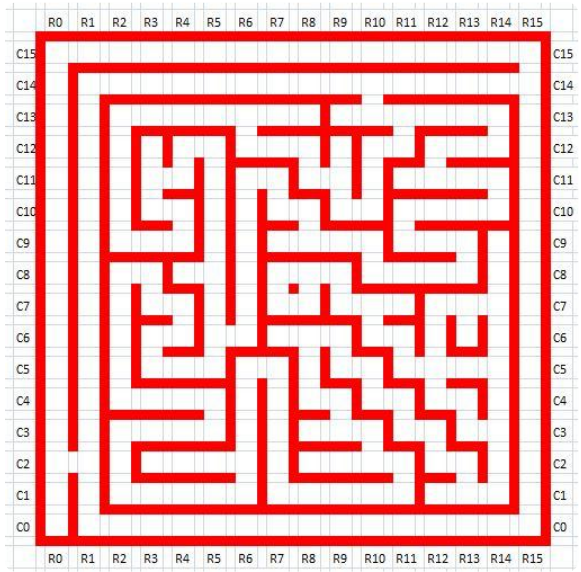
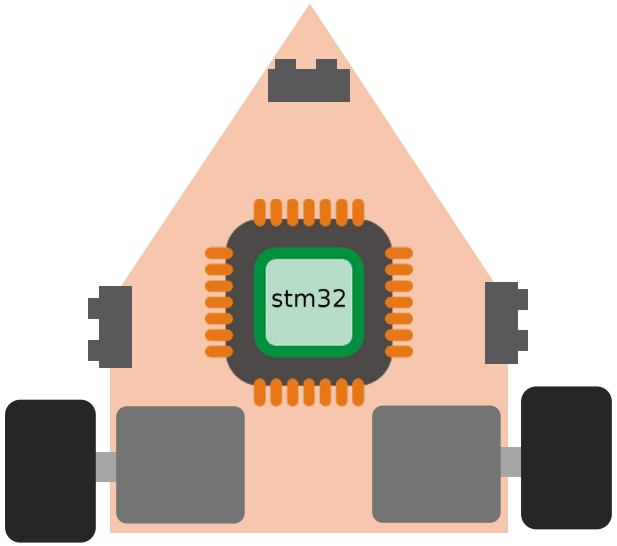
STM32 for Maze-solving Robotics

17 Dec 2024





Intro to STM32

The section for introduce about what is STM32.

Why STM32 ?

STM32 is better suited for projects requiring **high performance, advanced features, scalability, and reliability**, which are essential for professional and industrial applications.

- Higher Performance
- Industry-Grade Reliability
- Flexible Development Ecosystem
- Advanced Features
- Cost-Effective in the Long Run

Ecosystem of STM32

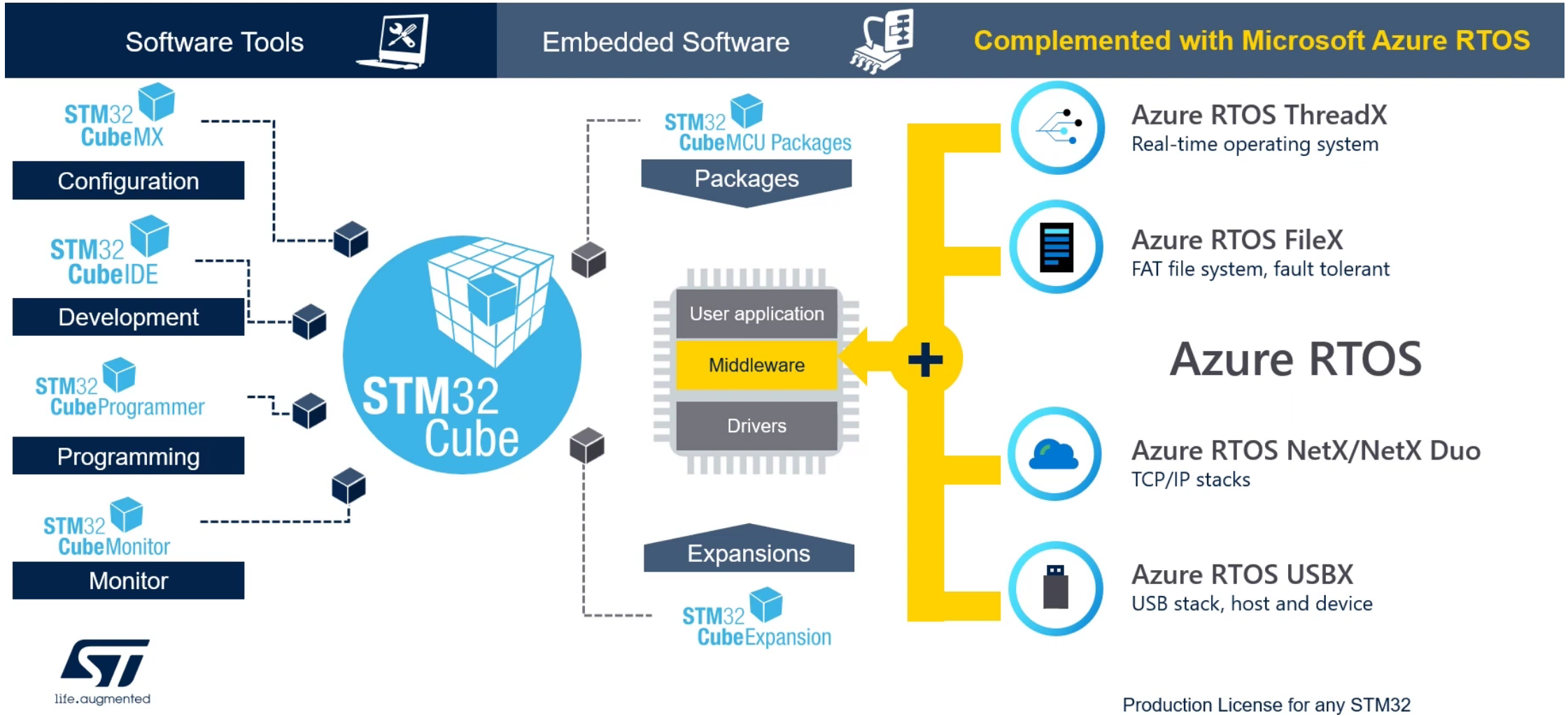
Advantage of STM32

1. Comprehensive offer of production-ready embedded software and tools
2. Compatibility across the STM32 hardware products
3. Free and business-friendly license

Ecosystem of STM32

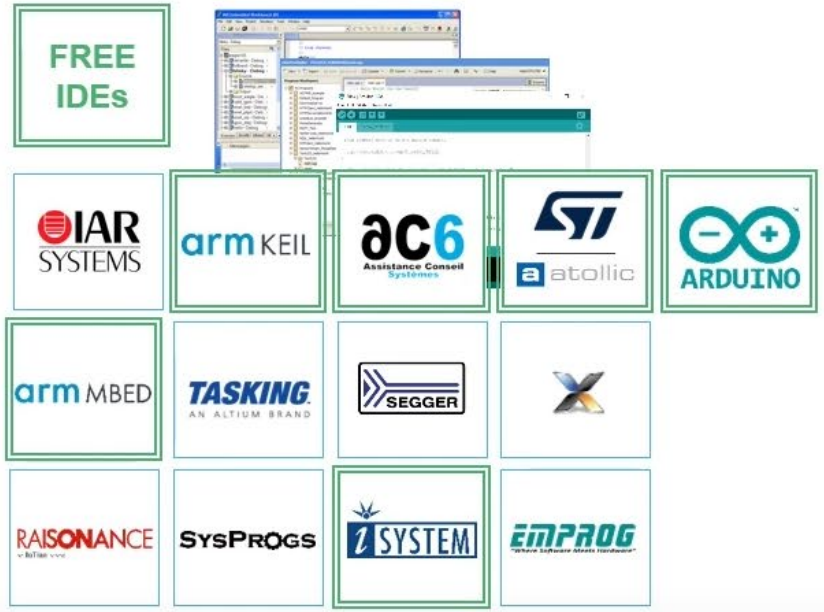
STM32Cube MCU ecosystem

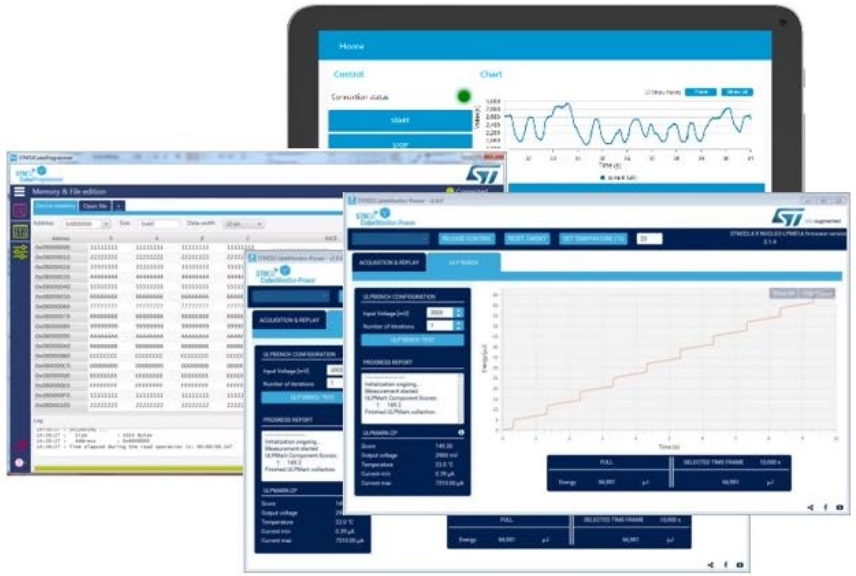
Ecosystem of STM32



Inside the STM32Cube MCU ecosystem (Software Tools)

