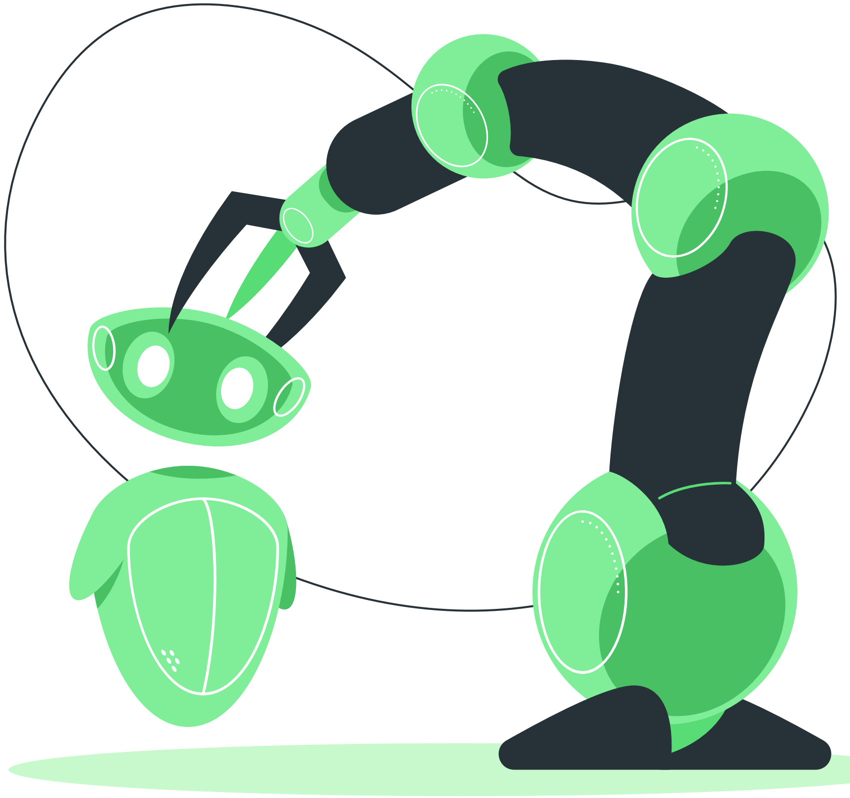
Freeways

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

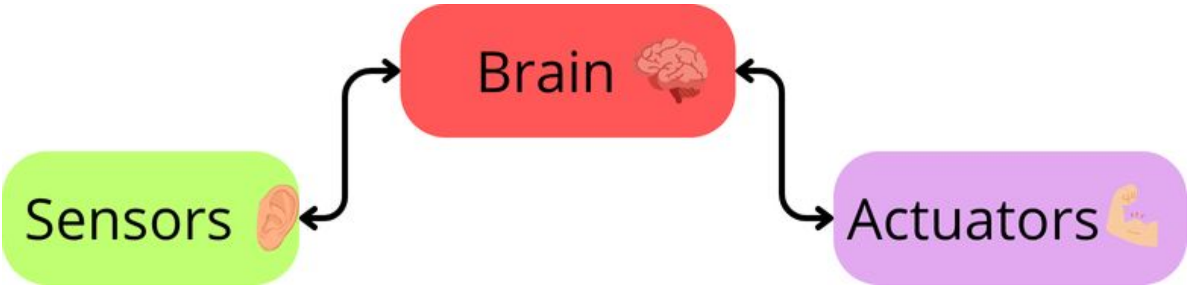
8 GPIO output STM32 and HAL

Why you should be here ?

- The field (AI, Cybersecurity, IoT) ...
- Why STM32?
- What you will learn

Introduction to embedded systems

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



Brain 🧠

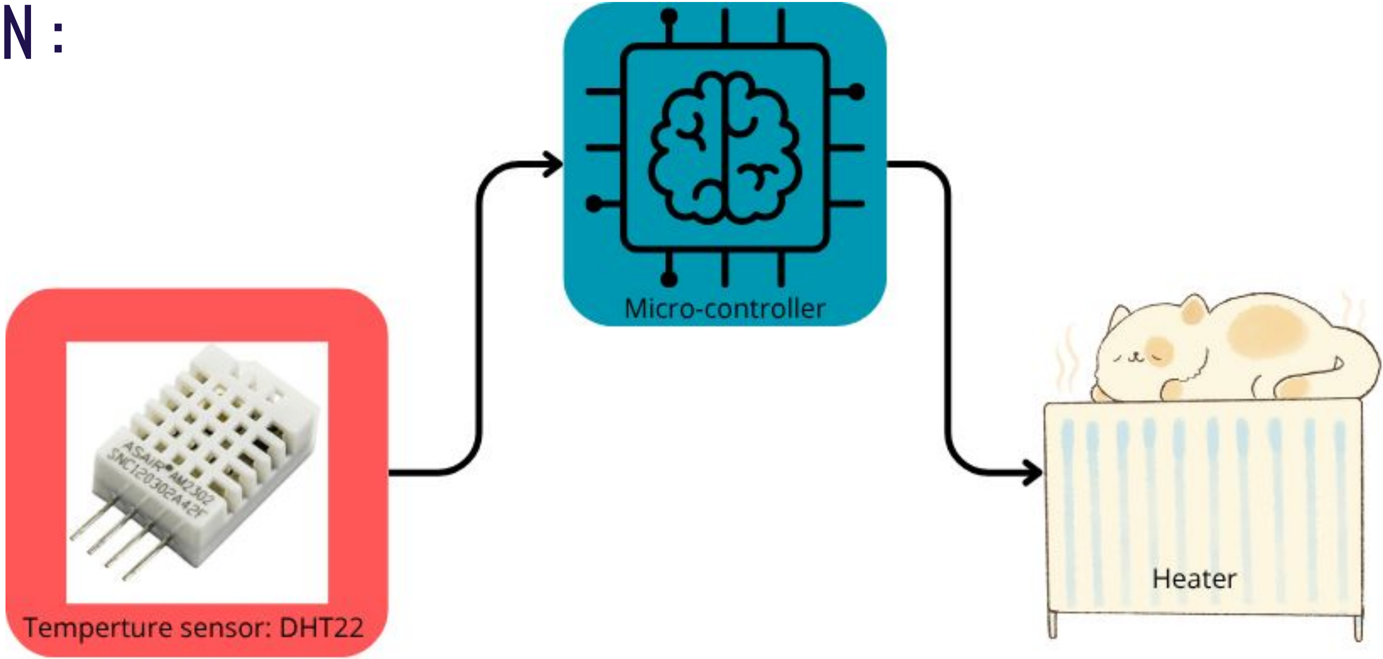
Sensors 👂

Actuators 💪

Thermostat

Introduction to embedded systems

THE BRAIN :



