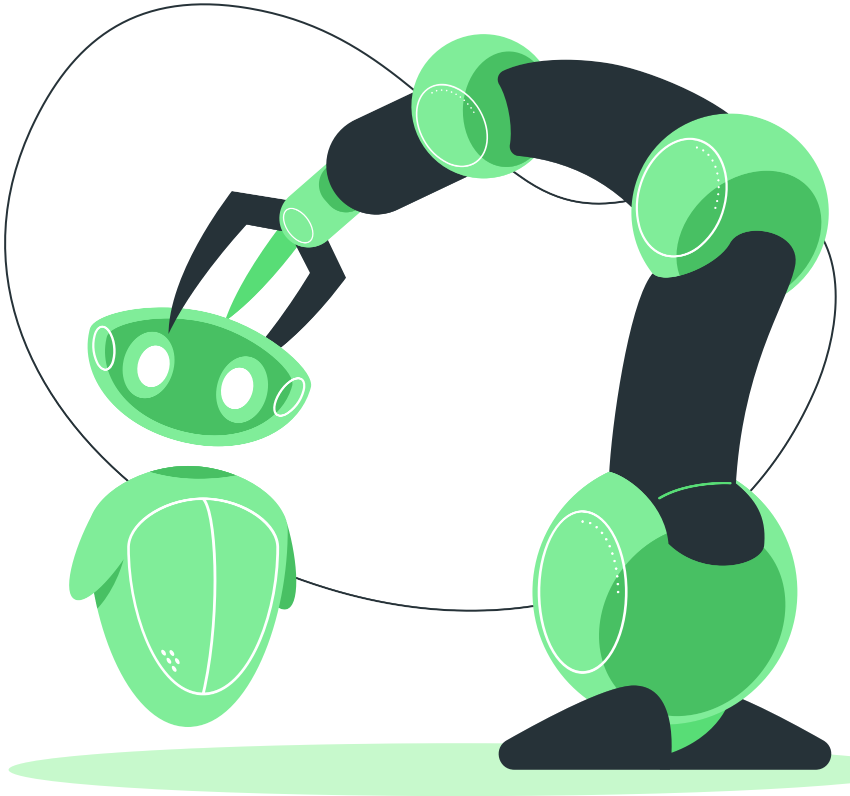
Freeways

free software club

STM32 Workshop

By Moktar SELLAMI



Plan

1 ***Why you should be here ?***

2 ***Intro to embedded systems***

3 ***Microcontroller***

4 ***Motherboard VS Microcontroller***

5 ***STM32***

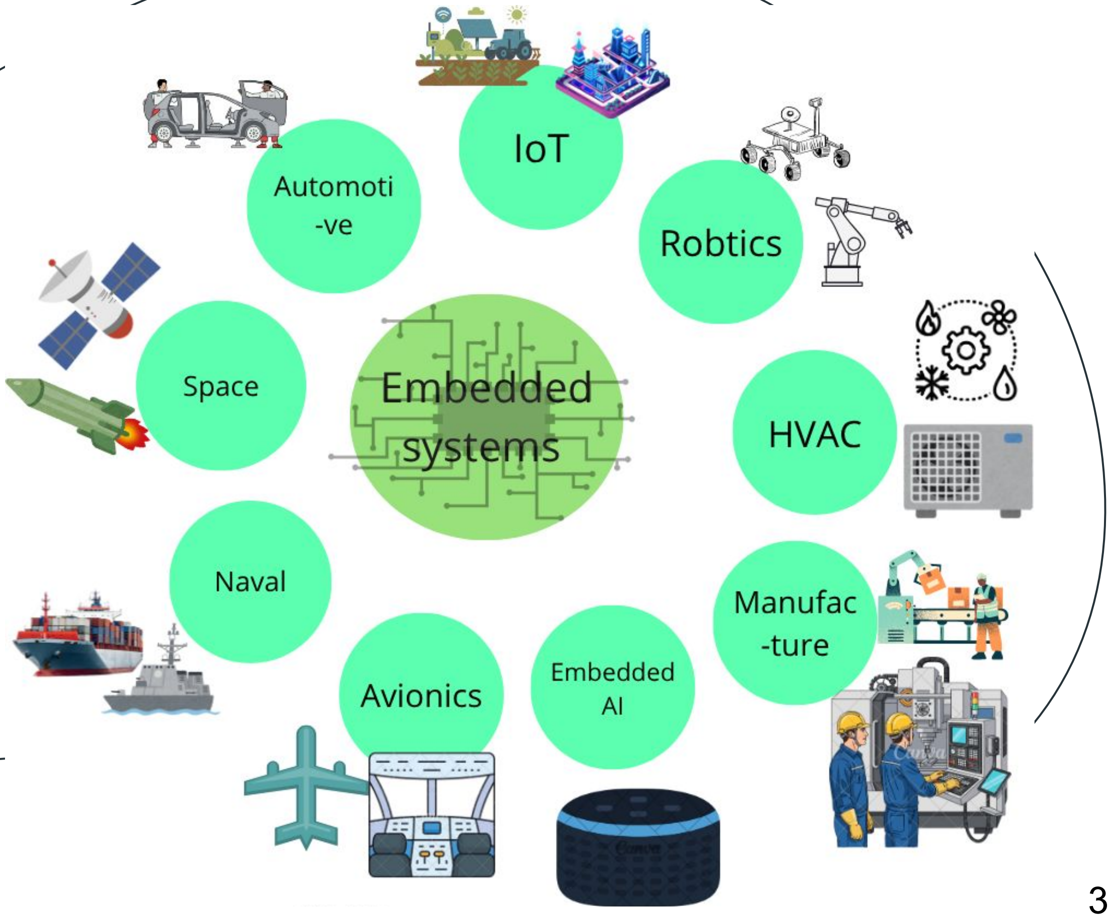
6 ***Let's do something***

7 ***STM32 GPIO***

8 ***Writing the Software: HAL***

Why you should be here ?

The field Embedded systems:



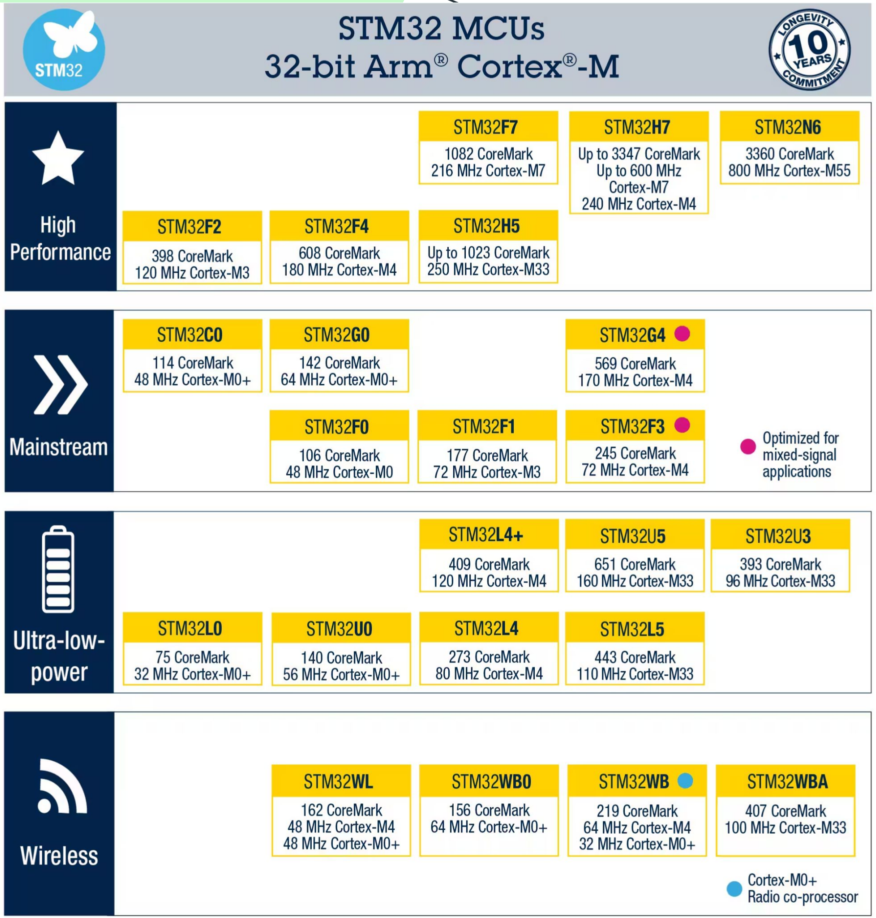
Why you should be here ?