Ecosystem of STM32





STM32 CubeMX

STM32 CubeIDE

STM32 CubeProgrammer

STM32 CubeMonitor

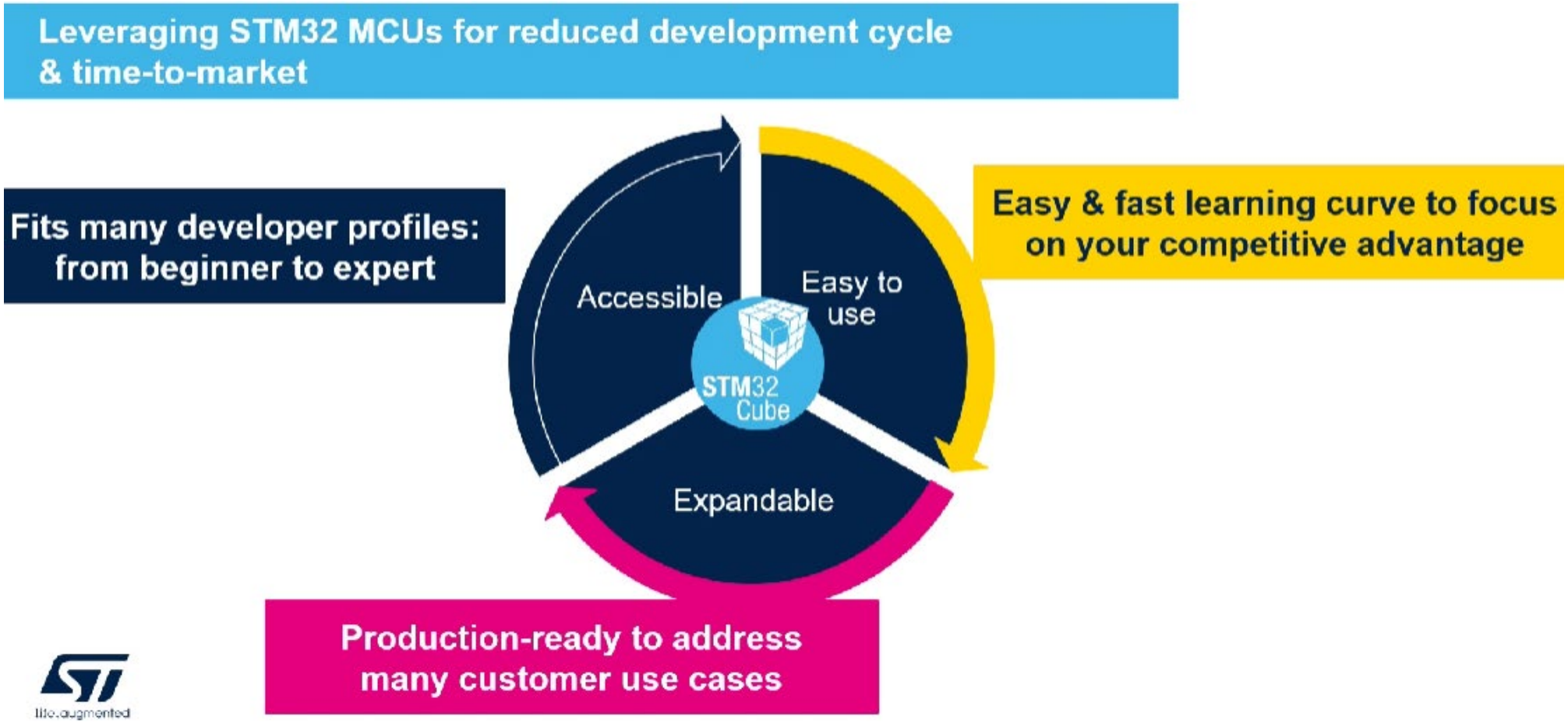
STM32CubeMX, GUI builders
Configure & generate code

ST and Partner IDEs
Compile and debug

STM32CubeProg/Monitor
Monitor, program & utilities

The programming flow in only three steps (C/C++)

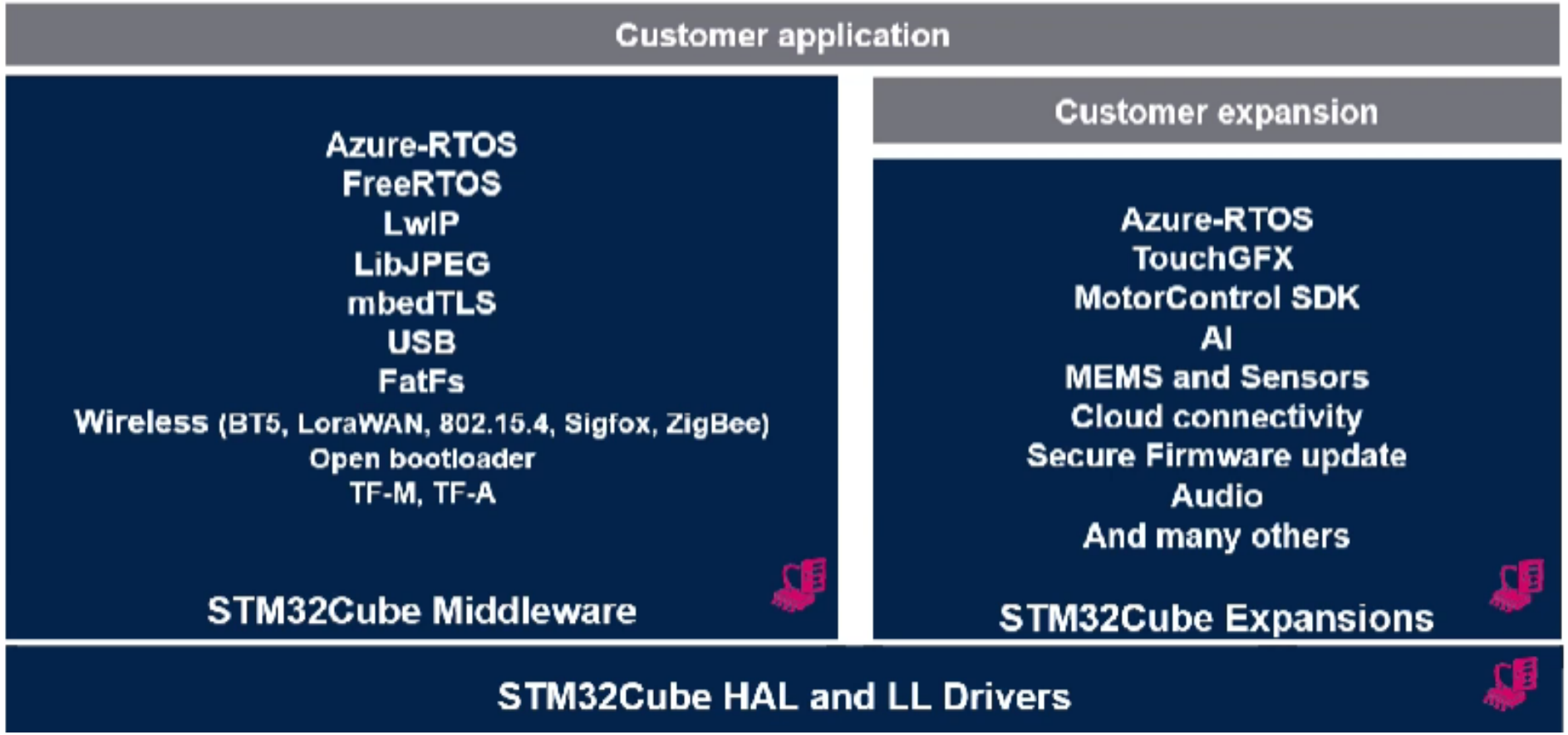
Ecosystem of STM32



Leveraging STM32 MCUs for reduced development cycle and time-to-market

Ecosystem of STM32

A flexible, scalable, and consistent embedded software offer for STM32 MCUs since 2014



Available on GitHub

Ecosystem of STM32

Development tools adapted to your needs

STM32 Nucleo	Discovery kits	Evaluation boards	STM32 Nucleo expansion
Flexible prototyping	Key feature prototyping	Full feature evaluation	Functionality add-on
www.st.com/stm32nucleo	www.st.com/stm32discovery	www.st.com/stm32evaltools	www.st.com/x-nucleo