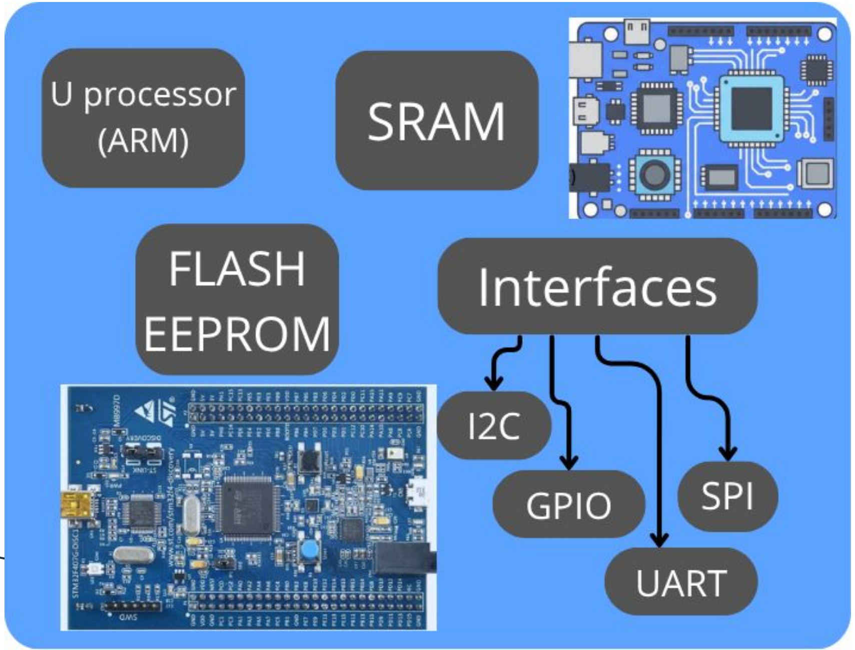
Microcontroller

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

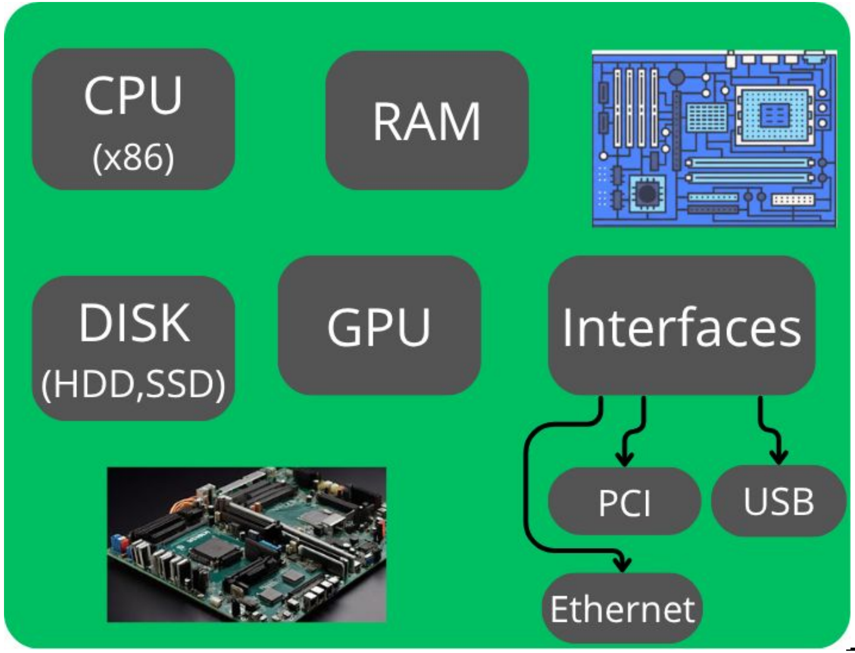
I2C

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)

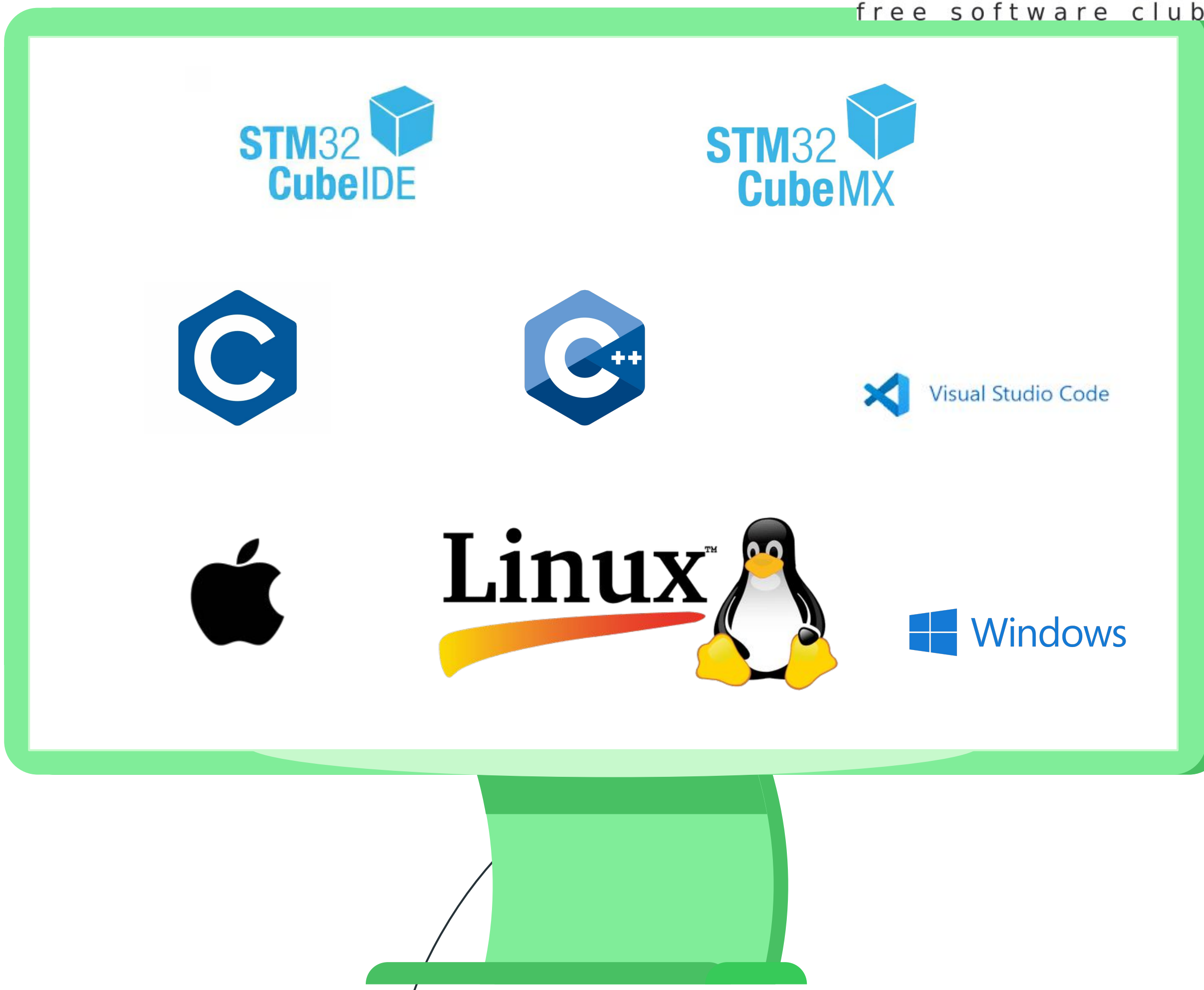
Consumer electronics (smart devices, wearables)