Why STM32: (For MCU)

1. Unmatched Scalability & Portfolio
 (+ 1500 MCU variants)
2. Powerful & Comprehensive Ecosystem
3. Leadership in Performance & Power Efficiency

Why STM32: (For Students)

1. A way to get an internship
2. A bridge from Arduino
3. Provides an ecosystem for newbies (CubeMX)
4. Community



STM32 MCUs
32-bit Arm® Cortex®-M

High Performance

STM32F7
1082 CoreMark
216 MHz Cortex-M7

STM32H7
Up to 3347 CoreMark
Up to 600 MHz
Cortex-M7
240 MHz Cortex-M4

STM32N6
3360 CoreMark
800 MHz Cortex-M55

STM32F2
398 CoreMark
120 MHz Cortex-M3

STM32F4
608 CoreMark
180 MHz Cortex-M4

STM32H5
Up to 1023 CoreMark
250 MHz Cortex-M33

Mainstream

STM32C0
114 CoreMark
48 MHz Cortex-M0+

STM32G0
142 CoreMark
64 MHz Cortex-M0+

STM32G4 ●
569 CoreMark
170 MHz Cortex-M4

STM32F0
106 CoreMark
48 MHz Cortex-M0

STM32F1
177 CoreMark
72 MHz Cortex-M3

STM32F3 ●
245 CoreMark
72 MHz Cortex-M4

● Optimized for mixed-signal applications

Ultra-low-power

STM32L4+
409 CoreMark
120 MHz Cortex-M4

STM32U5
651 CoreMark
160 MHz Cortex-M33

STM32U3
393 CoreMark
96 MHz Cortex-M33

STM32L0
75 CoreMark
32 MHz Cortex-M0+

STM32U0
140 CoreMark
56 MHz Cortex-M0+

STM32L4
273 CoreMark
80 MHz Cortex-M4

STM32L5
443 CoreMark
110 MHz Cortex-M33

Wireless

STM32WL
162 CoreMark
48 MHz Cortex-M4
48 MHz Cortex-M0+

STM32WB0
156 CoreMark
64 MHz Cortex-M0+

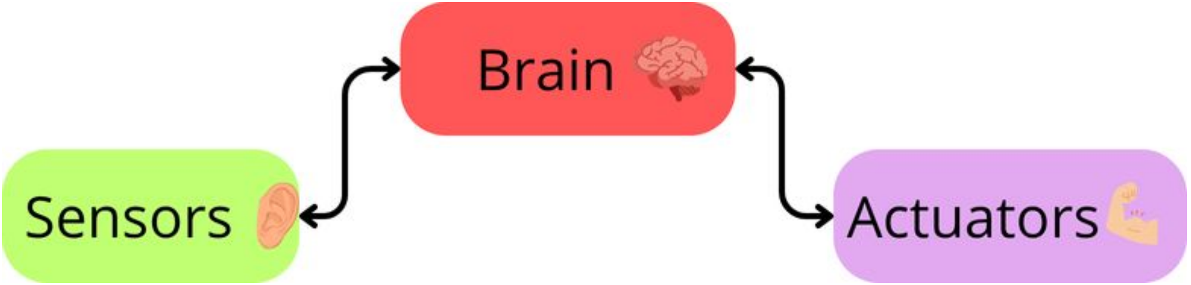
STM32WB ●
219 CoreMark
64 MHz Cortex-M4
32 MHz Cortex-M0+

STM32WBA
407 CoreMark
100 MHz Cortex-M33

● Cortex-M0+ Radio co-processor

Introduction to embedded systems

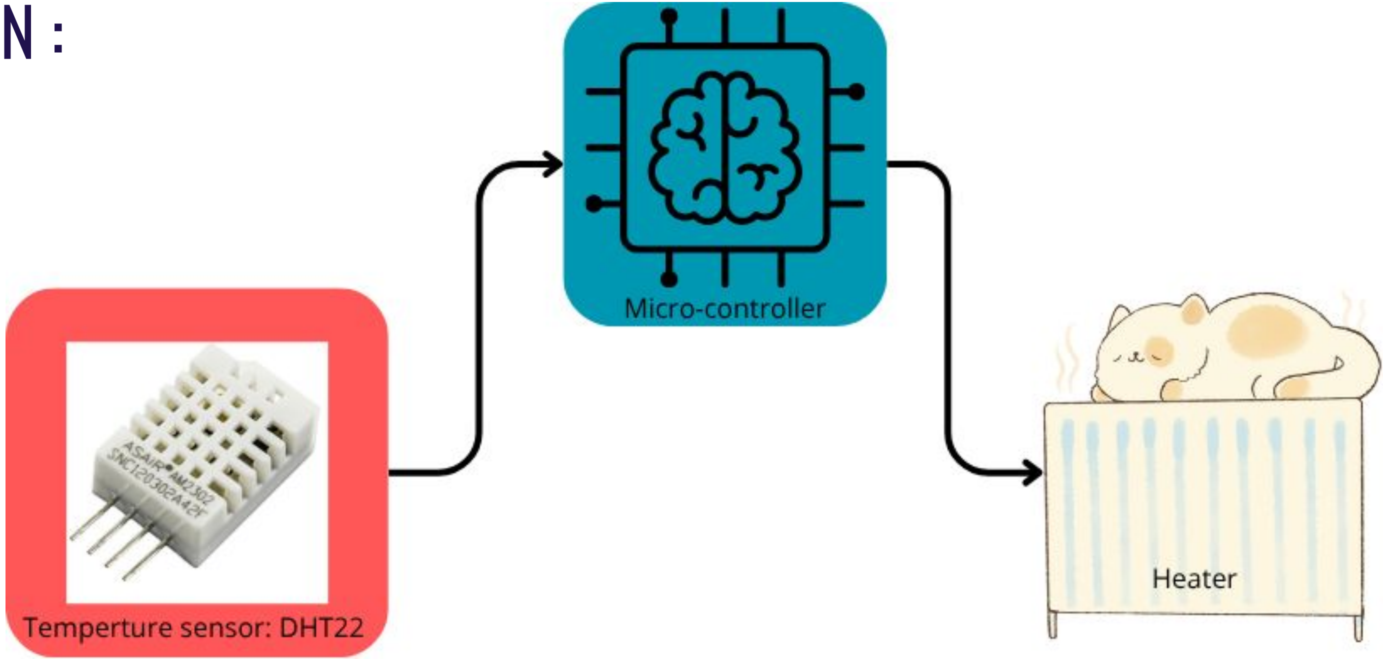
Embedded systems are specialized computing systems designed to perform specific functions within larger systems.



Thermostat

Introduction to embedded systems

THE BRAIN :



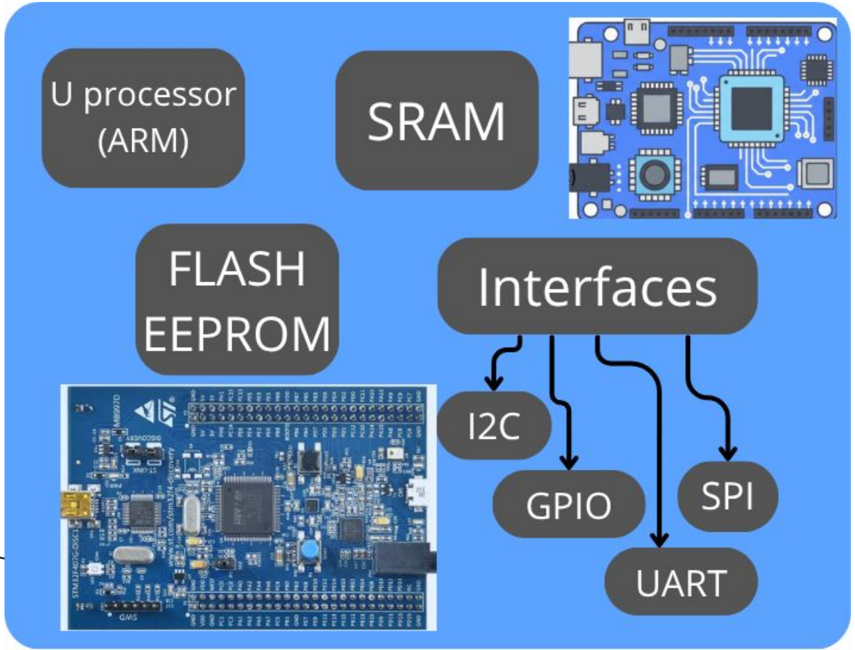
Microcontroller

A microcontroller is a compact integrated circuit that integrates many components.
It is categorized with its limit resources:

- Low processing power : 12 Mhz to 700 Mhz
- Low memory capacity: 2KB to 1MB
- Low storage capacity: 4KB to 20MB
- Energy consumption: 10 mW to 2W

Analogy between motherboard and Microcontroller

Microcontroller



U processor (ARM)