Ecosystem of STM32

Easy PCB design in STM32 family

Design the specific PCB for system

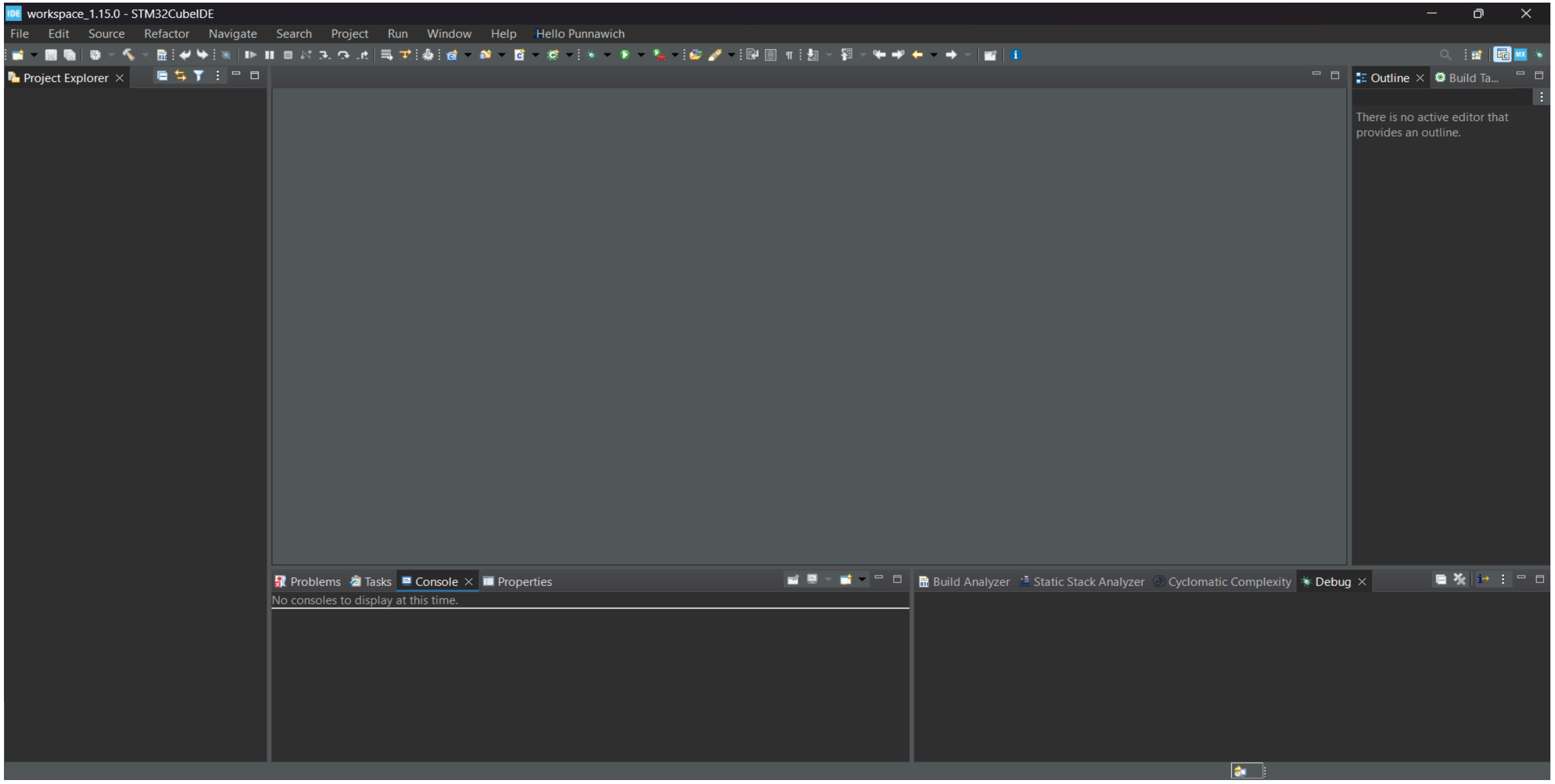
Order the PCB production

Programming the PCB with STM32 IDE

Setting up the environment for STM32 IDE

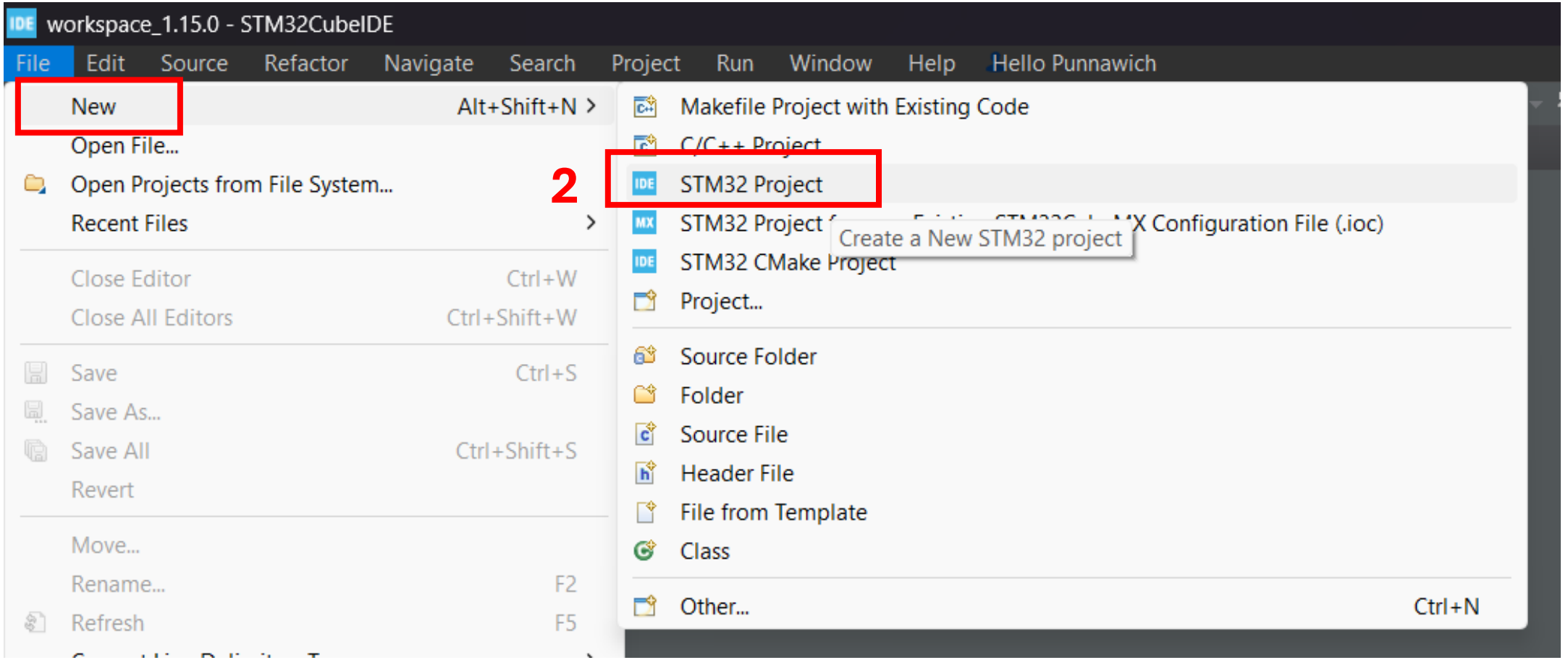
How to start your STM32 project and first project in STM32