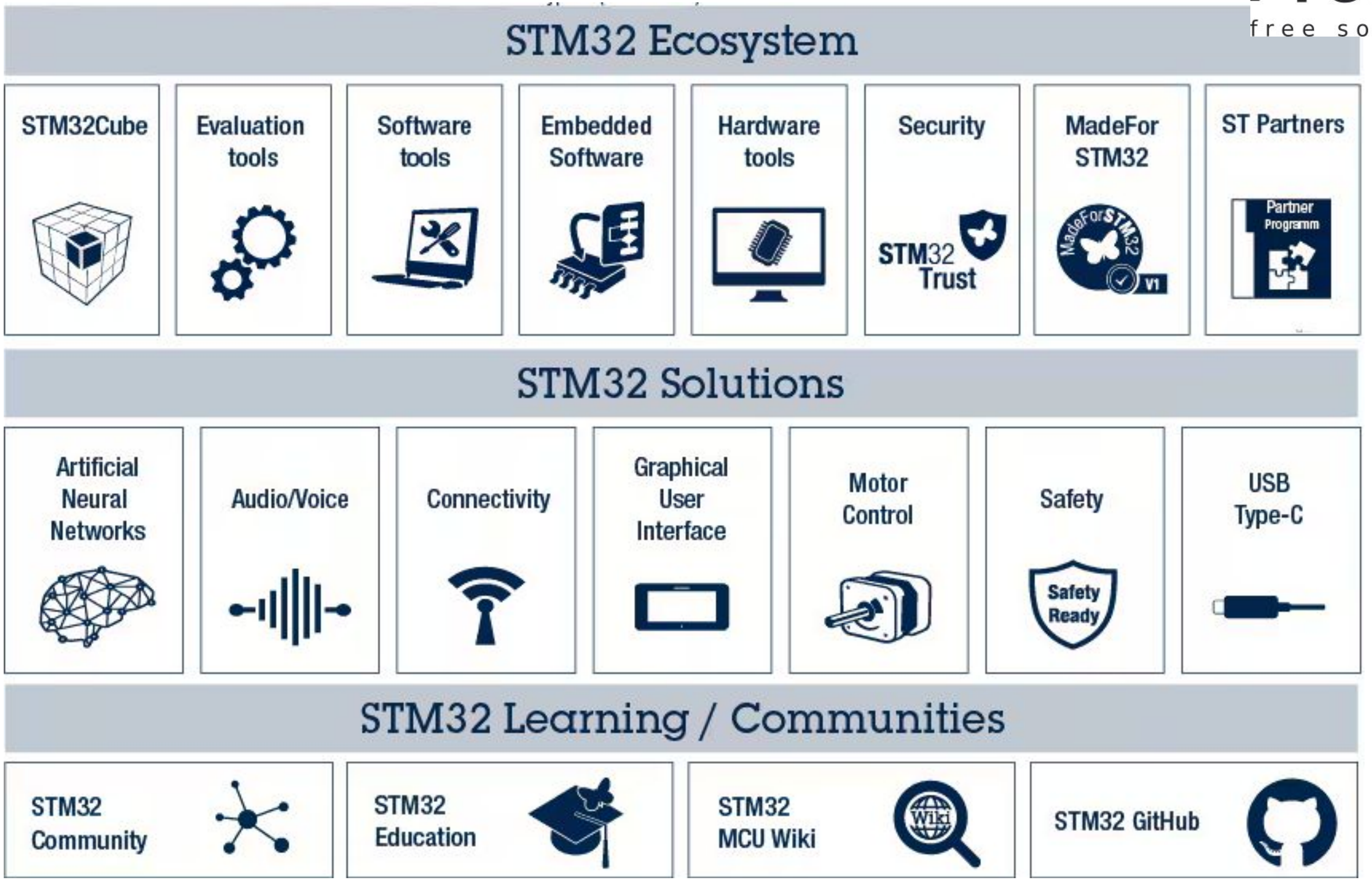
Internet of Things (IoT)

Medical equipment

Automotive systems

STM32 Ecosystem:





STM32 Ecosystem

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

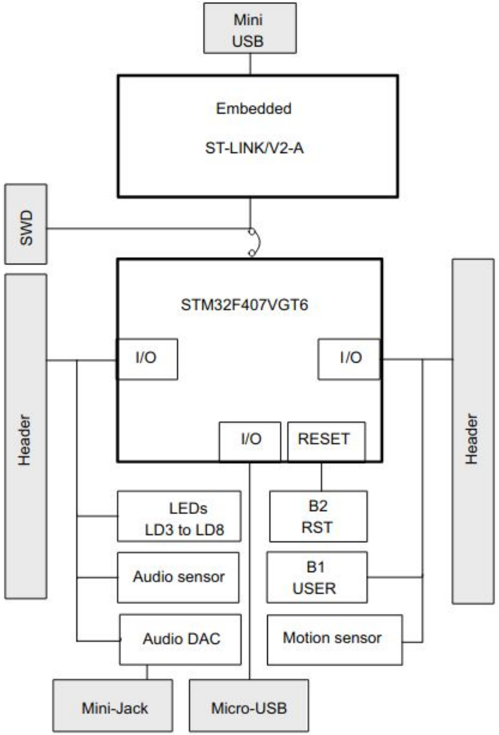
STM32 Solutions

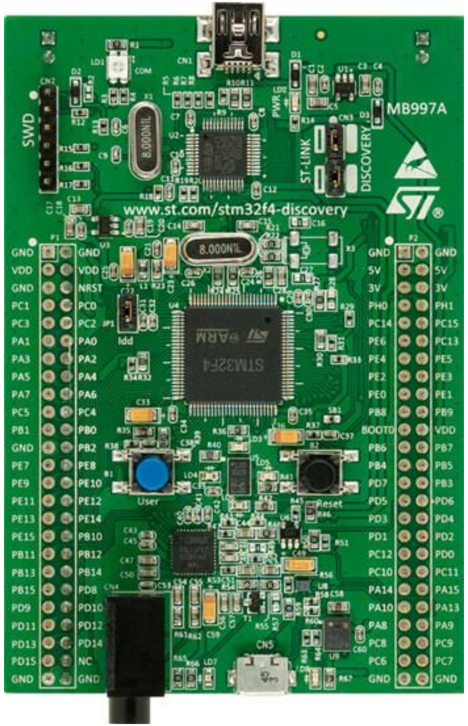
- Artificial Neural Networks
- Audio/Voice
- Connectivity
- Graphical User Interface
- Motor Control
- Safety
- USB Type-C

