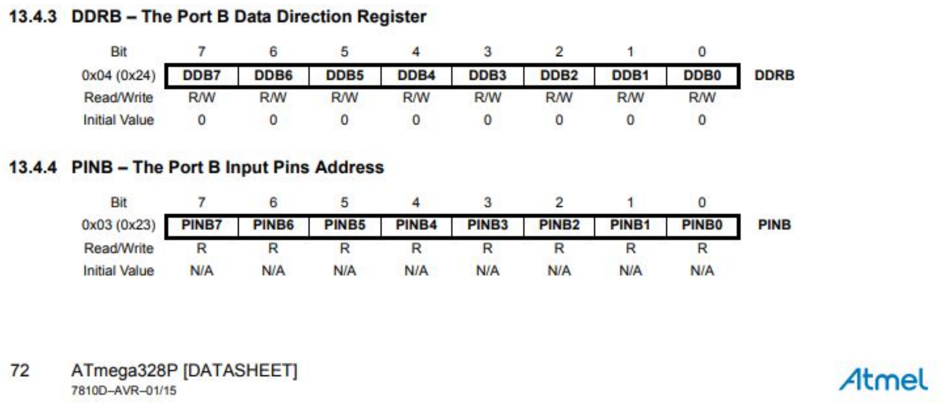
Pins - Port B: 0x4F

0	0	0	0	1	0	0	0
7	6	5	4	3	2	1	0

This sends an electrical signal in your system, activating the hardware logic of setting PIN3 output to high.

Datasheets

Detailed description of the MCU's hardware. Here we find the information needed to write firmware.



13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.4 PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

72 ATmega328P [DATASHEET]
7810D–AVR–01/15

Atmel

Firmware

Software that runs on our embedded system.

This includes several layers like:

- Application logic
- Hardware Abstraction Layer
- Drivers

Driver

Software written to enable interfacing between different components.

When we want two embedded systems to communicate with each other, we write drivers that defines how to communicate.

Essentially we are creating an API, which the unit that wants to communicate can call.

Interrupts

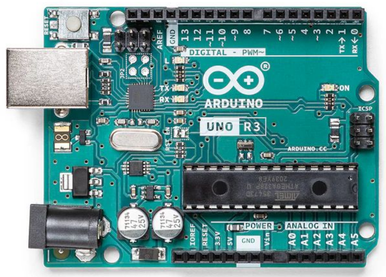
Interrupts are hardware signals that temporarily pause the processor’s current task so it can handle a more urgent event.

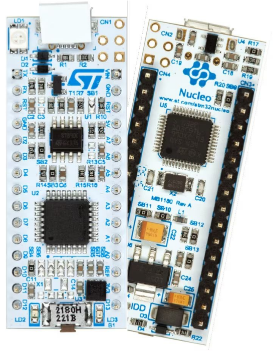
When an interrupt occurs, the processor saves its current context and jumps to a special function called an Interrupt Service Routine (ISR). The ISR quickly handles the event (such as reading a sensor) before returning control to the main program.

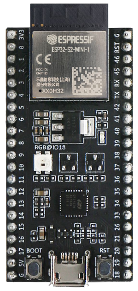
ISRs are typically short and efficient to minimize disruption to normal program execution.

Development Board / Kit

A programmable board used to develop and test firmware. It typically includes an on-board programmer/debugger, allowing the developer to flash code to the microcontroller via USB without needing external tools.







IC (Integrated Circuit)

An Integrated Circuit (IC) is a tiny chip that performs a designated function by containing all the necessary electronic components integrated together into a single piece of material.

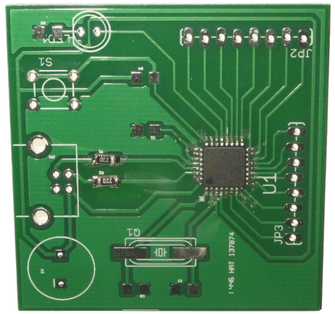
A microcontroller is an integrated circuit, but it can also be something with more specific functions such as a clock.