SRAM

FLASH EEPROM

Interfaces

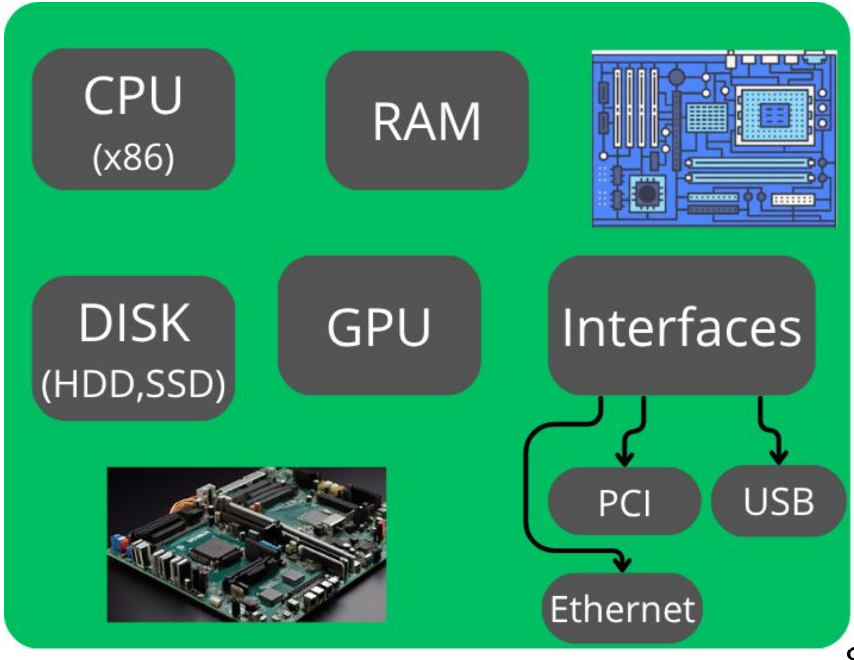
I2C

GPIO

UART

SPI

Motherboard



CPU (x86)

RAM

DISK (HDD,SSD)

GPU

Interfaces

PCI

USB

Ethernet

STM32

What is STM32?

Family of 32-bit microcontrollers by STMicroelectronics

Use ARM Cortex-M cores (M0, M0+, M3, M4, M7, M33)

Launched in 2007 with F1 series

STMicroelectronics

Largest semiconductor company in Europe

Founded in 1987 (France + Italy merger)

Headquarters: Geneva, Switzerland

49,602 employees, $13.27B revenue (2024)

Families

Mainstream:	C0/G0/G4/F0/F1/F3
High Performance:	H7/H5/F7/F4/F2
Low Power:	L0/L4/L5/U0/U3/U5
Wireless:	WL/WB0/WB/WA
AI:	N6

Applications & Fields

Industrial automation (PLCs, robots, HMIs)

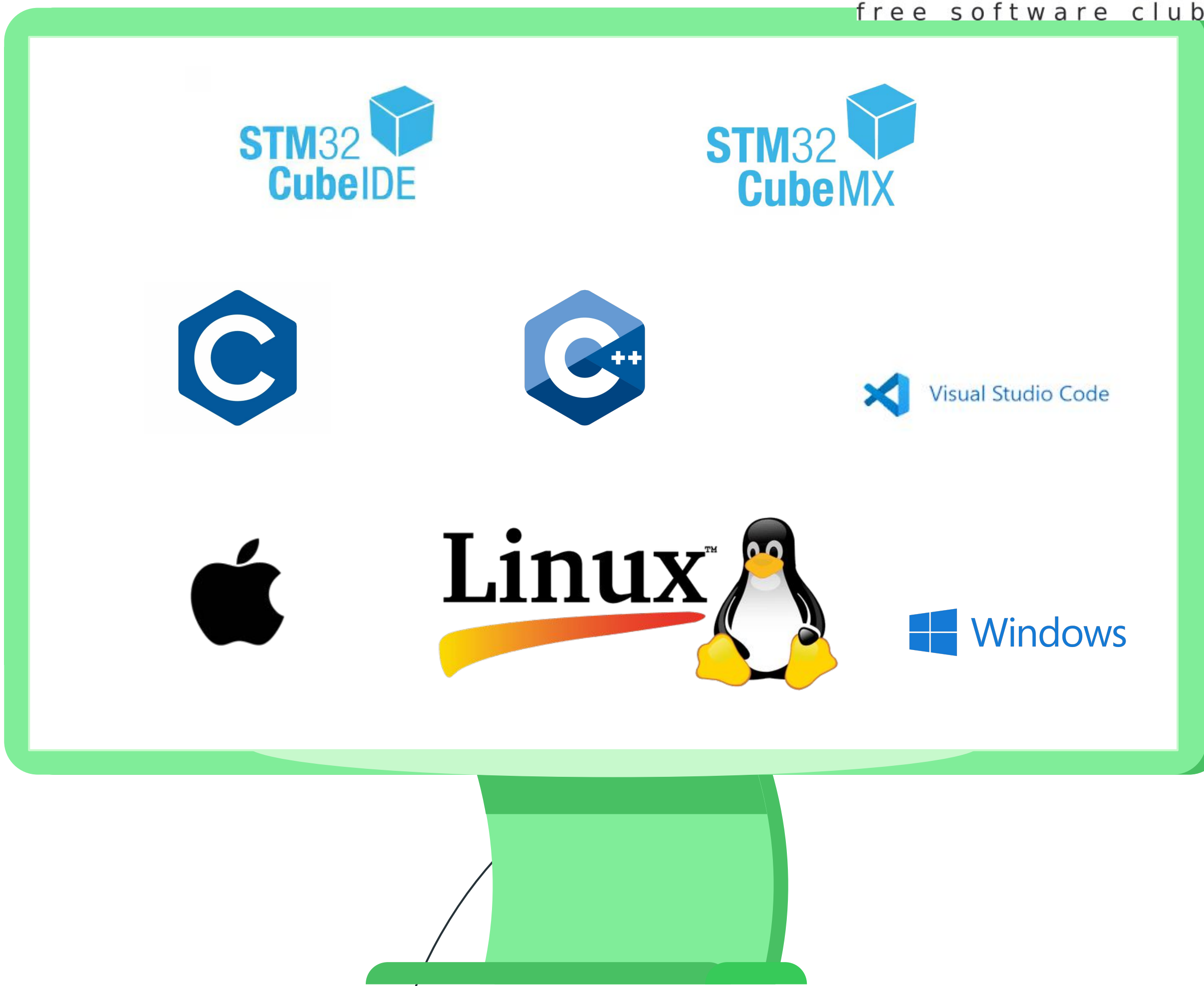
Consumer electronics (smart devices, wearables)

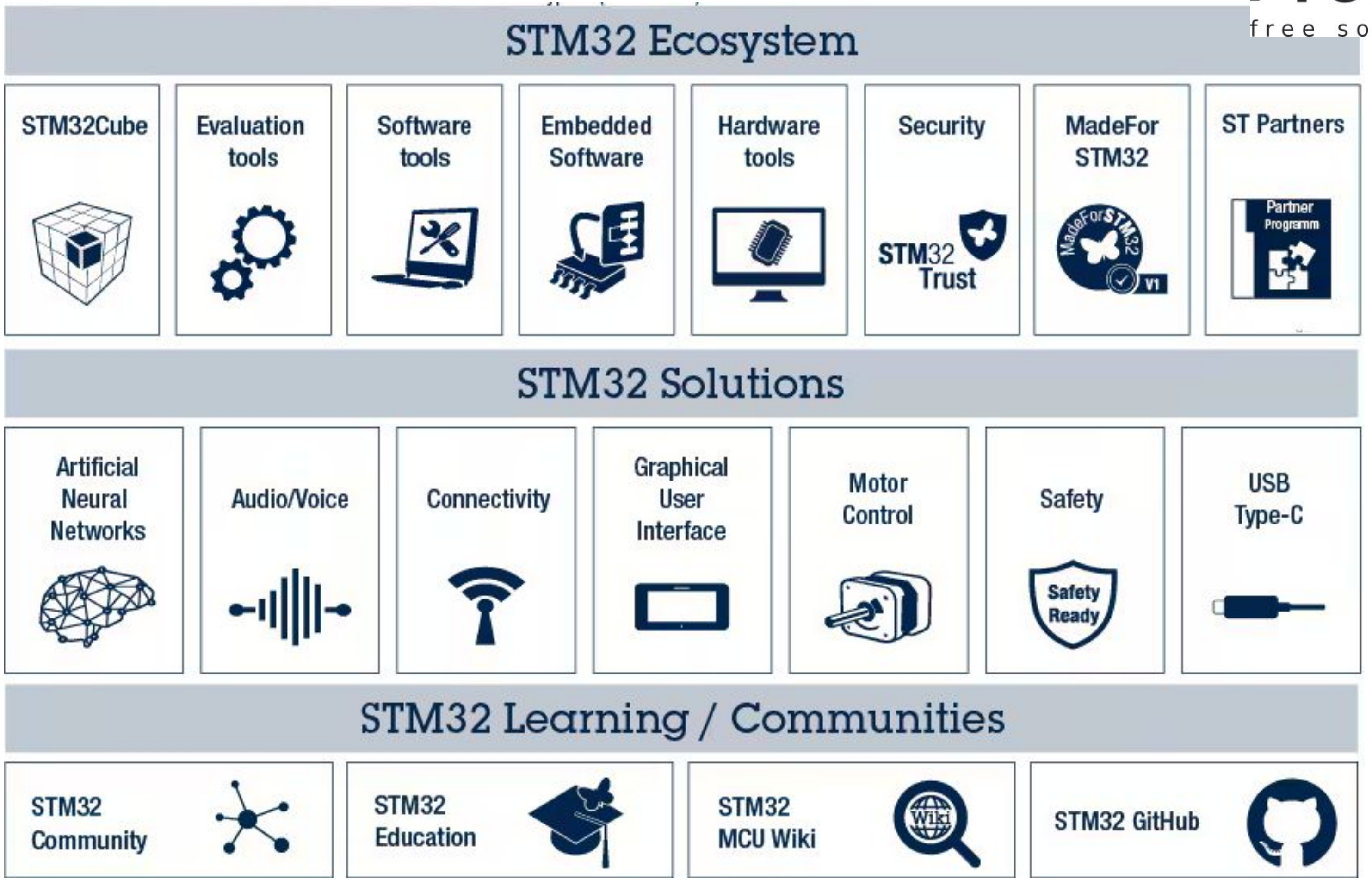
Internet of Things (IoT)