STM32 Learning / Communities

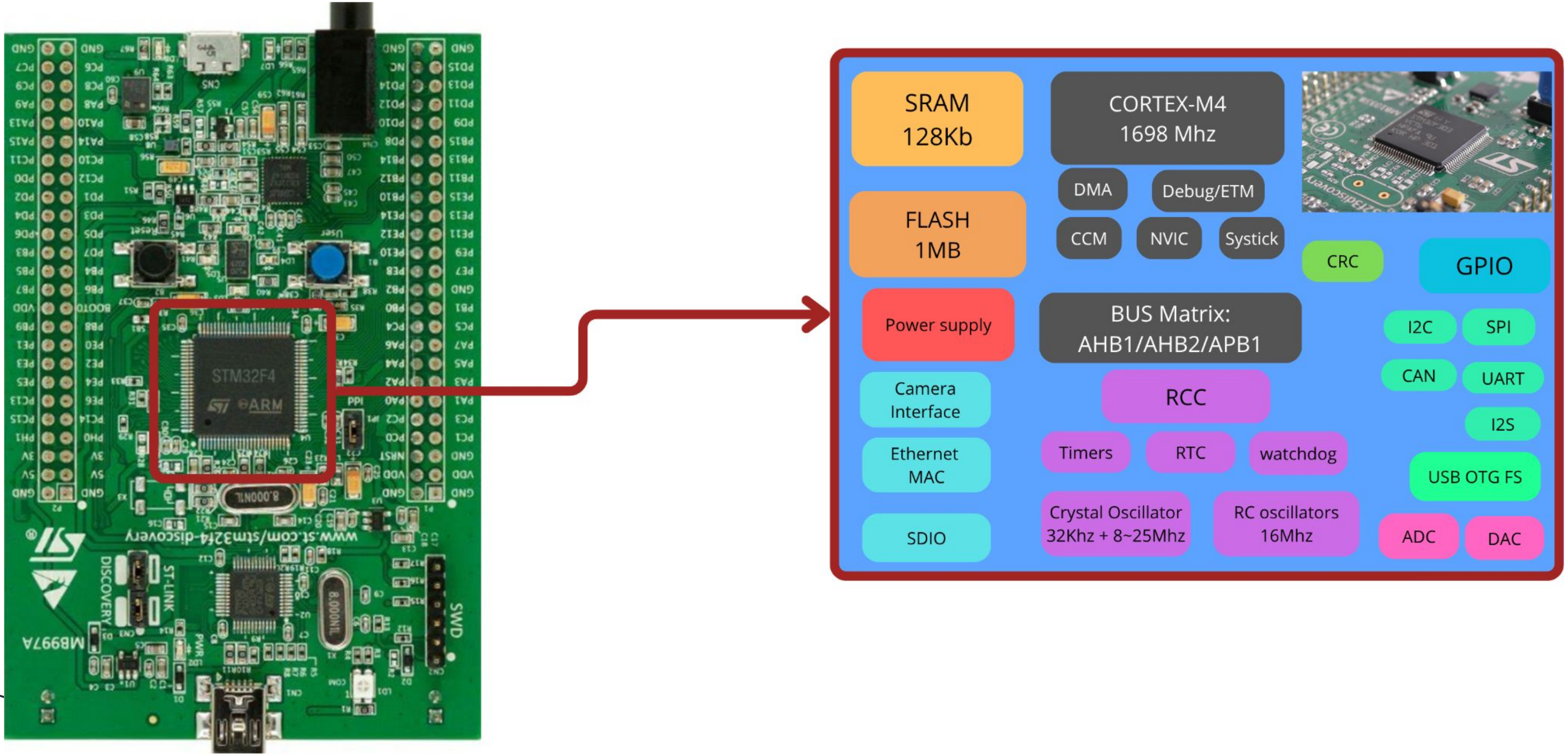
- STM32 Community
- STM32 Education
- STM32 MCU Wiki
- STM32 GitHub

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

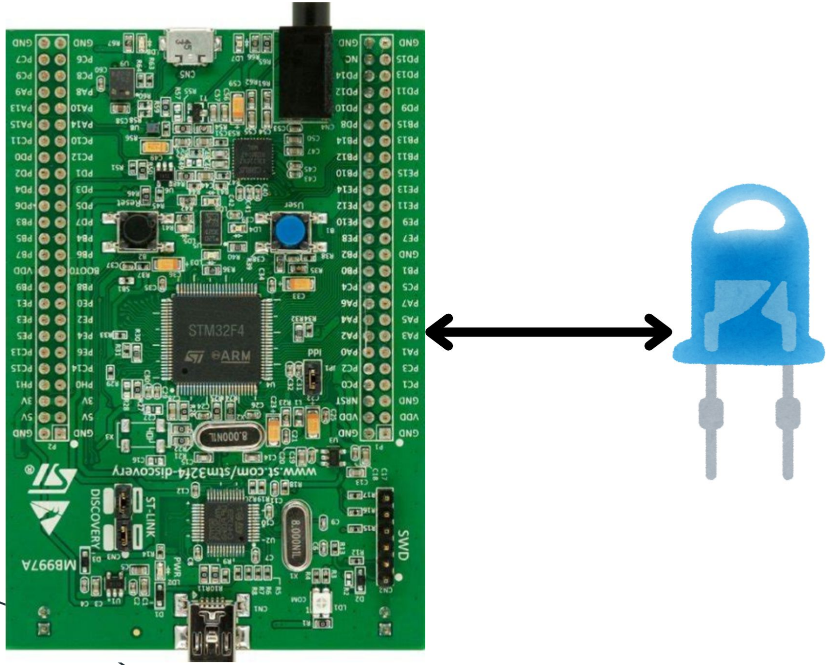
RC oscillators 16Mhz

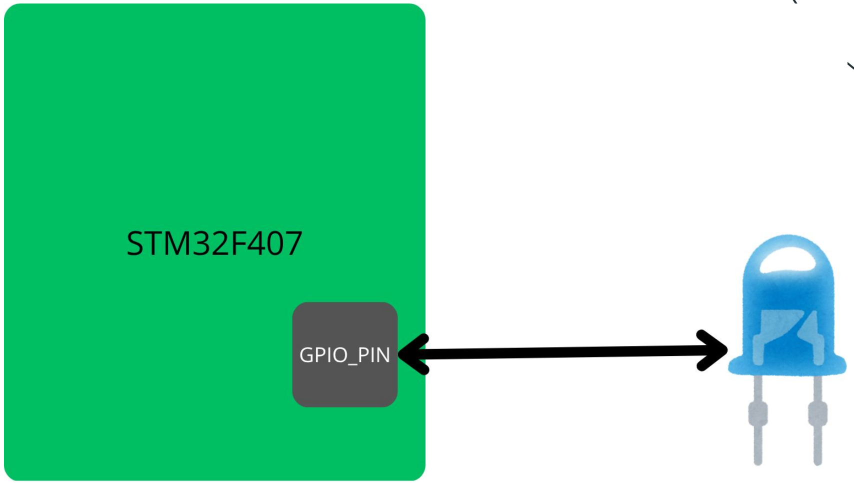
ADC

DAC

Let's Do something :