Setting up the environment

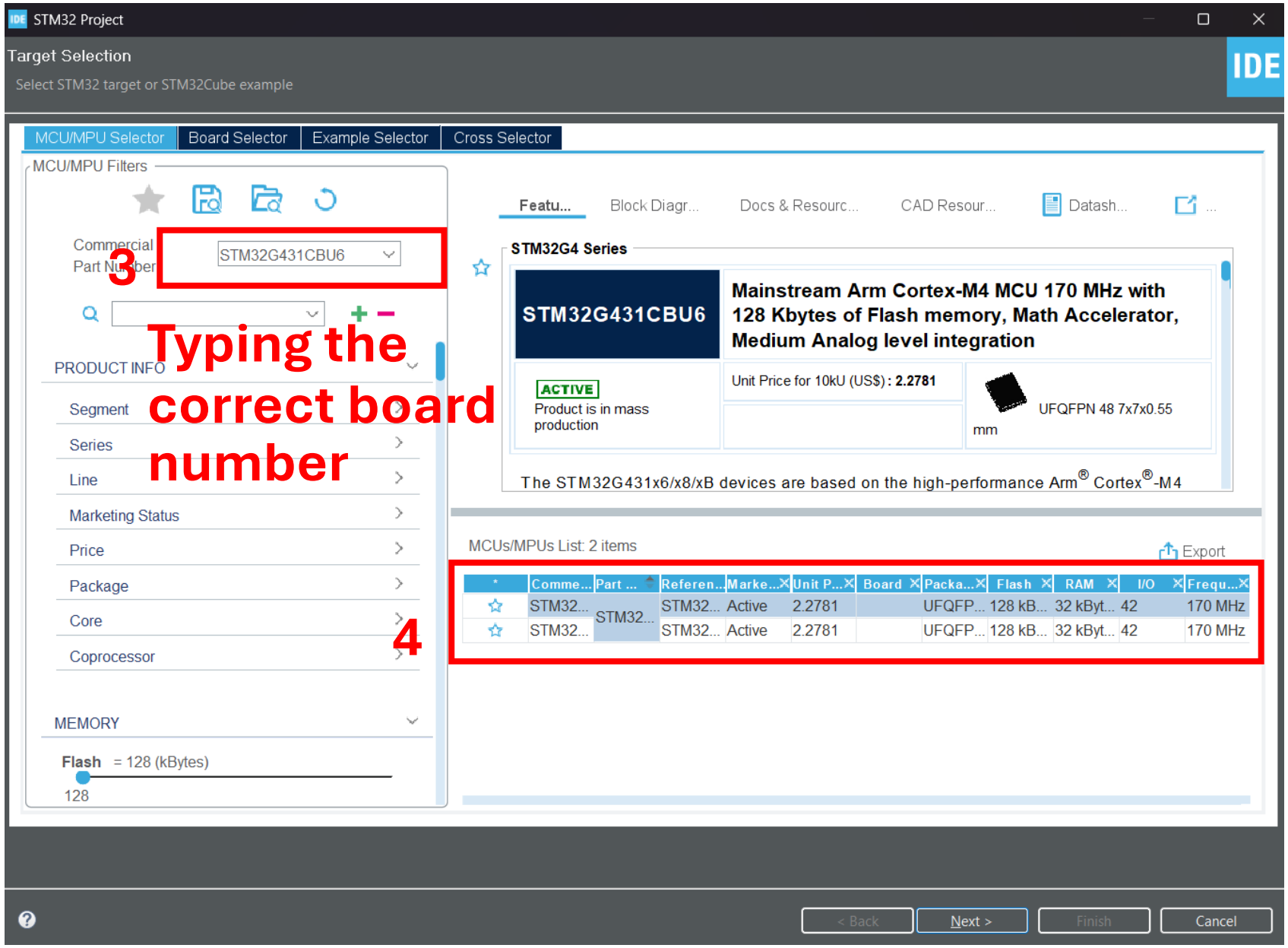


Setting up the environment

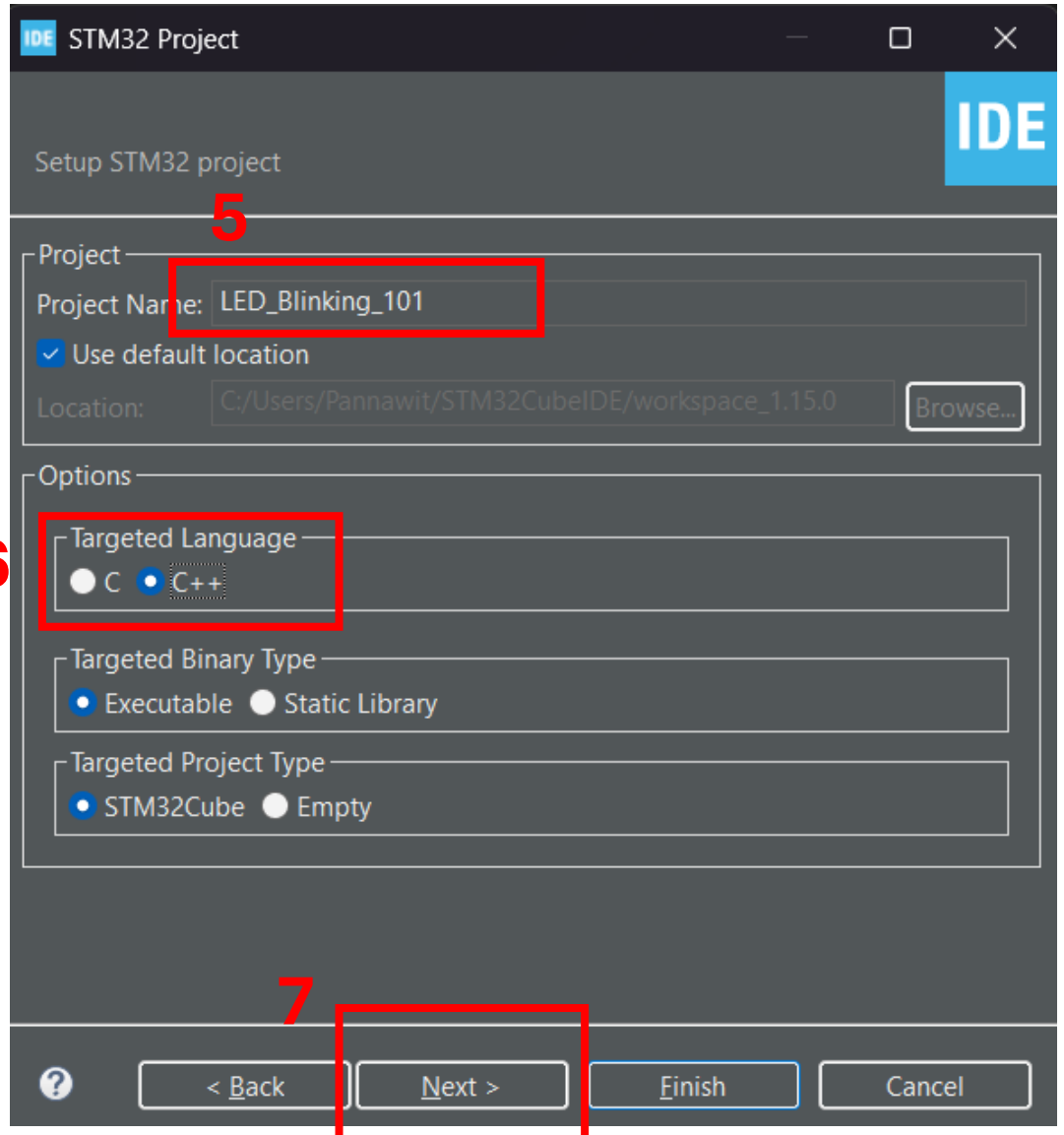
Creating the STM32 Project

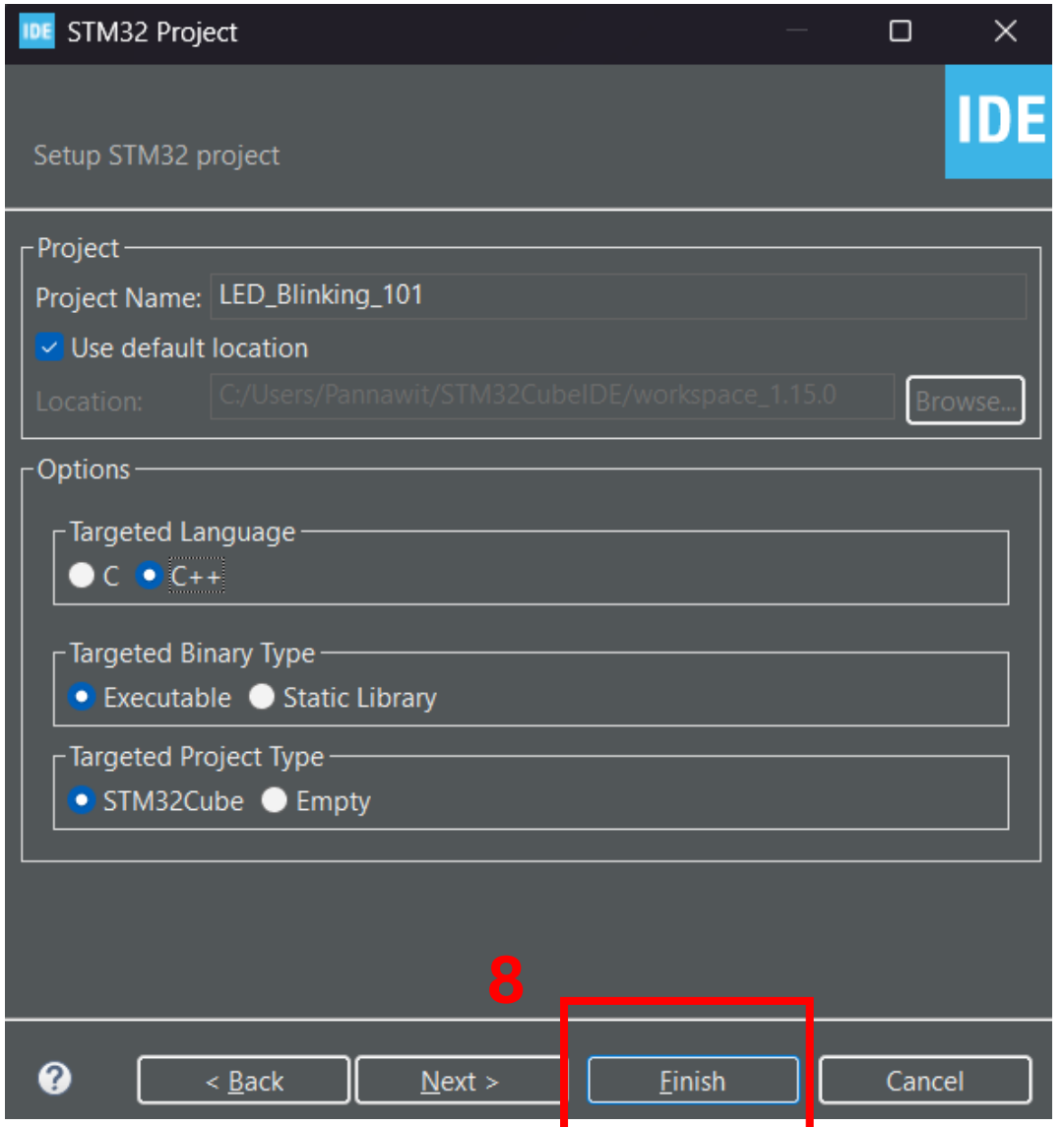


Setting up the environment

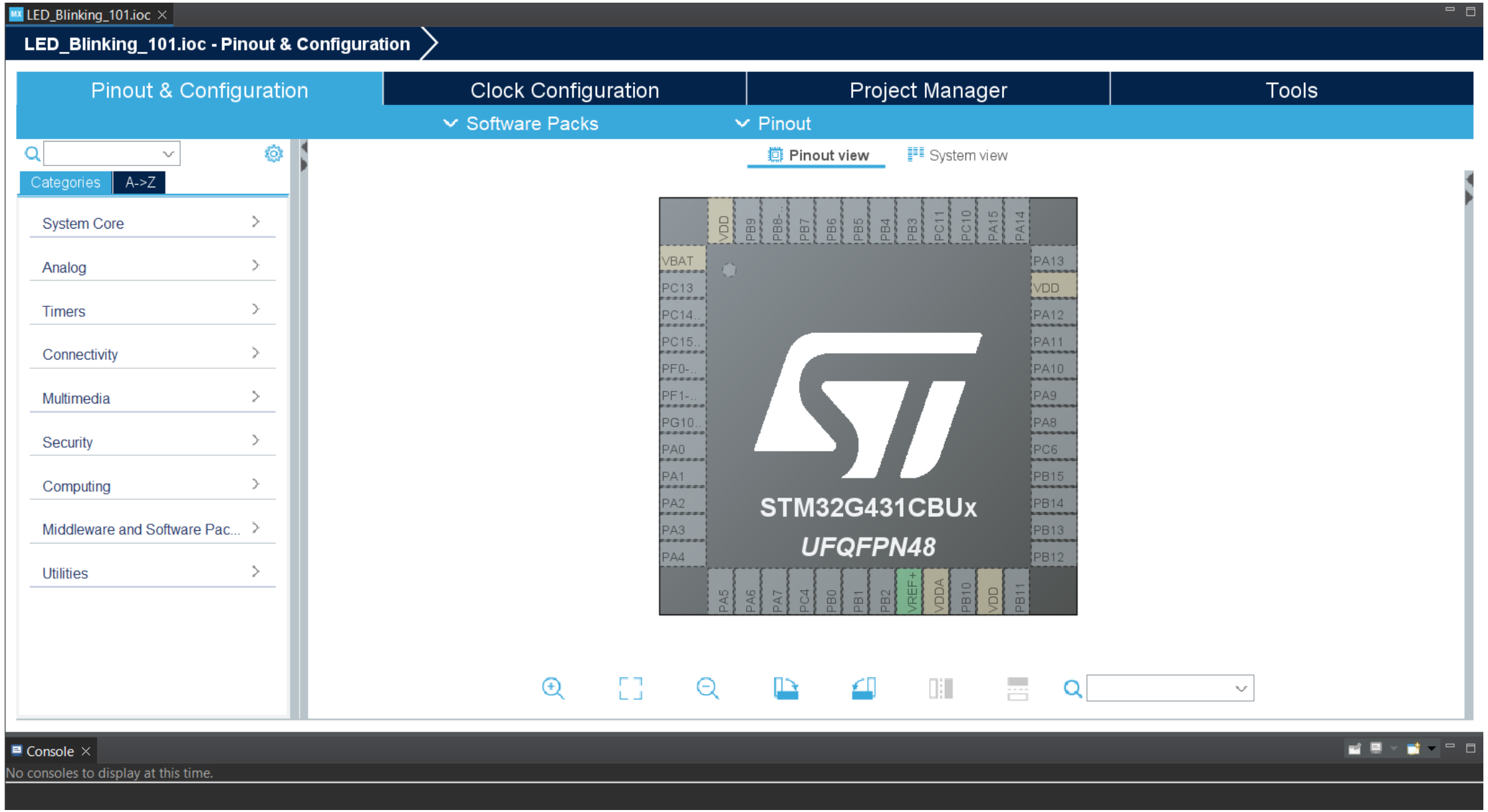


Setting up the environment

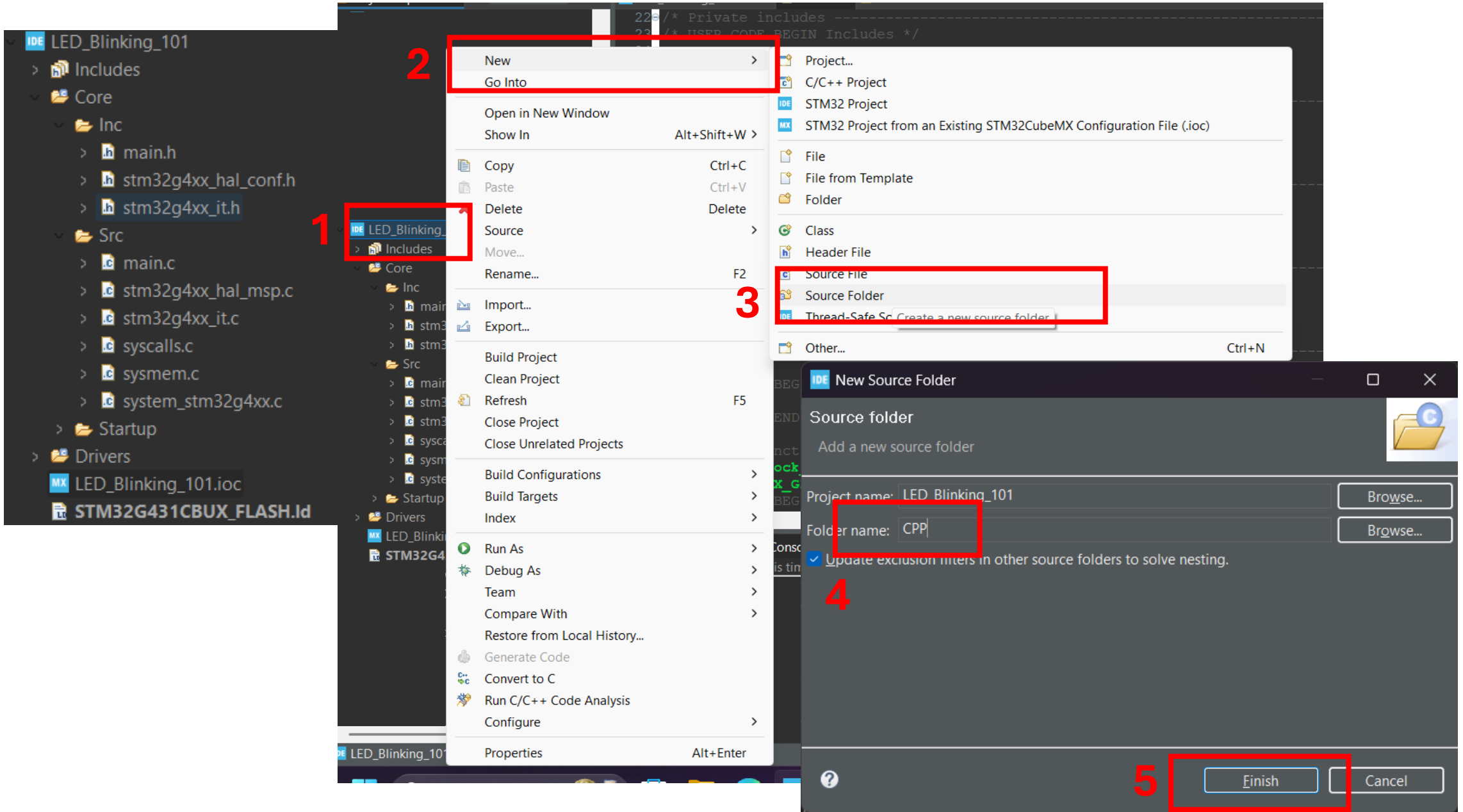




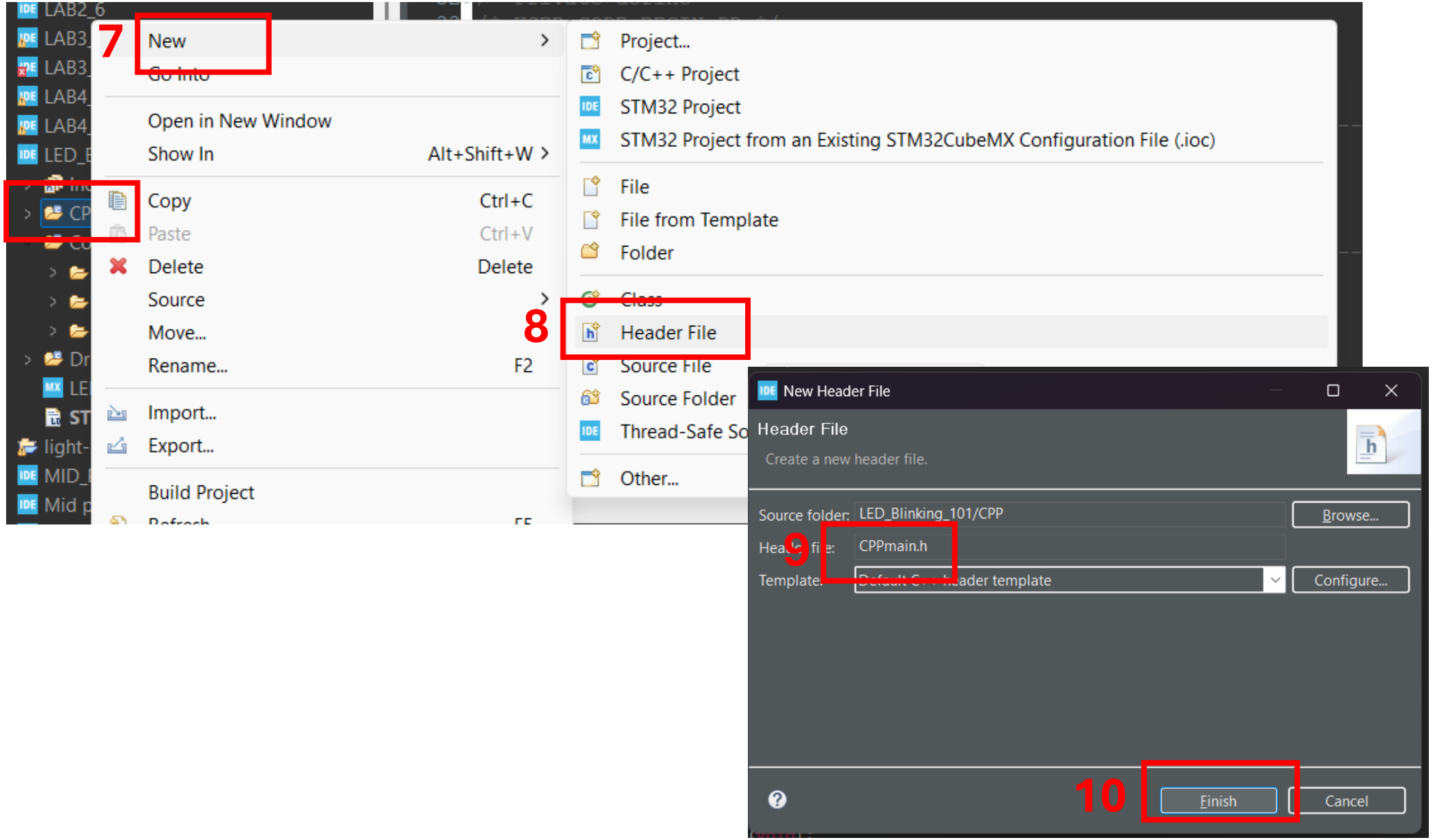