lafftale
for developers

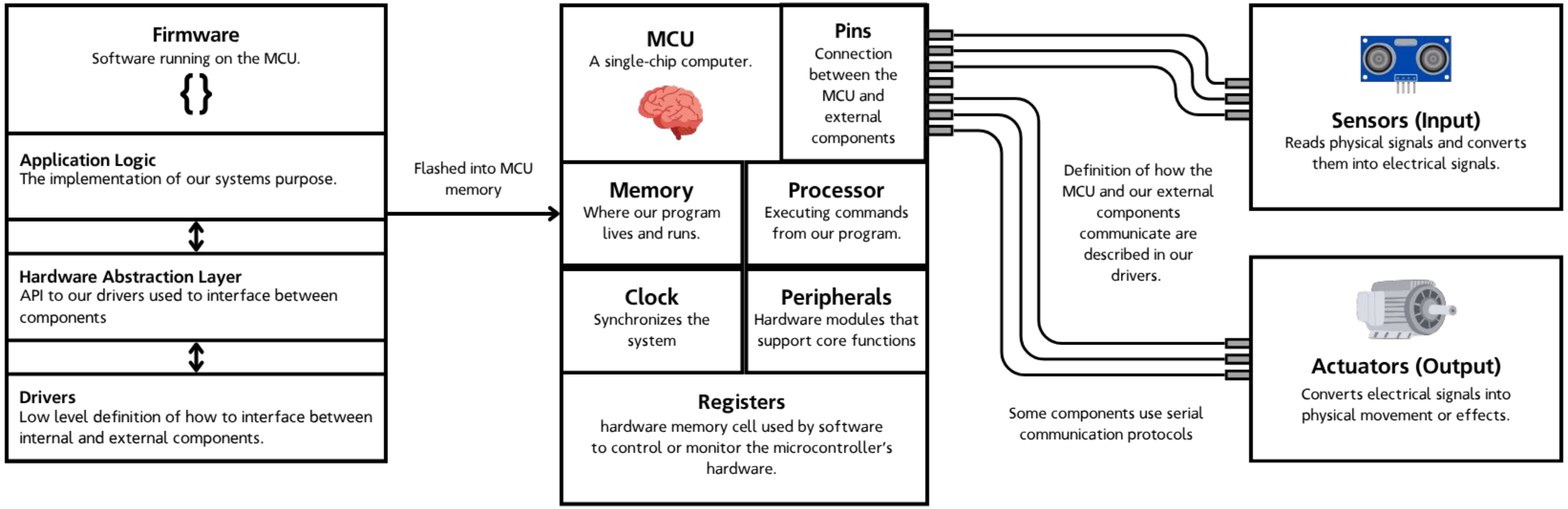
Printed Circuit Board (PCB)

A Printed Circuit Board (PCB) is a board that provides both physical support and electrical connections for electronic components.

You can think of it as a city map, where the roads (copper traces) connect buildings (components) so they can communicate and work together.



Summary

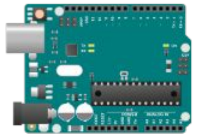


Firmware
Software running on the MCU.
{}
Application Logic
The implementation of our systems purpose.
Hardware Abstraction Layer
API to our drivers used to interface between components
Drivers
Low level definition of how to interface between internal and external components.
Flashed into MCU memory
MCU
A single-chip computer.
Pins
Connection between the MCU and external components
Memory
Where our program lives and runs.
Processor
Executing commands from our program.
Clock
Synchronizes the system
Peripherals
Hardware modules that support core functions
Registers
hardware memory cell used by software to control or monitor the microcontroller's hardware.
Definition of how the MCU and our external components communicate are described in our drivers.
Some components use serial communication protocols
Sensors (Input)
Reads physical signals and converts them into electrical signals.
Actuators (Output)
Converts electrical signals into physical movement or effects.

Summary

Development Kit

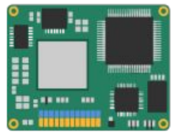
A devkit is a programmable board used to develop and test firmware, with an on-board programmer that allows flashing via USB.



During development stage

Printed Circuit Bord Assembly

A PCBA is the finished circuit board built and programmed for our system, ready to go into the product.



In production

Embedded Systems

C++ for Developers

lafftale
for developers

Bare Metal

Bare metal programming means running code directly on the microcontroller’s hardware without an operating system.

It typically follows a simple structure of declaration, initialization, and a main “super loop” that repeatedly executes the application logic.

Often used in narrow, lightweight embedded systems which don’t require scheduling or high real-time requirements.

Real-Time Operating Systems

An RTOS manages tasks on a microcontroller, allowing multiple functions to run seemingly in parallel.

It provides scheduling, timing, and synchronization, making it easier to build reliable and responsive embedded applications.

Some examples are: Zephyr or FreeRTOS.

General-Purpose Operating System

A General-Purpose Operating System (GPOS) is a full-featured OS designed to handle complex applications, multitasking, and user interfaces.

In embedded systems, a GPOS (often Linux-based) may be used on more capable hardware that can manage file systems, networking, and even GUIs.

Examples include embedded Linux distributions built using projects like Yocto or Buildroot.

Bare Metal Programming

C++ for Developers

AVR ATMega328p

During this course we will write firmware for the ATMega328p MCU. Best recognized from the Arduino Uno Development Board.

For this we need to download the library for AVR microcontrollers!

lafftale
for developers