Case Study: Toggling an LED

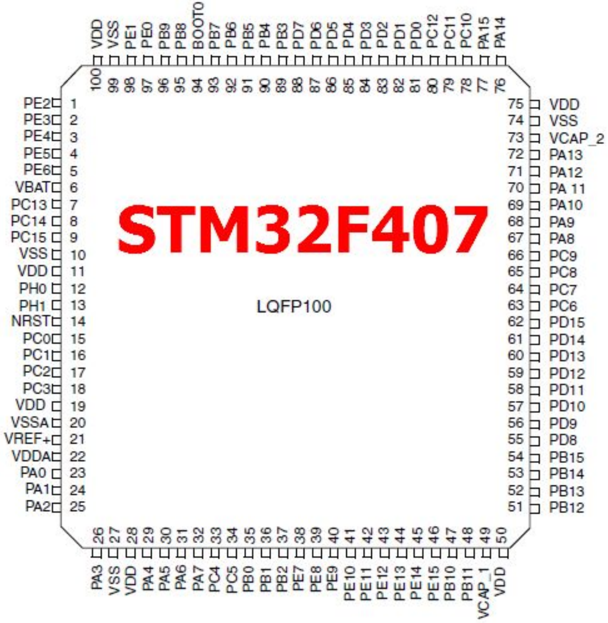


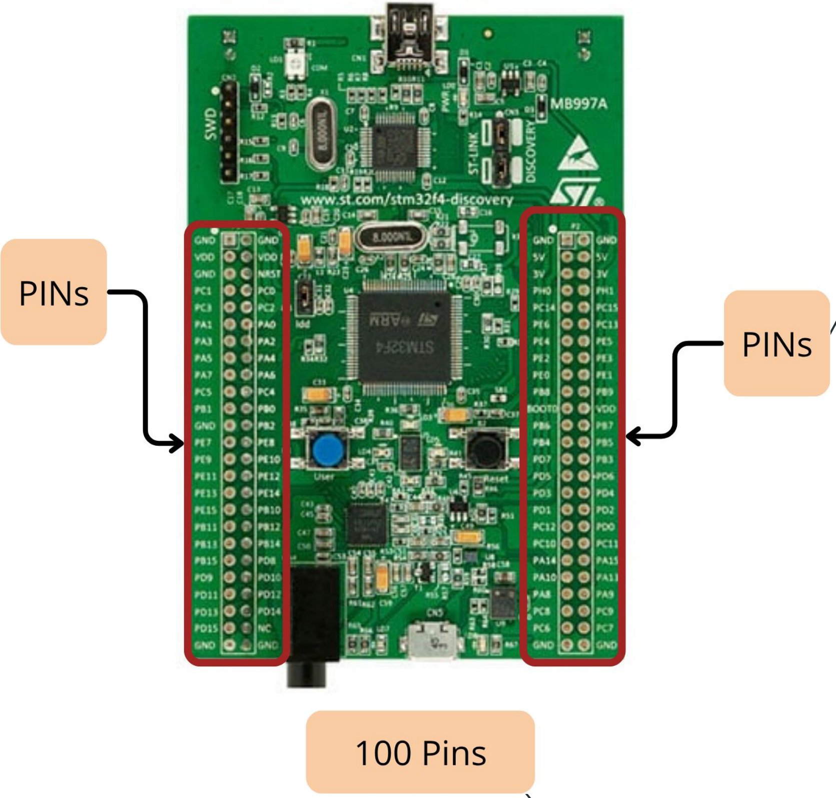


STM32F407

GPIO_PIN

STM32 GPIO : General purpose Input Output



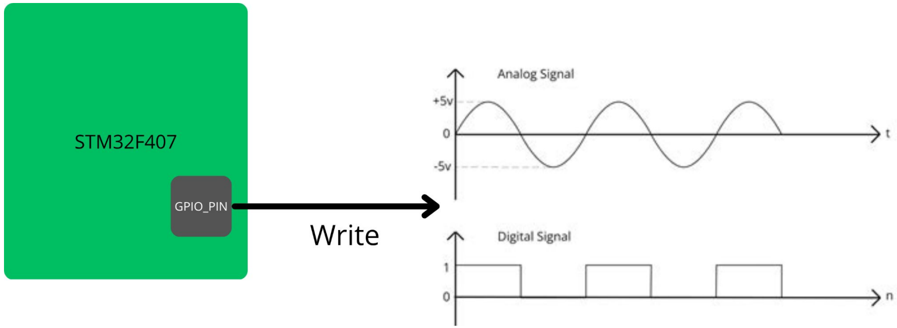


STM32 GPIO : General purpose Input Output

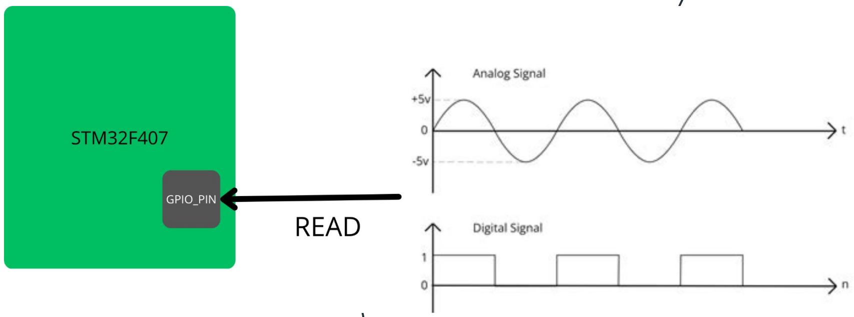
GPIO stands for General Purpose Input/Output.
It’s the most basic and versatile feature of a microcontroller, the way it interacts with the outside world.

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

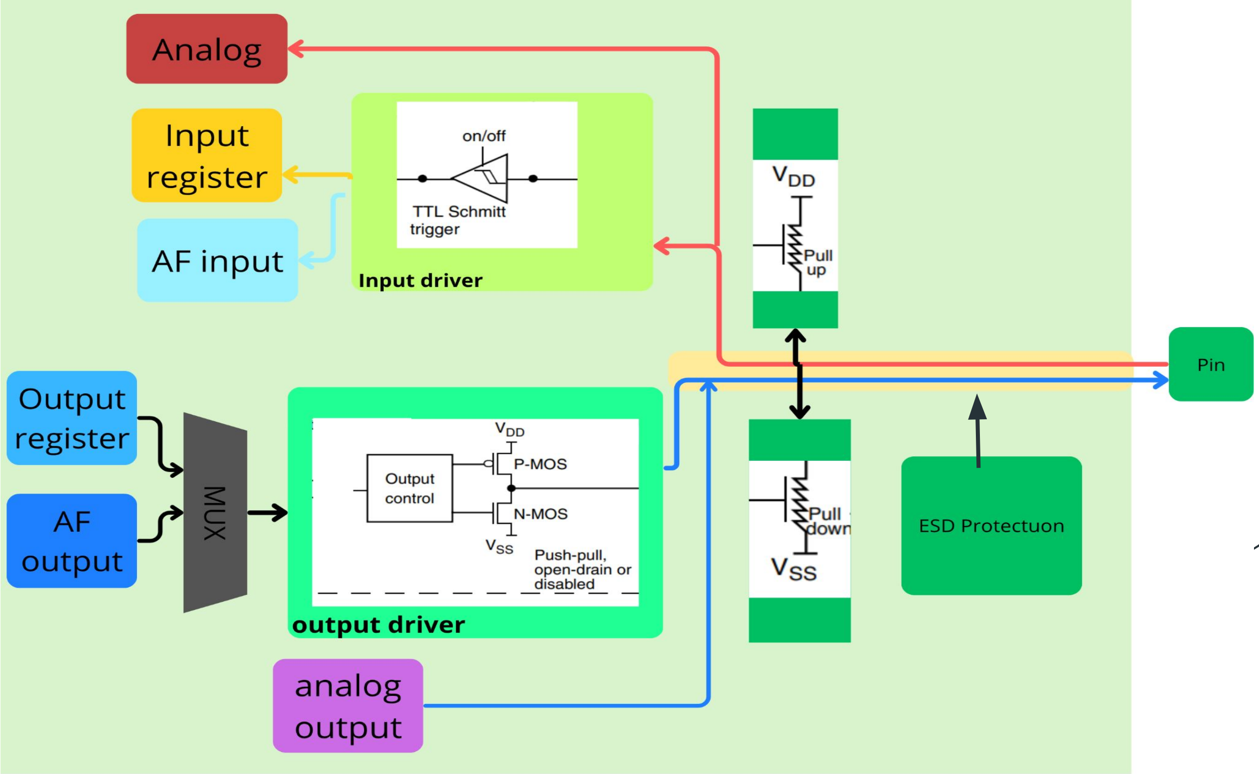
The STM32 groups the GPIOS in to clusters called PORTs indicated by GPIOx .
By x we mean: GPIOA to GPIOI .
Each port has 16 pins: From 0 to 15
For example:

- 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

STM32 GPIO : GPIO Structure

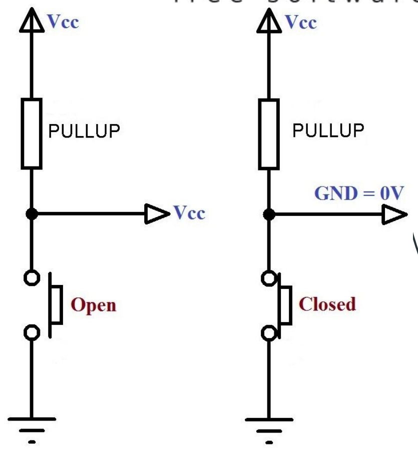


STM32 GPIO : GPIO Modes

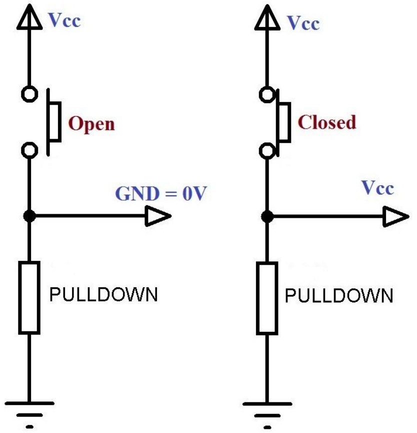
Input Mode:

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

PullUP



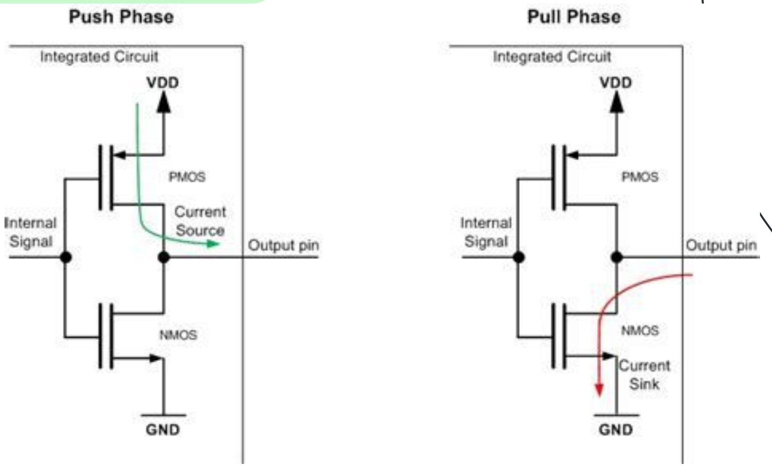
PullDown



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



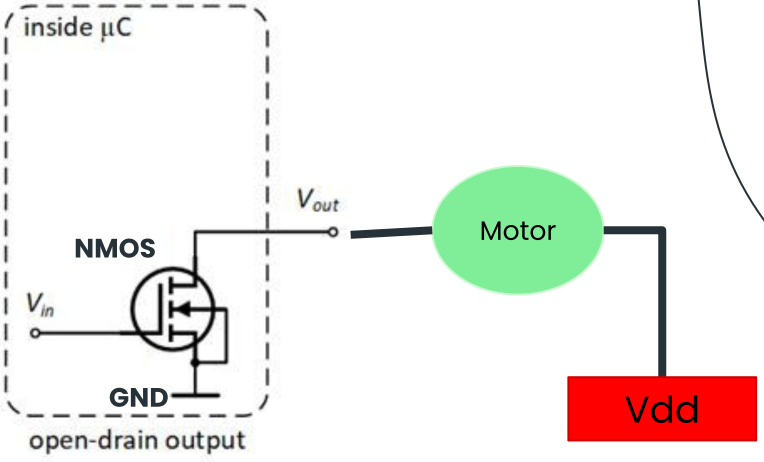
- It contains 2 transistors: **PMOS** and **NMOS**, They both act as switches letting current in and out.
- The **PMOS** is **ON** when he gets a **LOGIC 1,** hence he lets current pass form **Vdd** to the **output pin (3.3V)**.
- The **NMOS** is **ON** when he gets **LOGIC 0**, hence connecting the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

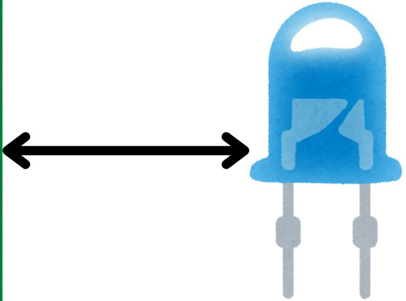
OpenDrain: Drives the pin Low (0V).
Used when there is a external energy Source (**Vdd**)
Example: Motor control, or controlling and LED or sensors that needs higher than (3.3V) to work.

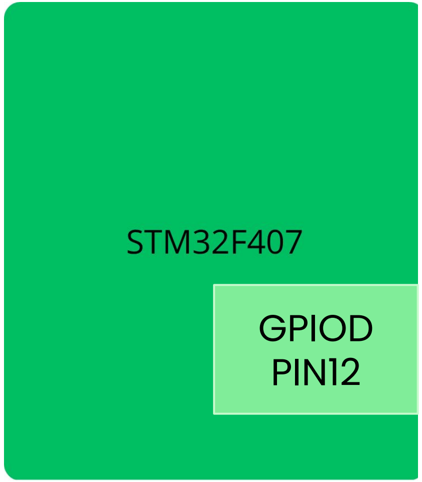


- It contains 1 transistors: **NMOS**:
- The **NMOS** is **ON** when he gets **LOGIC 0**, hence connecting the **GND** the the **output pin (0V)**, the circuit closes and the motor spins.
- **LOGIC 1** closes the circuit and the **NMOS** is **OFF**.

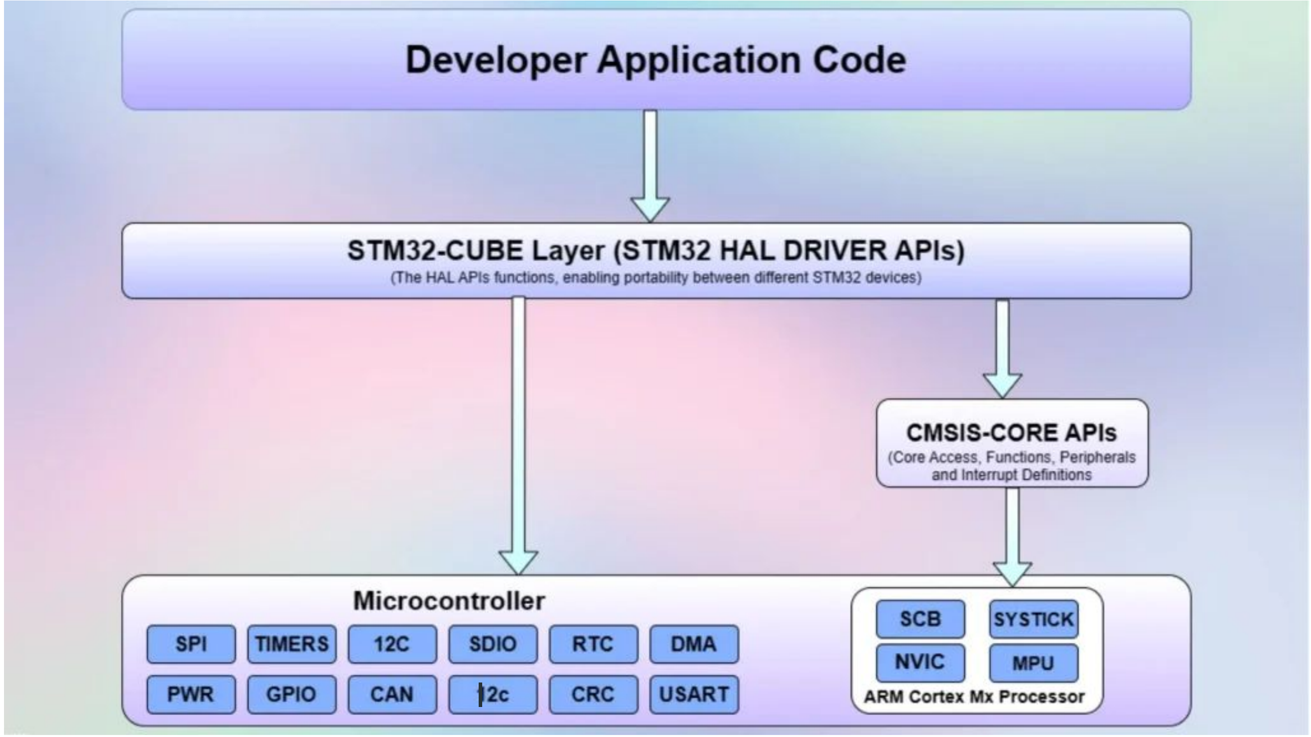
GPIO output STM32 and HAL: blinking an LED:

Toggling an LED





HAL:



Developer Application Code

STM32-CUBE Layer (STM32 HAL DRIVER APIs)
(The HAL APIs functions, enabling portability between different STM32 devices)

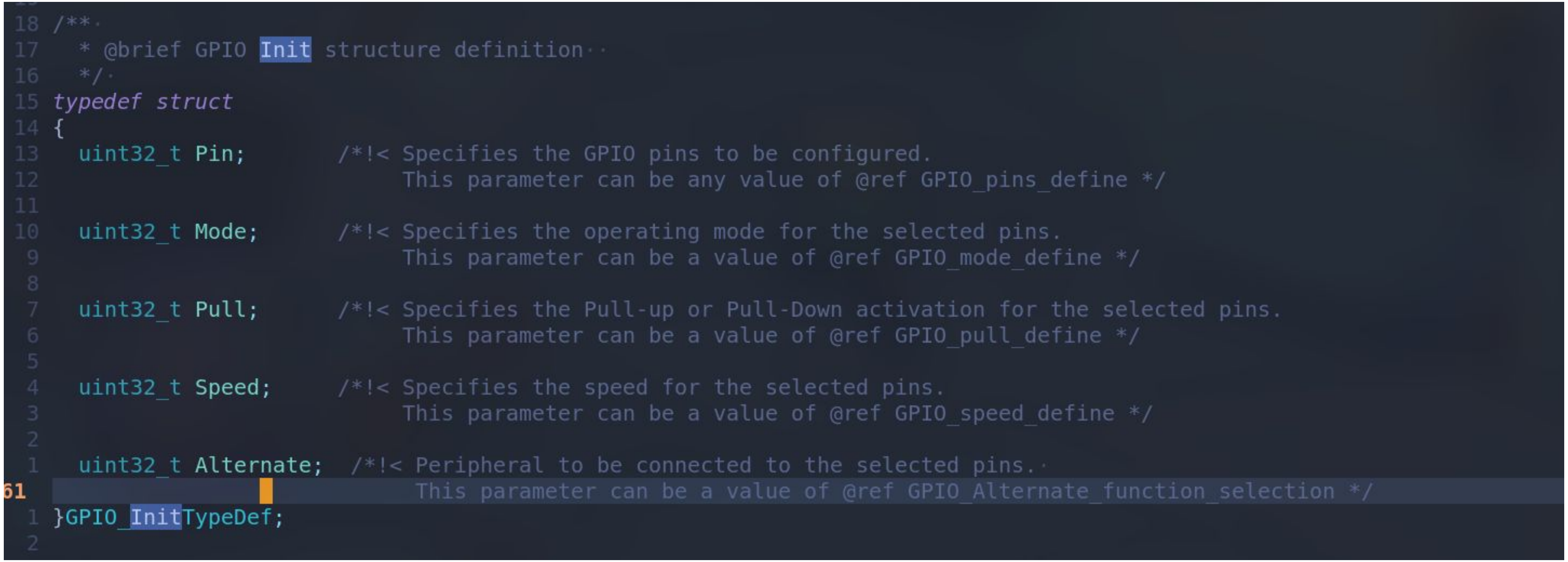
CMSIS-CORE APIs
(Core Access, Functions, Peripherals and Interrupt Definitions

Microcontroller
SPI
TIMERS
12C
SDIO
RTC
DMA
PWR
GPIO
CAN
I2c
CRC
USART

SCB
SYSTICK
NVIC
MPU
ARM Cortex Mx Processor

HAL:

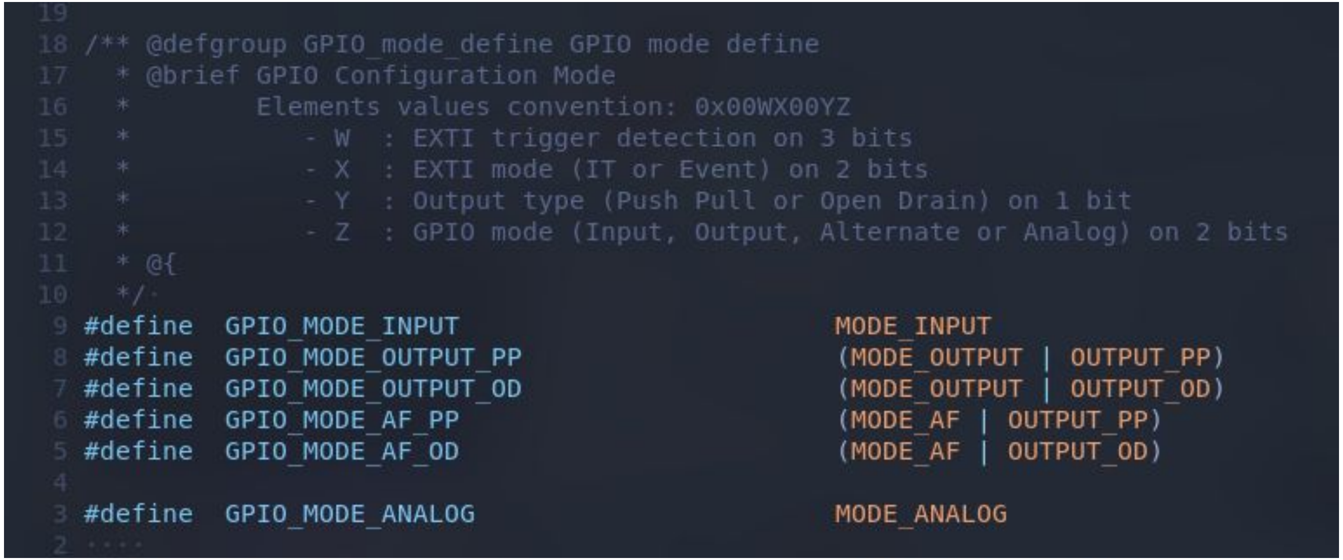
GPIO InitStruct



```
/**
  * @brief GPIO Init structure definition
  */
typedef struct
{
  uint32_t Pin;       /*!< Specifies the GPIO pins to be configured.
                           This parameter can be any value of @ref GPIO_pins_define */

  uint32_t Mode;      /*!< Specifies the operating mode for the selected pins.
                           This parameter can be a value of @ref GPIO_mode_define */

  uint32_t Pull;      /*!< Specifies the Pull-up or Pull-Down activation for the selected pins.
                           This parameter can be a value of @ref GPIO_pull_define */

  uint32_t Speed;     /*!< Specifies the speed for the selected pins.
                           This parameter can be a value of @ref GPIO_speed_define */

  uint32_t Alternate;  /*!< Peripheral to be connected to the selected pins.
                            This parameter can be a value of @ref GPIO_Alternate_function_selection */
}GPIO_InitTypeDef;
```

HAL:

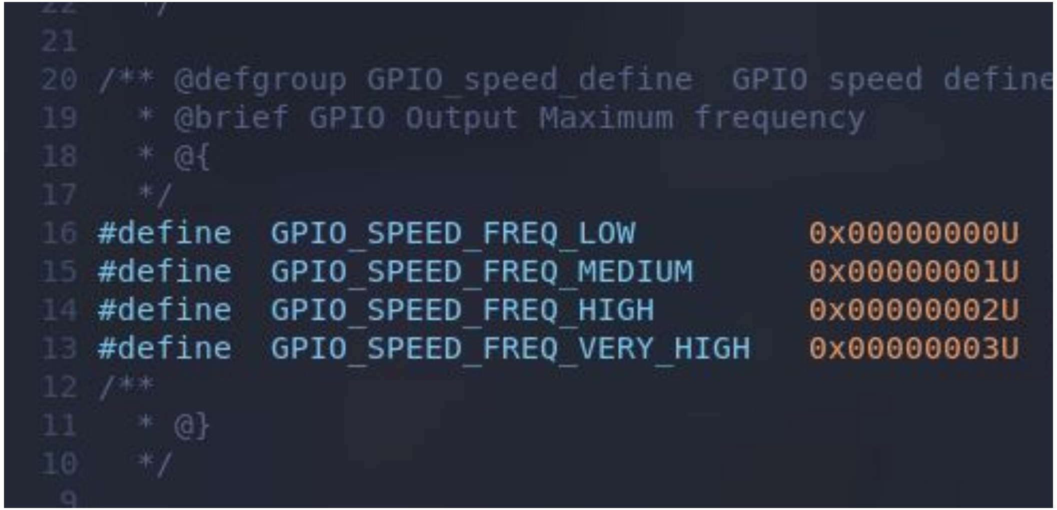
GPIO Mode



```
/** @defgroup GPIO_mode_define GPIO mode define
  * @brief GPIO Configuration Mode
  *        Elements values convention: 0x00WX00YZ
  *           - W  : EXTI trigger detection on 3 bits
  *           - X  : EXTI mode (IT or Event) on 2 bits
  *           - Y  : Output type (Push Pull or Open Drain) on 1 bit
  *           - Z  : GPIO mode (Input, Output, Alternate or Analog) on 2 bits
  * @{
  */
#define  GPIO_MODE_INPUT                        MODE_INPUT
#define  GPIO_MODE_OUTPUT_PP                    (MODE_OUTPUT | OUTPUT_PP)
#define  GPIO_MODE_OUTPUT_OD                    (MODE_OUTPUT | OUTPUT_OD)
#define  GPIO_MODE_AF_PP                        (MODE_AF | OUTPUT_PP)
#define  GPIO_MODE_AF_OD                        (MODE_AF | OUTPUT_OD)

#define  GPIO_MODE_ANALOG                       MODE_ANALOG
```

HAL:

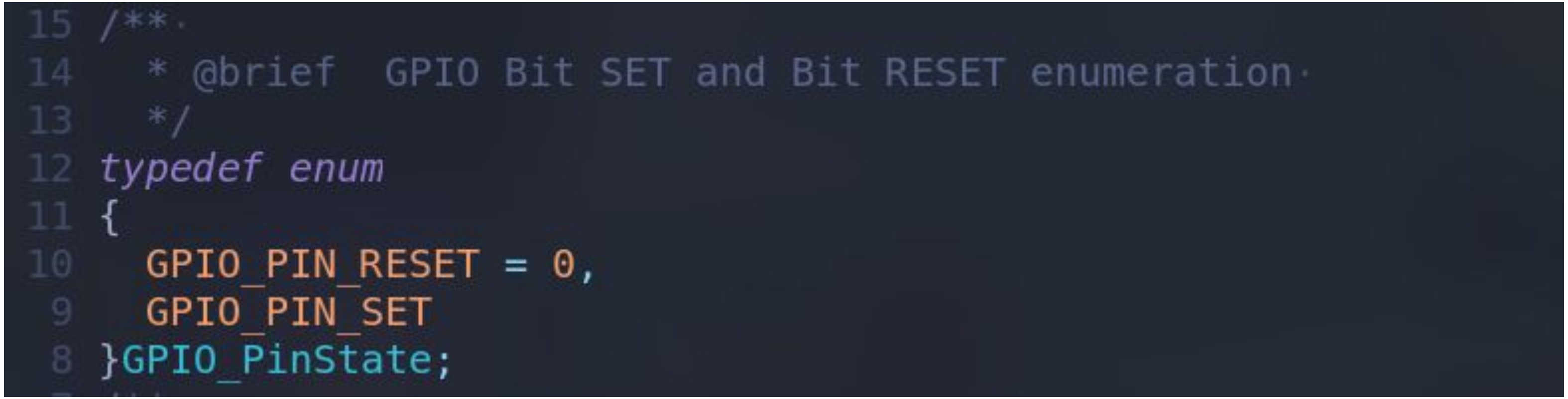
GPIO Speed



```
  */

/** @defgroup GPIO_speed_define  GPIO speed define
  * @brief GPIO Output Maximum frequency
  * @{
  */
#define  GPIO_SPEED_FREQ_LOW         0x00000000U
#define  GPIO_SPEED_FREQ_MEDIUM      0x00000001U
#define  GPIO_SPEED_FREQ_HIGH        0x00000002U
#define  GPIO_SPEED_FREQ_VERY_HIGH   0x00000003U
/**
  * @}
  */
```

HAL:

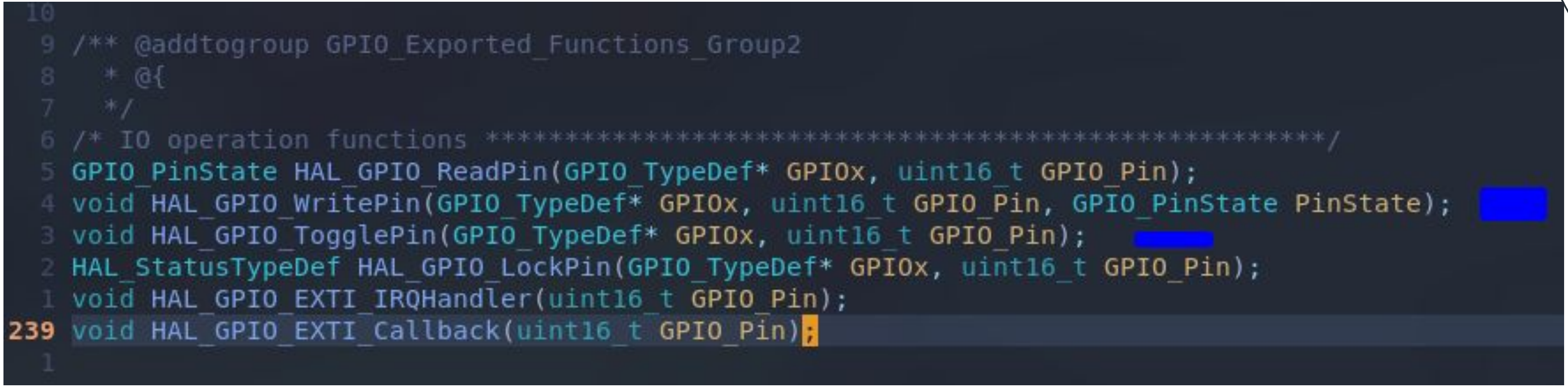
GPIO Pin state



```
/**
  * @brief  GPIO Bit SET and Bit RESET enumeration
  */
typedef enum
{
  GPIO_PIN_RESET = 0,
  GPIO_PIN_SET
}GPIO_PinState;
```

HAL:

HAL function



```
/** @addtogroup GPIO_Exported_Functions_Group2
  * @{
  */
/* IO operation functions *****************************************************/
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
HAL_StatusTypeDef HAL_GPIO_LockPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin);
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
```

Freeways
free software club

Thank You