Setting up the environment



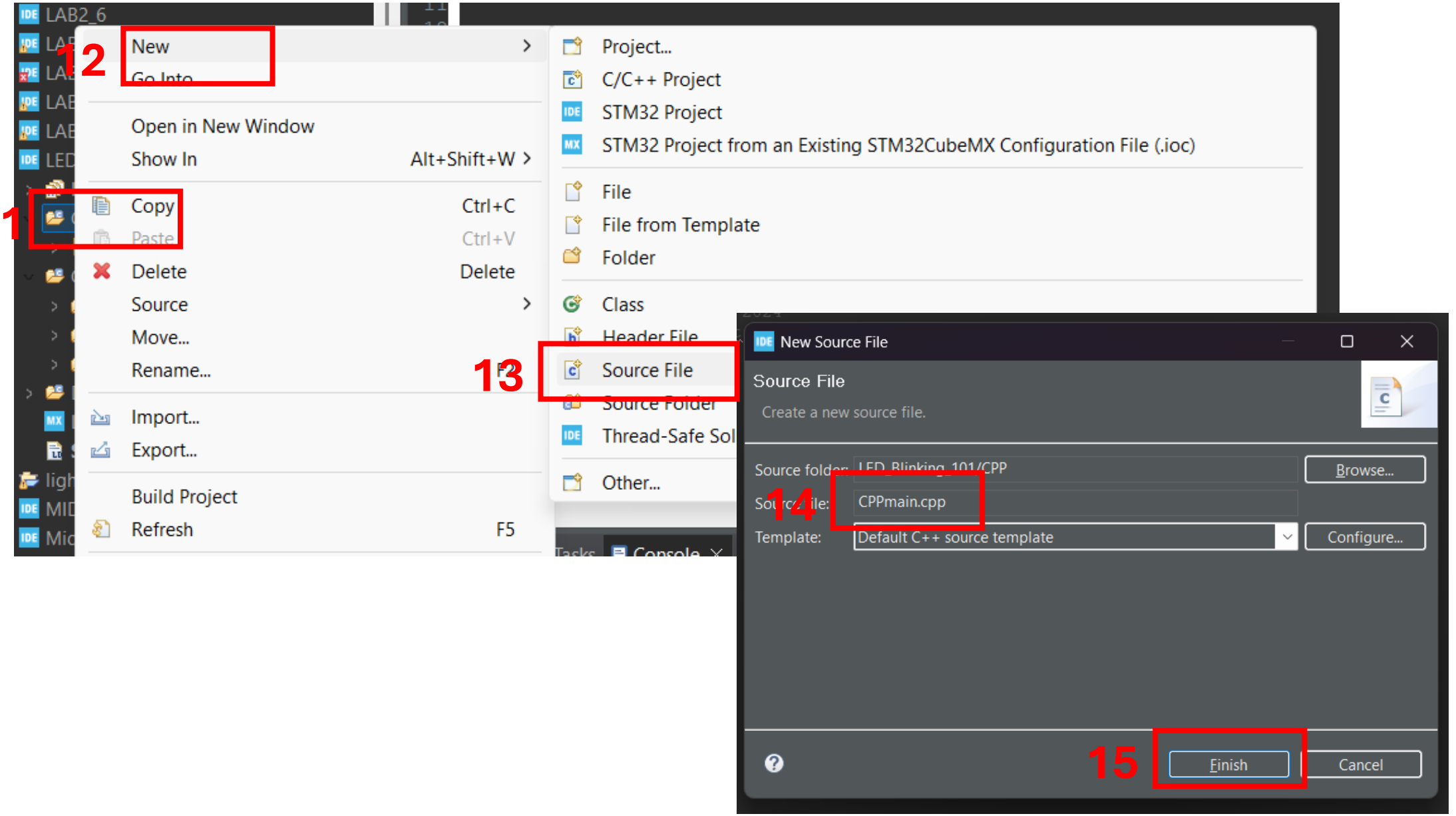
Setting up the environment



Setting up the environment



Setting up the environment



Setting up the environment

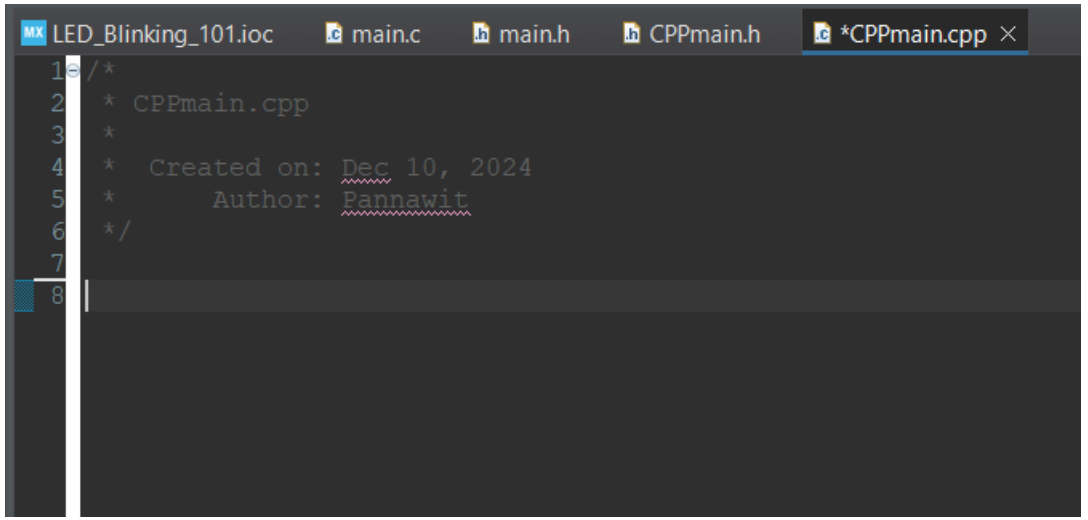
LED_Blinking_101.ioc | main.c | main.h | CPPmain.h

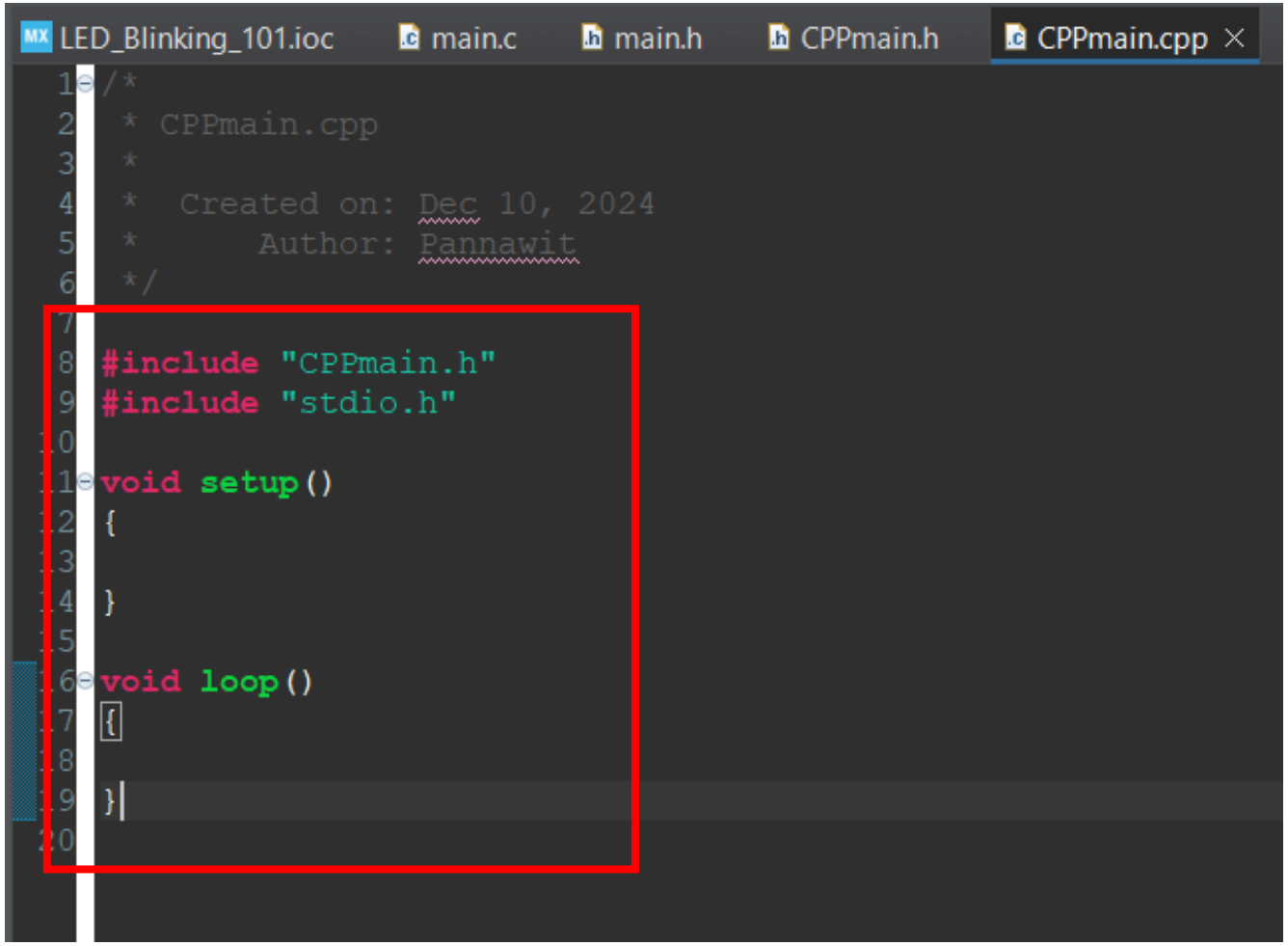
```
/*
 * CPPmain.h
 *
 *  Created on: Dec 10, 2024
 *      Author: Pannawit
 */

#ifndef CPPMAIN_H_
#define CPPMAIN_H_




#endif /* CPPMAIN_H_ */
```

LED_Blinking_101.ioc | main.c | main.h | CPPmain.h | CPPmain.cpp

```
/*
 * CPPmain.h
 *
 *  Created on: Dec 10, 2024
 *      Author: Pannawit
 */

#ifndef CPPMAIN_H_
#define CPPMAIN_H_


#include "main.h"
#ifdef __cplusplus
extern "C" {
#endif

void setup();
void loop();
void gpio_init();

#ifdef __cplusplus
}
#endif


#endif /* CPPMAIN_H_ */
```