Medical equipment

Automotive systems

STM32 Ecosystem:





STM32 Ecosystem

STM32Cube | Evaluation tools | Software tools | Embedded Software | Hardware tools | Security | MadeFor STM32 | ST Partners

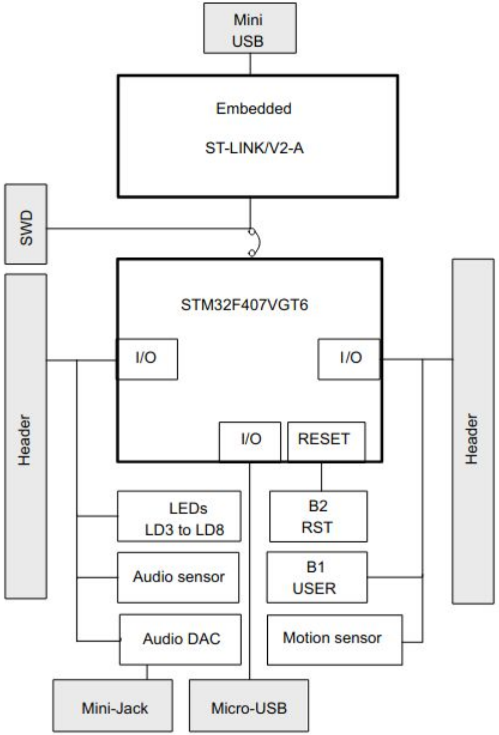
STM32 Solutions

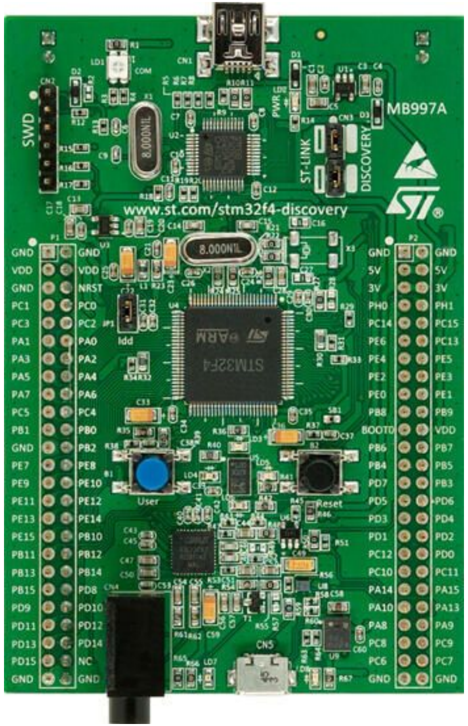
Artificial Neural Networks | Audio/Voice | Connectivity | Graphical User Interface | Motor Control | Safety | USB Type-C

STM32 Learning / Communities

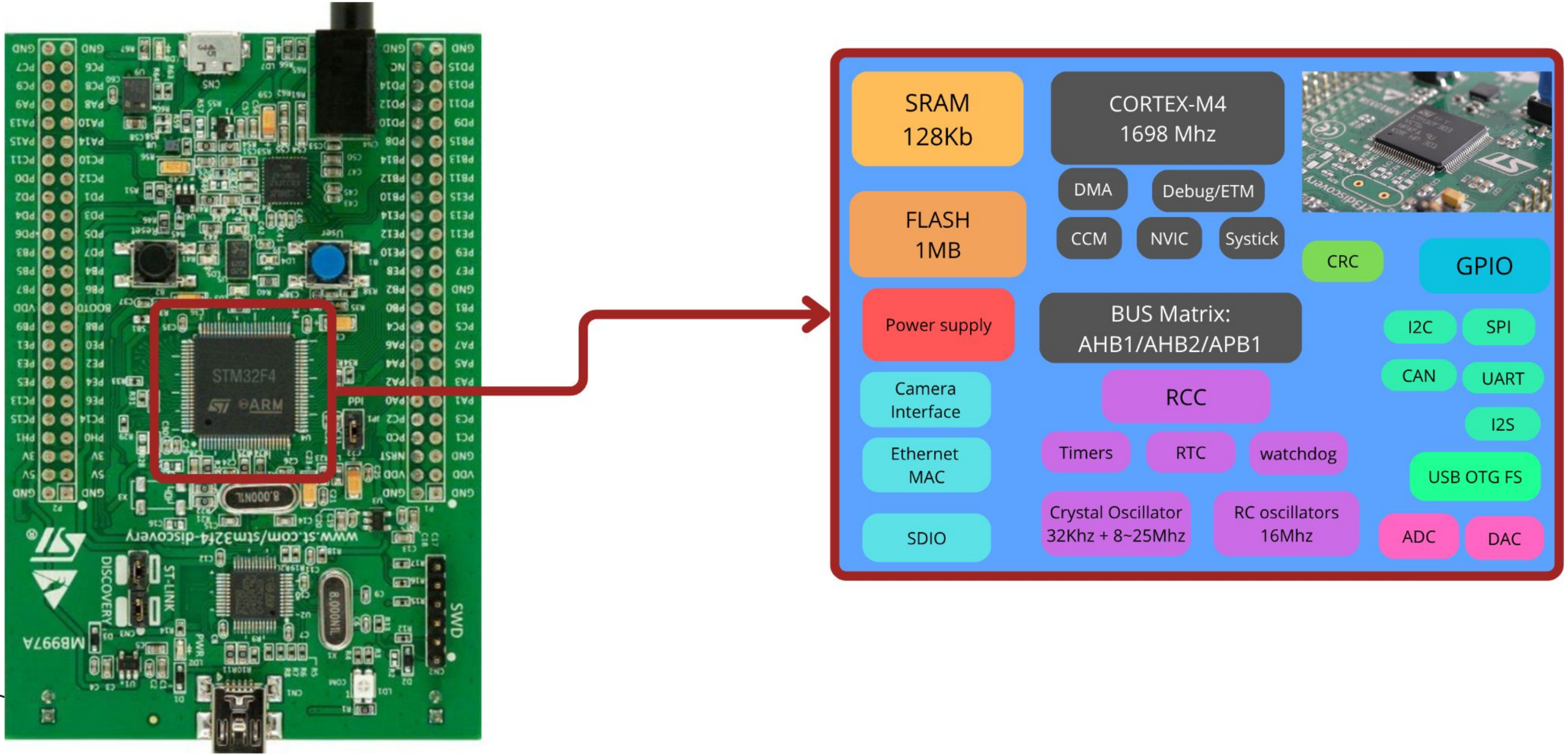
STM32 Community | STM32 Education | STM32 MCU Wiki | STM32 GitHub