Setting up the environment





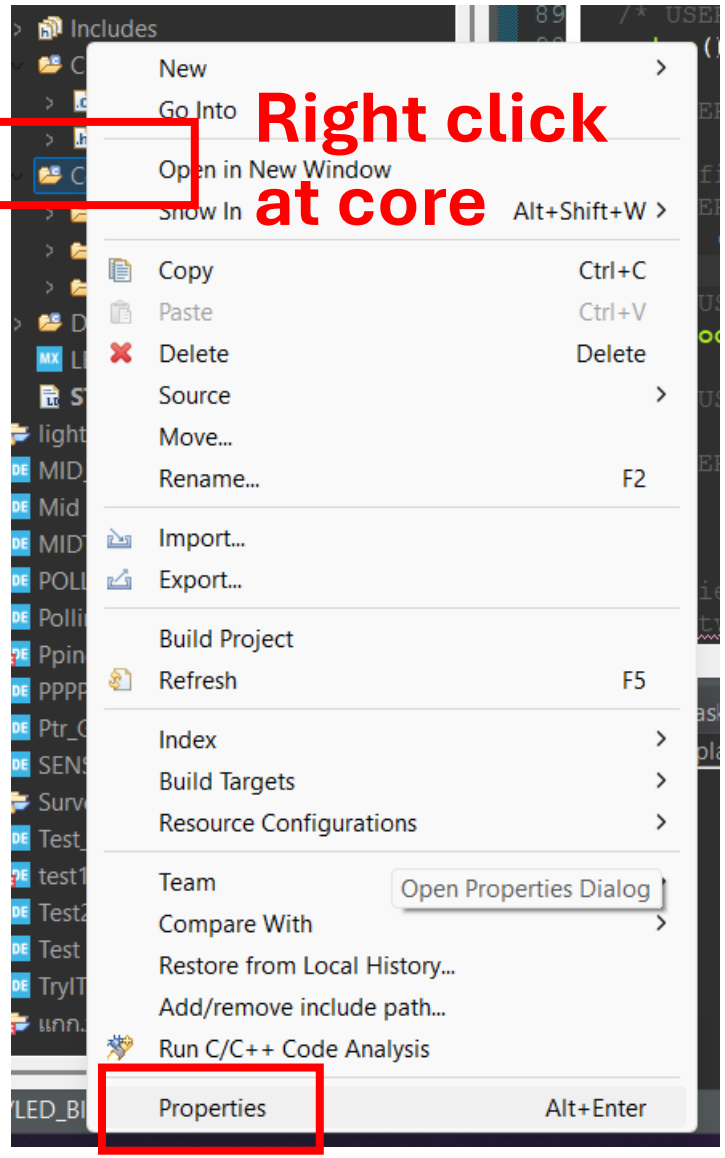
Setting up the environment

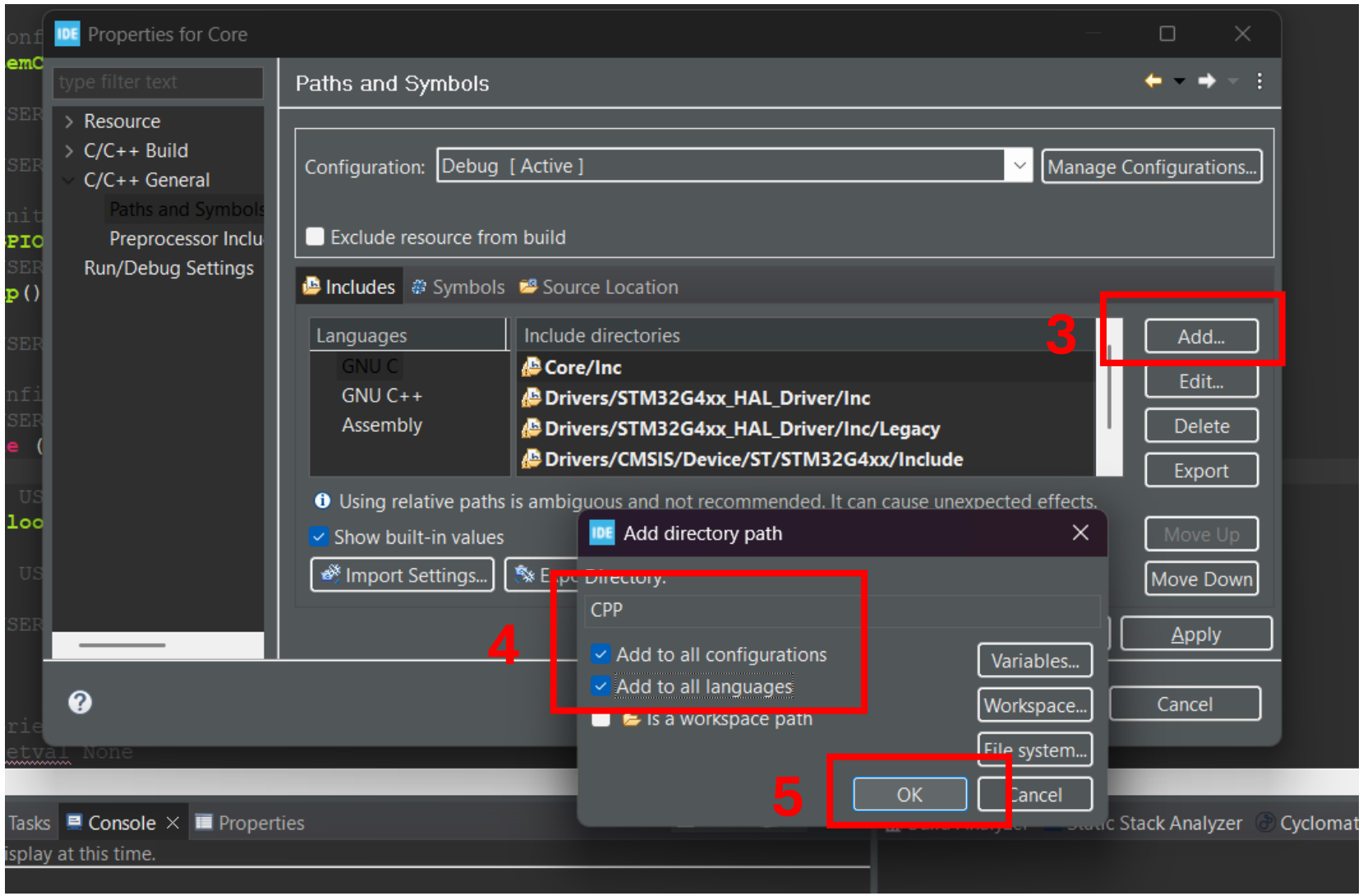
LED_Blinking_101.ioc | main.c | main.h

```
16    ******************************
17    */
18  /* USER CODE END Header */
19  /* Includes ------------------------
20  #include "main.h"
21
22  /* Private includes ----------------
23  /* USER CODE BEGIN Includes */
24  #include "CPPmain.h"
25  /* USER CODE END Includes */
26
27  /* Private typedef -----------------
28  /* USER CODE BEGIN PTD */
```

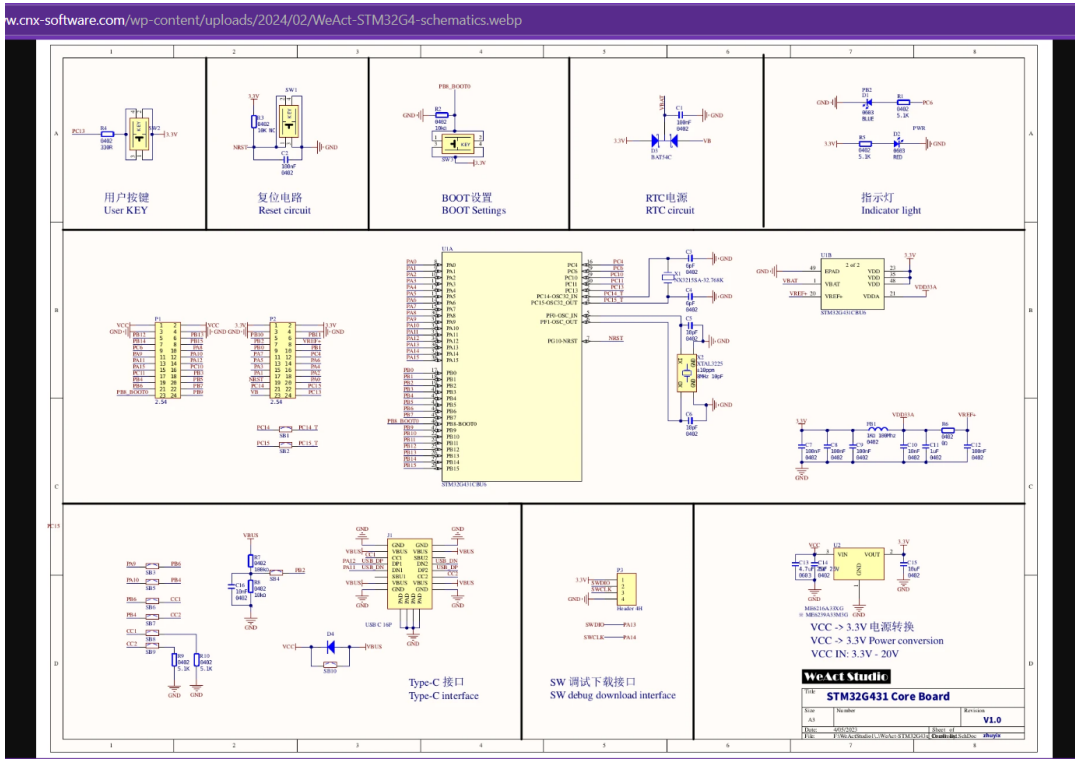
```
87    /* Initialize all configured peripherals
88    MX_GPIO_Init();
89    /* USER CODE BEGIN 2 */
90    setup();
91
92    /* USER CODE END 2 */
93
94    /* Infinite loop */
95    /* USER CODE BEGIN WHILE */
96    while (1)
97    {
98      /* USER CODE END WHILE */
99      loop();
100
101     /* USER CODE BEGIN 3 */
102   }
103   /* USER CODE END 3 */
104 }
```