Overview: STM32F407-DISCO1





Overview: STM32F407 Microcontroller



SRAM
128Kb

CORTEX-M4
1698 Mhz

DMA

Debug/ETM

FLASH
1MB

CCM

NVIC

Systick

CRC

GPIO

Power supply

BUS Matrix:
AHB1/AHB2/APB1

I2C

SPI

CAN

UART

Camera
Interface

RCC

I2S

Ethernet
MAC

Timers

RTC

watchdog

USB OTG FS

SDIO

Crystal Oscillator
32Khz + 8~25Mhz

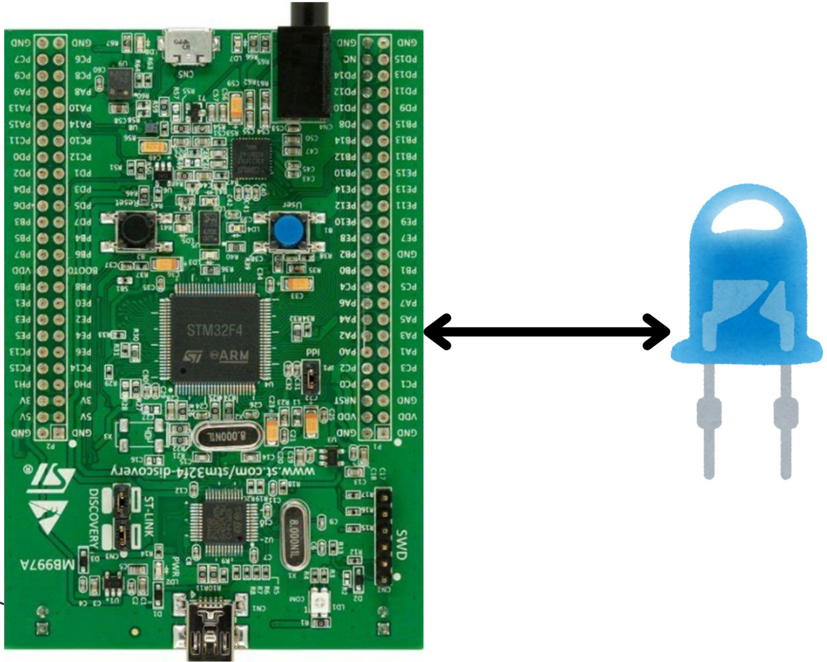
RC oscillators
16Mhz

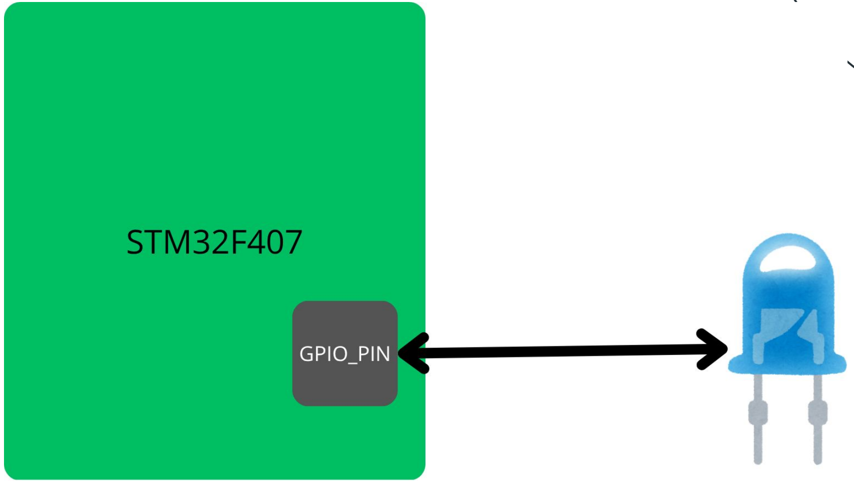
ADC

DAC

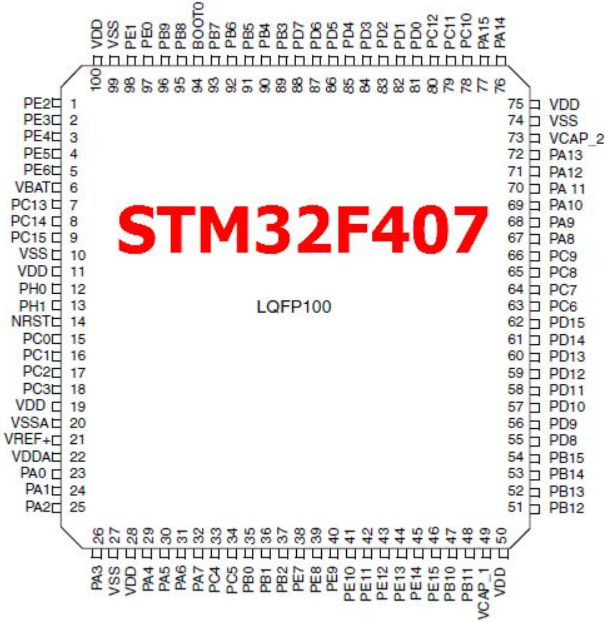
Let’s Do something :

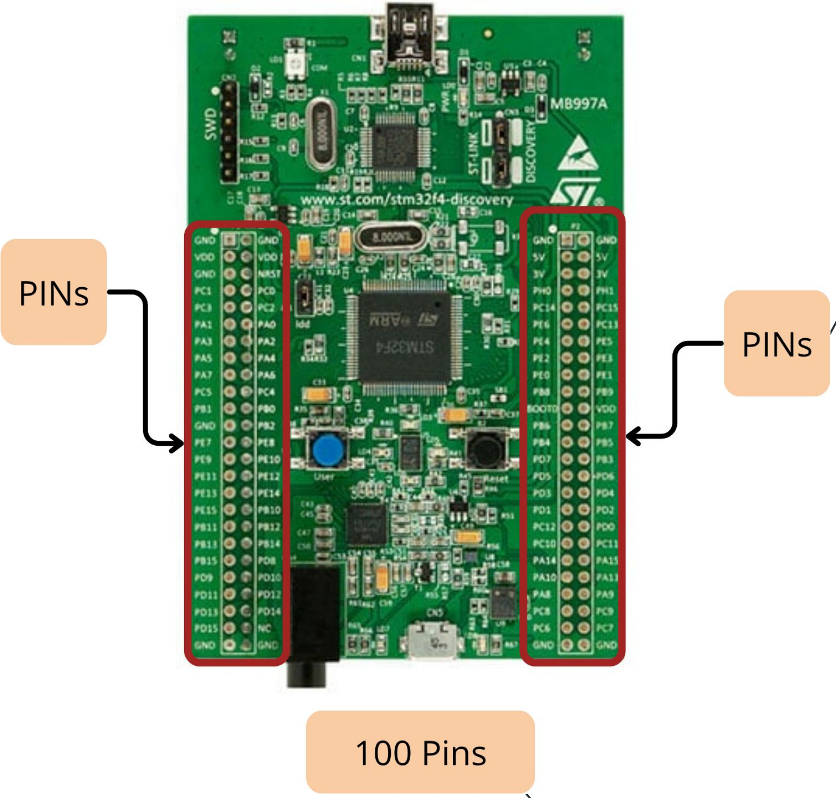
Case Study: Toggling an LED





STM32 GPIO : General purpose Input Output





PINs

PINs

100 Pins

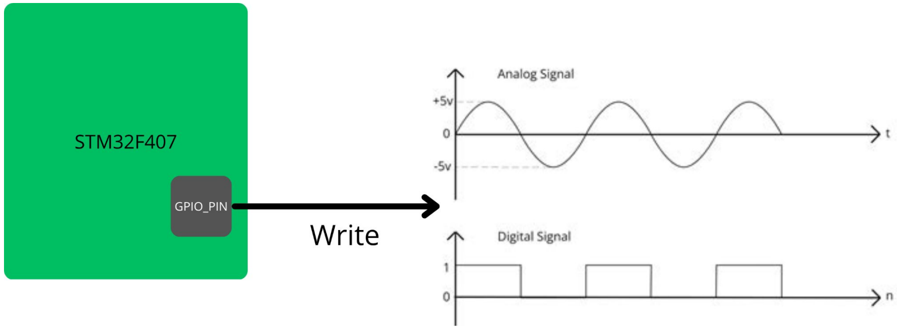
STM32 GPIO : General purpose Input Output

GPIO stands for General Purpose Input/Output.

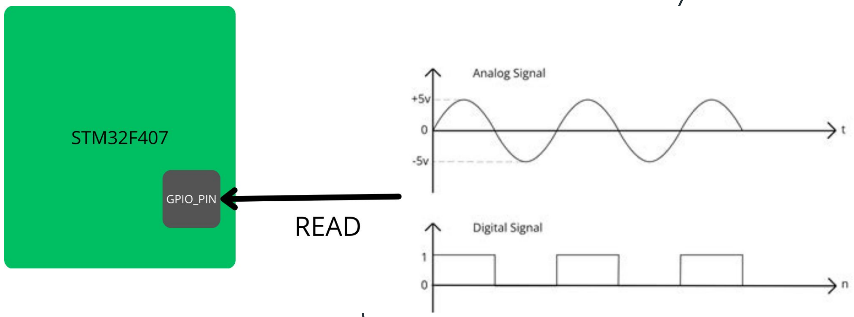
It’s the most basic and versatile feature of a microcontroller, the way it interacts with the outside world.

You can think of a GPIO pin as a configurable electrical pin on the chip that can either:

Send a signal to outside (as an output)



Read a signal coming from outside (as an input



STM32 GPIO : GPIO Modes

The STM32 groups the GPIOS in to clusters called PORTs indicated by GPIOx .
By x we mean: GPIOA to GPIOI .
Each port has 16 pins: From 0 to 15
For example:

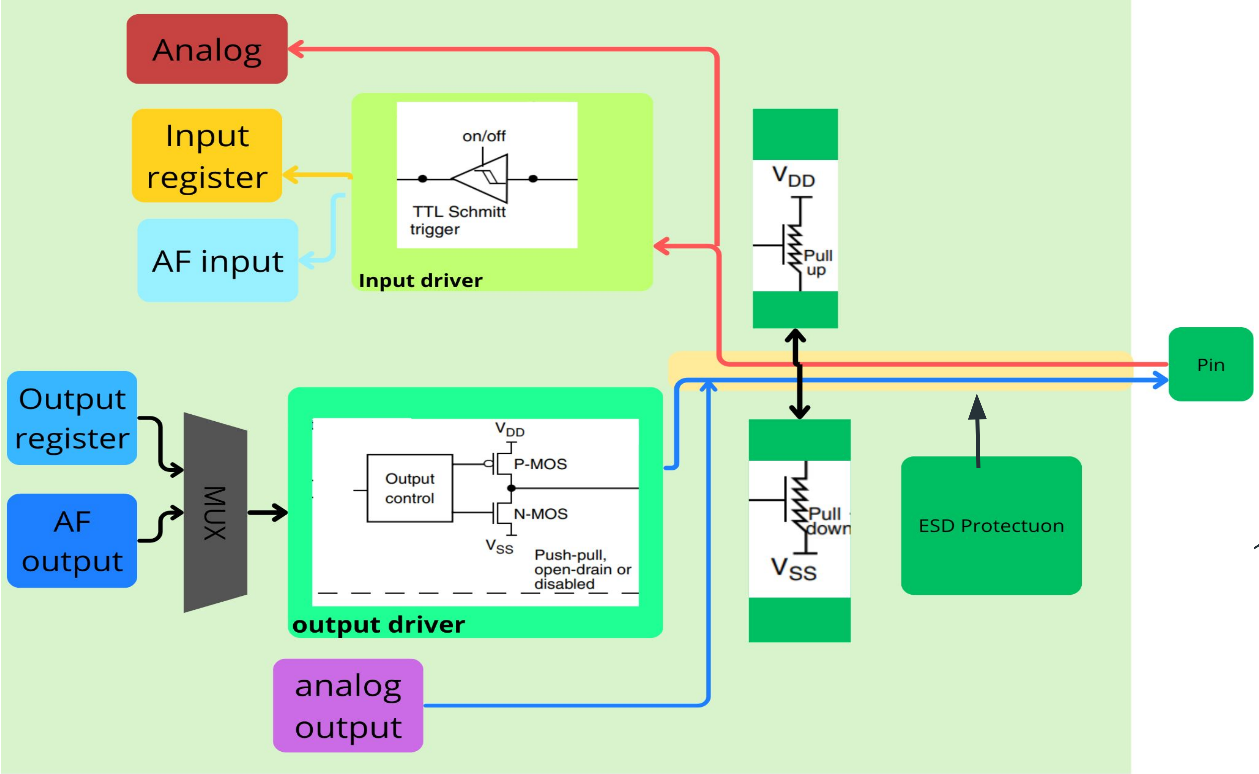
- the internal Green LED PD12: GPIOD pin 12
- The internal BTN PA0: GPIOA pin 0

The GPIO has 4 Modes:

- Input
- Output
- Alternate function
- analog

Freeways
free software club

STM32 GPIO : GPIO Structure

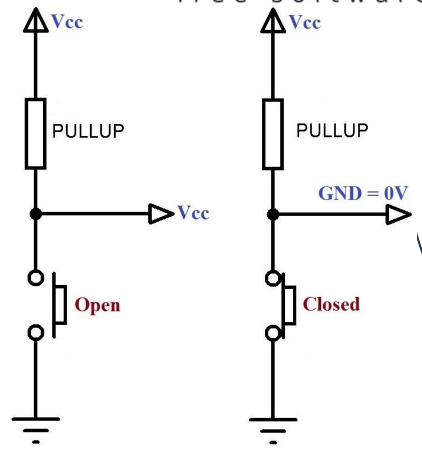


STM32 GPIO : GPIO Modes

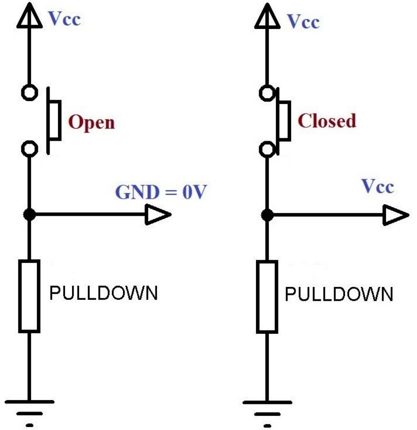
Input Mode:

NoPull: Floating input
PullDown: The input is set to logic low (0)
PullUp: The input is set to logic High (1)

PullUP



PullDown

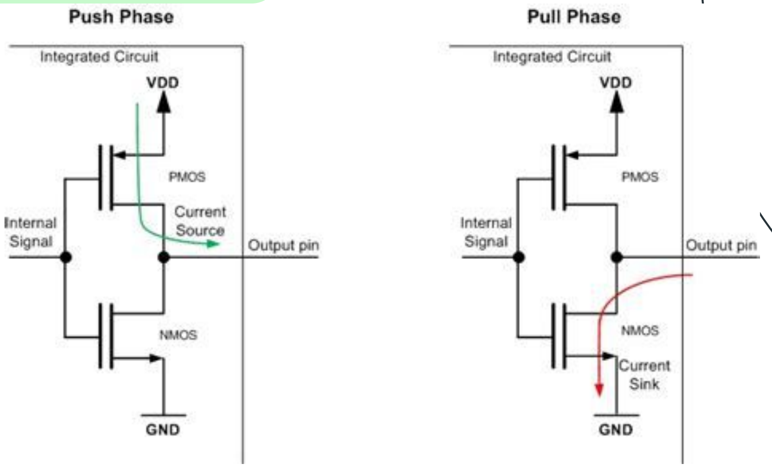


Freeways
free software club

STM32 GPIO : GPIO Modes

Output Mode:

PushPull: Drives the pin to output a steady and stable voltage (3.3v).
Used for devices that doesn’t exceed the maximum output voltage that the stm32 can support
Example: Powering and LED, sensor



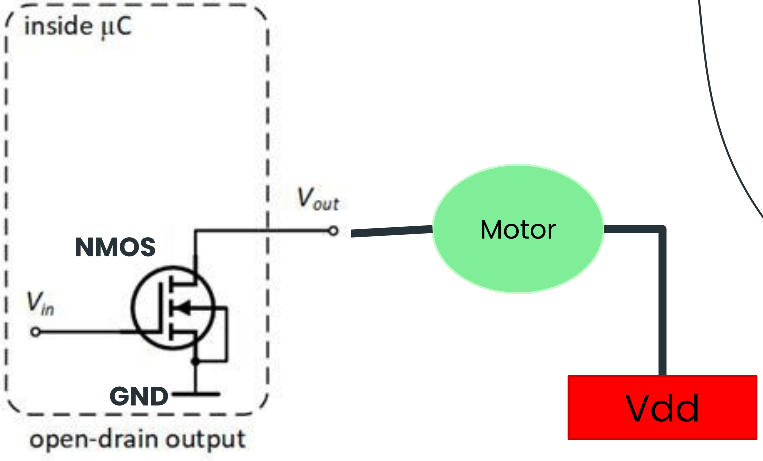
- It contains 2 transistors: **PMOS** and **NMOS**, They both act as switches letting current in and out.
- The **PMOS** is **ON** when he gets a **LOGIC 0,** hence he lets current pass form **Vdd** to the **output pin (3.3V).**
- The **NMOS** is **ON** when he gets **LOGIC 1**, hence connecting the **GND** the the **output pin (0V)**.

NOTE: Only one transistor works at a time, so when the **PMOS** is **ON** the **NMOS** is **OFF** and vice versa.

STM32 GPIO : GPIO Modes

Output Mode:

OpenDrain: Drives the pin Low (0V).
Used when there is a external energy Source (**Vdd**)
Example: Motor control, or controlling and LED or sensors that needs higher than (3.3V) to work.

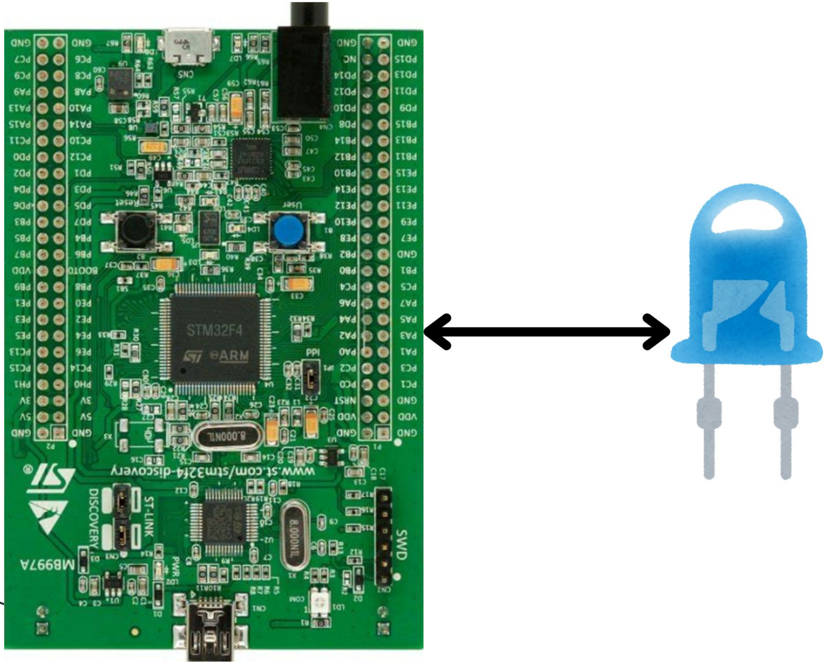


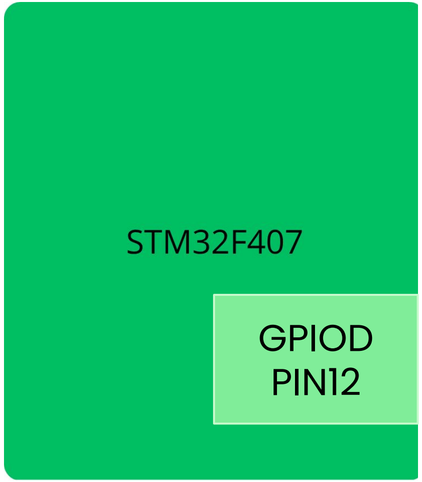
open-drain output

- It contains 1 transistors: **NMOS**:
- The **NMOS** is **ON** when he gets **LOGIC 0**, hence connecting the **GND** the the **output pin (0V)**, the circuit closes and the motor spins.
- **LOGIC 1** closes the circuit and the **NMOS** is **OFF**.

Programming:

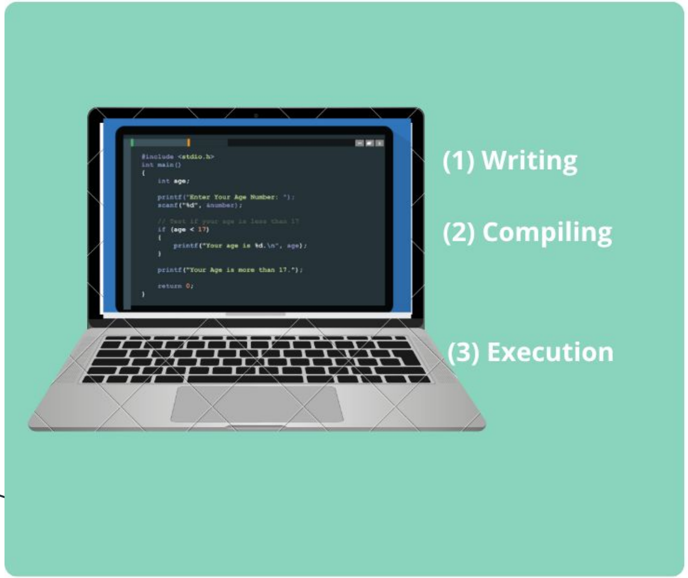
Toggling an LED connected in PD12





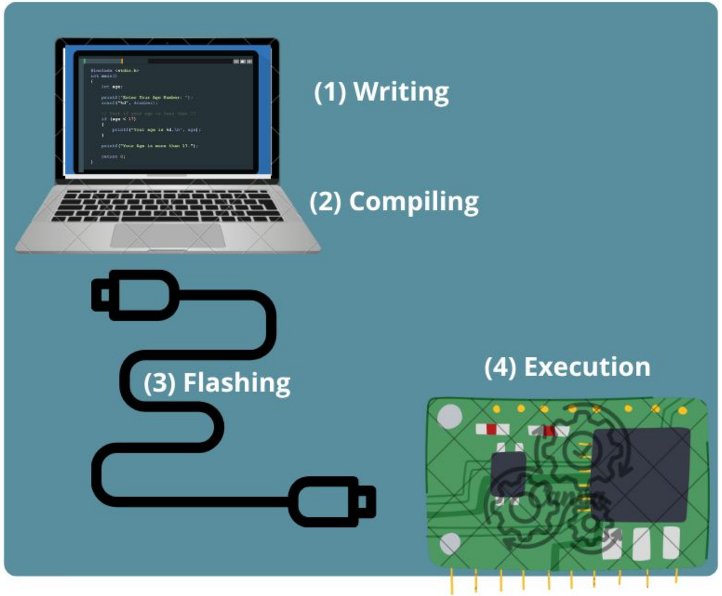
Programming:

Software Engineer



```
#include <stdio.h>
int main()
{
    int age;

    printf("Enter Your Age Number: ");
    scanf("%d", &number);

    // Test if your age is less than 17
    if (age < 17)
    {
        printf("Your age is %d.\n", age);
    }

    printf("Your Age is more than 17.");

    return 0;
}
```

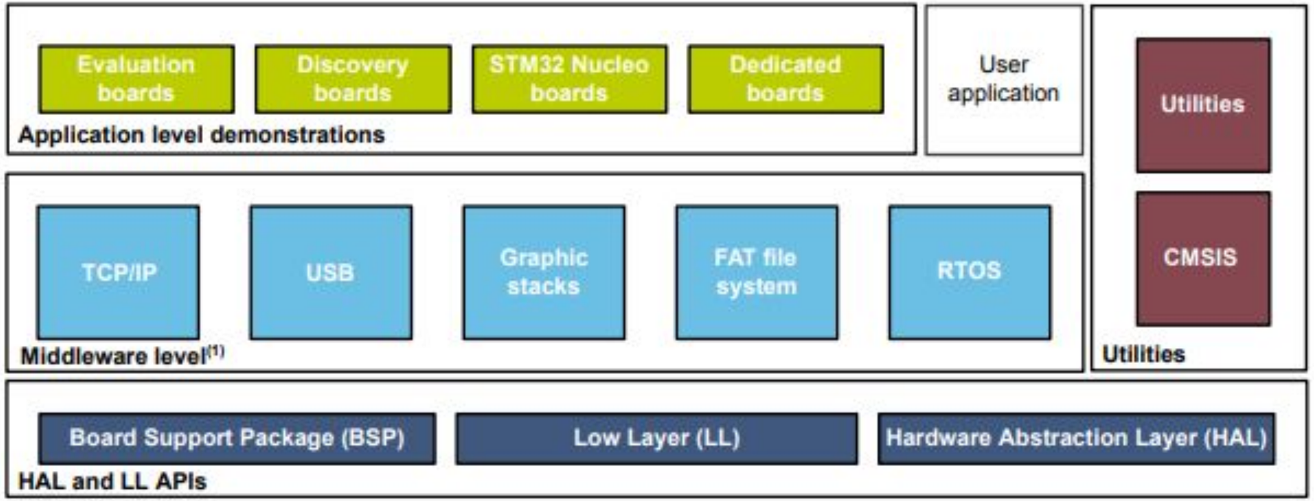
Embedded systems Engineer



```
#include <stdio.h>
int main()
{
    int age;

    printf("Enter Your Age Number: ");
    scanf("%d", &number);

    // Test if your age is less than 17
    if (age < 17)
    {
        printf("Your age is %d.\n", age);
    }

    printf("Your Age is more than 17.");

    return 0;
}
```

Writing the Software: HAL

STM32CubeF4 package: Github Link



Writing the Software: HAL

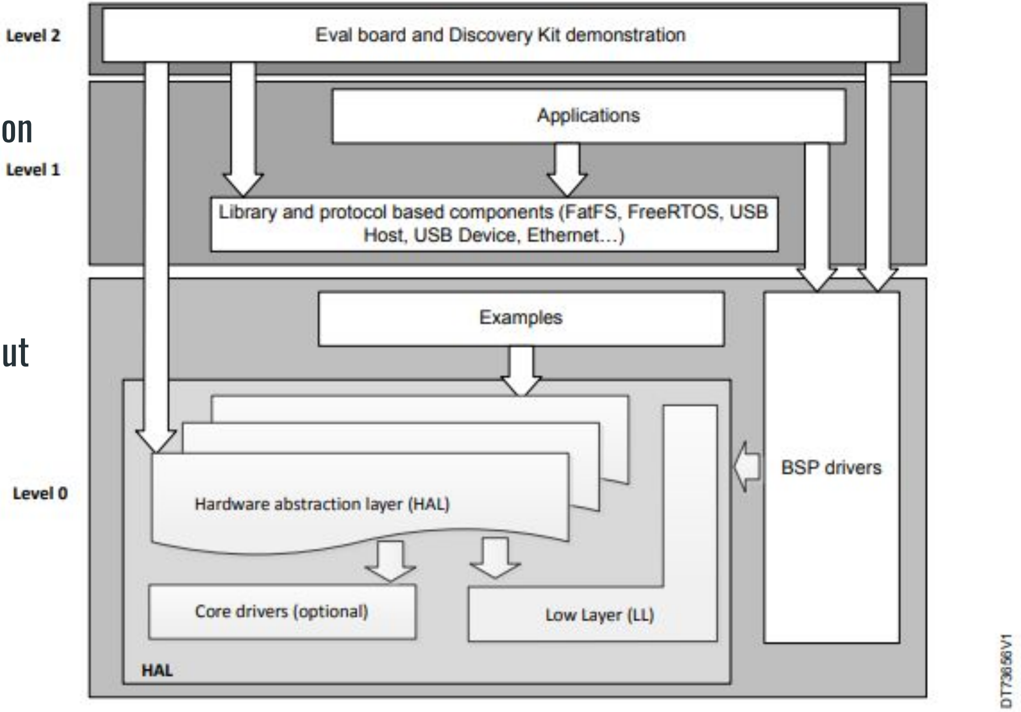
Level 0: Hardware Interaction
This is the foundation that talks directly to the MCU and board components.

BSP (Board Support Package): Drivers for components on the board (LEDs, buttons, LCD screen). Example: BSP_LED_On().

HAL (Hardware Abstraction Layer): Easy & Portable. Generic APIs to use MCU peripherals (like UART, I2C) without deep register knowledge. Uses interrupts/DMA for you.

LL (Low-Layer): Lean & Fast. Lightweight, register-level drivers for experts who need maximum performance and minimal overhead.

Figure 2. **STM32CubeF4 firmware architecture**


Level 2
Eval board and Discovery Kit demonstration
Applications
Level 1
Library and protocol based components (FatFS, FreeRTOS, USB Host, USB Device, Ethernet...)
Examples
BSP drivers
Level 0
Hardware abstraction layer (HAL)
Core drivers (optional)
Low Layer (LL)
HAL
DT73658V1

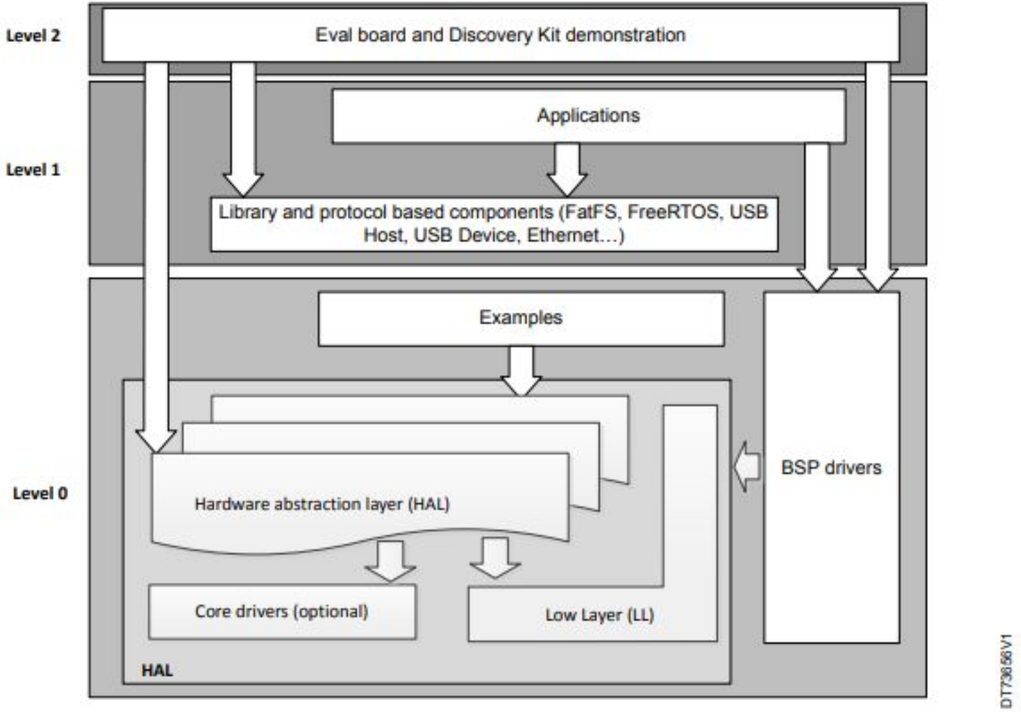
Writing the Software: HAL

Level 1: Middleware & Services
This level provides advanced software features that sit on top of the HAL/LL.
Libraries & Stacks: Ready-to-use components like:

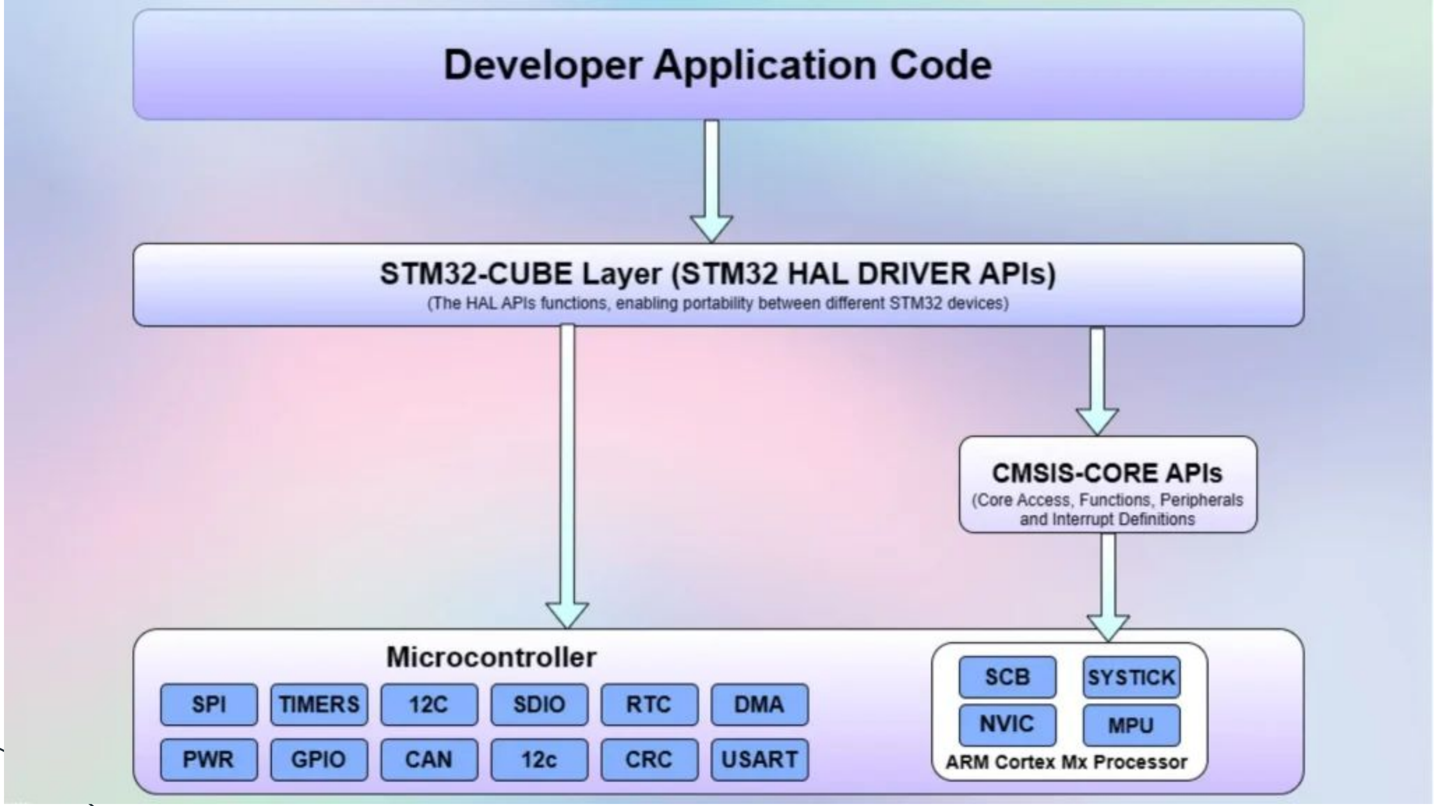
- USB Host/Device libraries
- Graphics Libraries (STemWin, TouchGFX)
- Real-Time OS (FreeRTOS)
- File System (FatFS)
- TCP/IP Network Stack (LwIP)
- SSL/TLS Security (mbedTLS)

Level 2: Final Application
This is the top level, where everything comes together.

Figure 2. STM32CubeF4 firmware architecture


Level 2
Eval board and Discovery Kit demonstration
Level 1
Applications
Library and protocol based components (FatFS, FreeRTOS, USB Host, USB Device, Ethernet...)
Level 0
Examples
Hardware abstraction layer (HAL)
BSP drivers
Core drivers (optional)
Low Layer (LL)
HAL
DT73656V1

Writing the Software: HAL


Developer Application Code
STM32-CUBE Layer (STM32 HAL DRIVER APIs)
(The HAL APIs functions, enabling portability between different STM32 devices)
CMSIS-CORE APIs
(Core Access, Functions, Peripherals and Interrupt Definitions
Microcontroller
SPI
TIMERS
12C
SDIO
RTC
DMA
PWR
GPIO
CAN
12c
CRC
USART
SCB
SYSTICK
NVIC
MPU
ARM Cortex Mx Processor

Freeways
free software club

Thank You