Setting up the environment





Important thing for embedded system



life.augmented

STM32G431x6 STM32G431x8 STM32G431xB

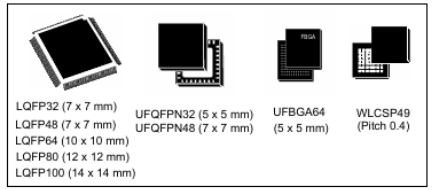
Arm® Cortex®-M4 32-bit MCU+FPU, 170 MHz /213 DMIPS, up to 128 KB Flash, 32 KB SRAM, rich analog, math accelerator

Datasheet - production data

Features

Includes ST state-of-the-art patented technology

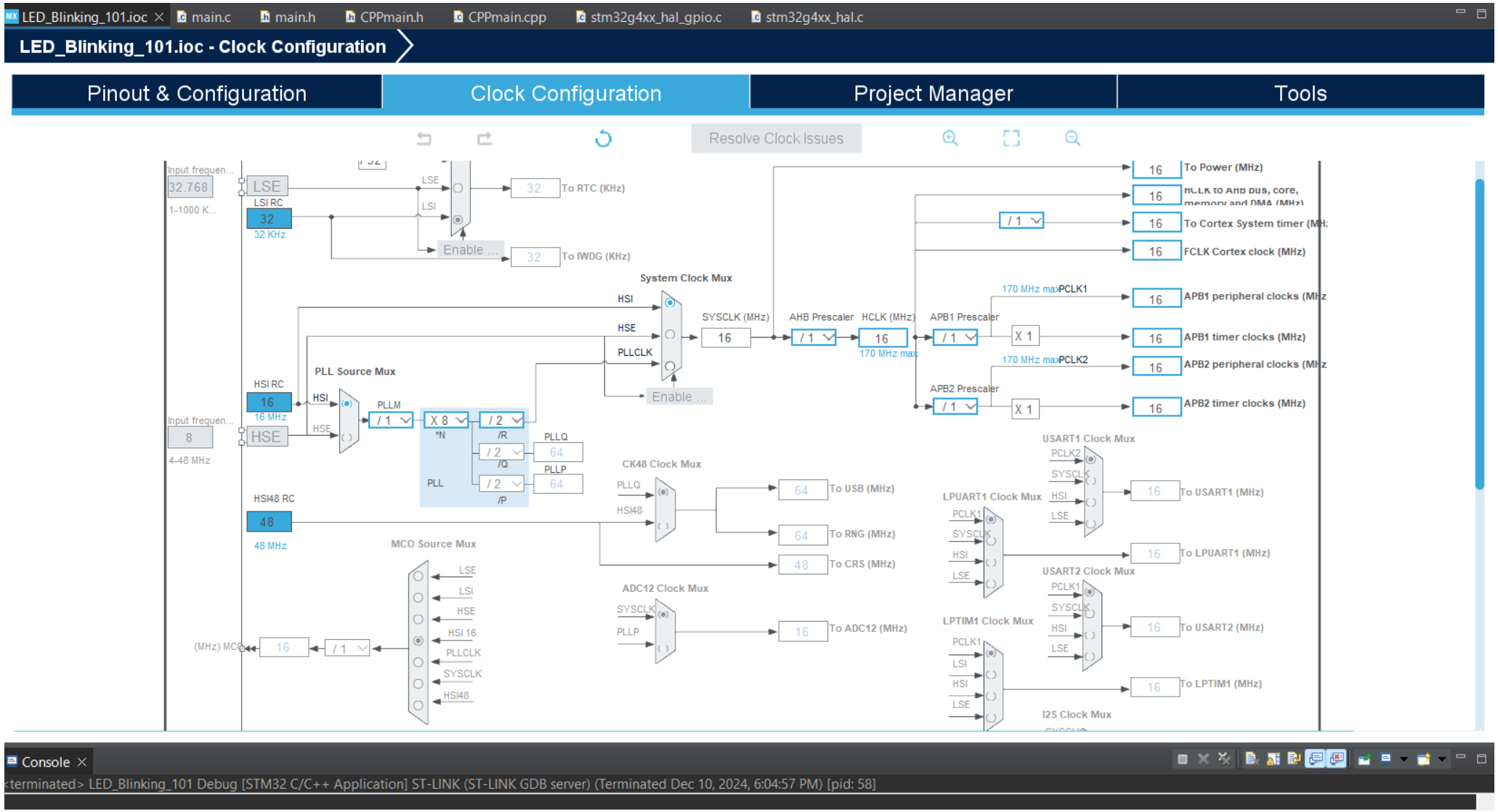
- Core: Arm® 32-bit Cortex®-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator) allowing 0-wait-state execution from Flash memory, frequency up to 170 MHz with 213 DMIPS, MPU, DSP instructions
- Operating conditions:
  - V_{DD}, V_{DDA} voltage range: 1.71 V to 3.6 V
- Mathematical hardware accelerators
  - CORDIC for trigonometric functions acceleration
  - FMAC: filter mathematical accelerator
- Memories
  - 128 Kbytes of Flash memory with ECC support, proprietary code readout protection (PCROP), securable memory area, 1 Kbyte OTP
  - 22 Kbytes of SRAM, with hardware parity check implemented on the first 16 Kbytes
  - Routine booster: 10 Kbytes of SRAM on



- Clock management
  - 4 to 48 MHz crystal oscillator
  - 32 kHz oscillator with calibration
  - Internal 16 MHz RC with PLL option (± 1%)
  - Internal 32 kHz RC oscillator (± 5%)
- Up to 86 fast I/Os
  - All mappable on external interrupt vectors
  - Several I/Os with 5 V tolerant capability
- Interconnect matrix
- 12-channel DMA controller
- 2 x ADCs 0.25 µs (up to 23 channels). Resolution up to 16-bit with hardware oversampling, 0 to 3.6 V conversion range
- 4 x 12-bit DAC channels
  - 2 x buffered external channels 1 MSPS

Datasheet, schematic, and pinout are the most important thing in STM32 project

Additional setup for environment

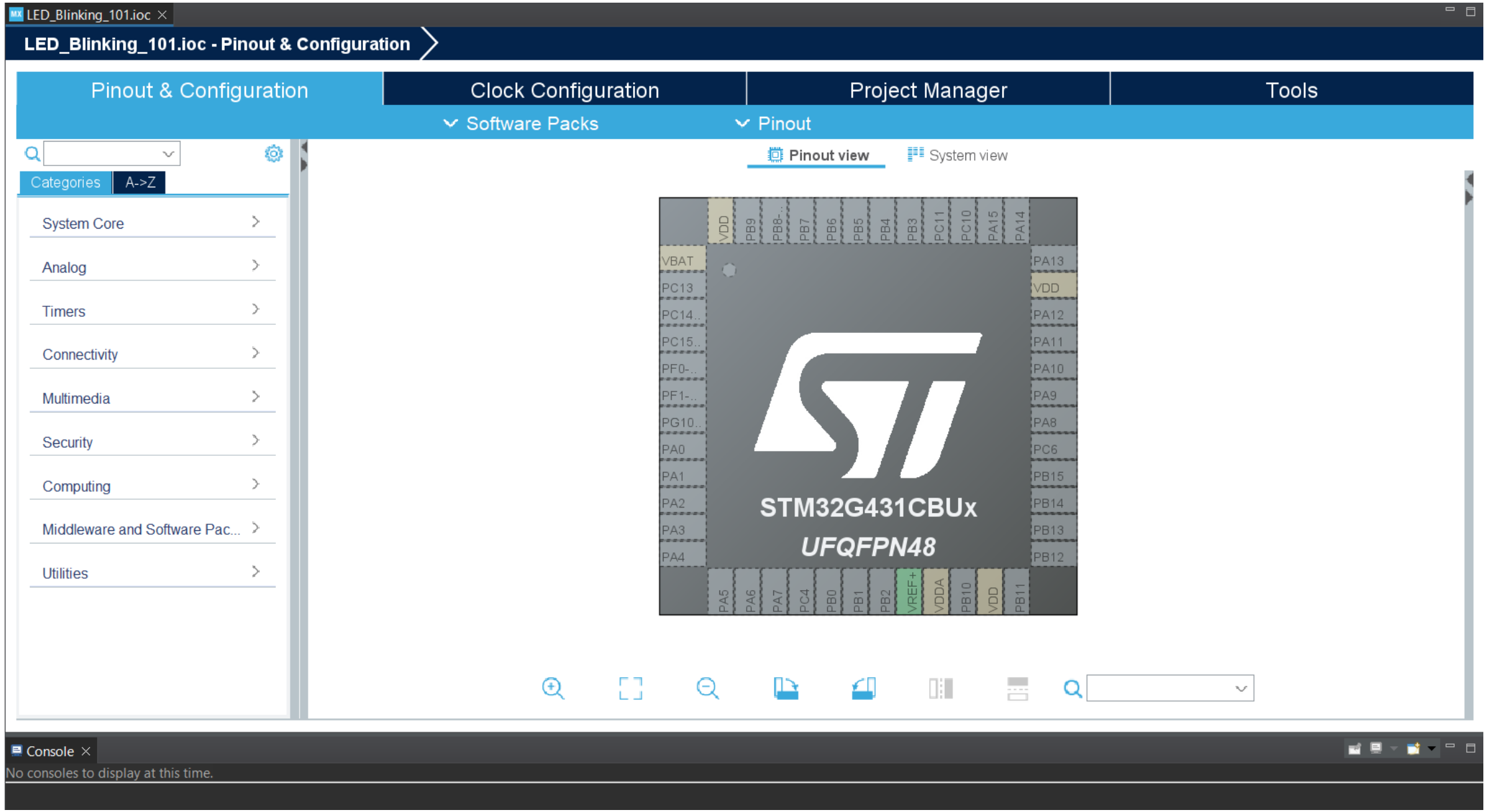


You can setup the clock configuration for any peripheral required.

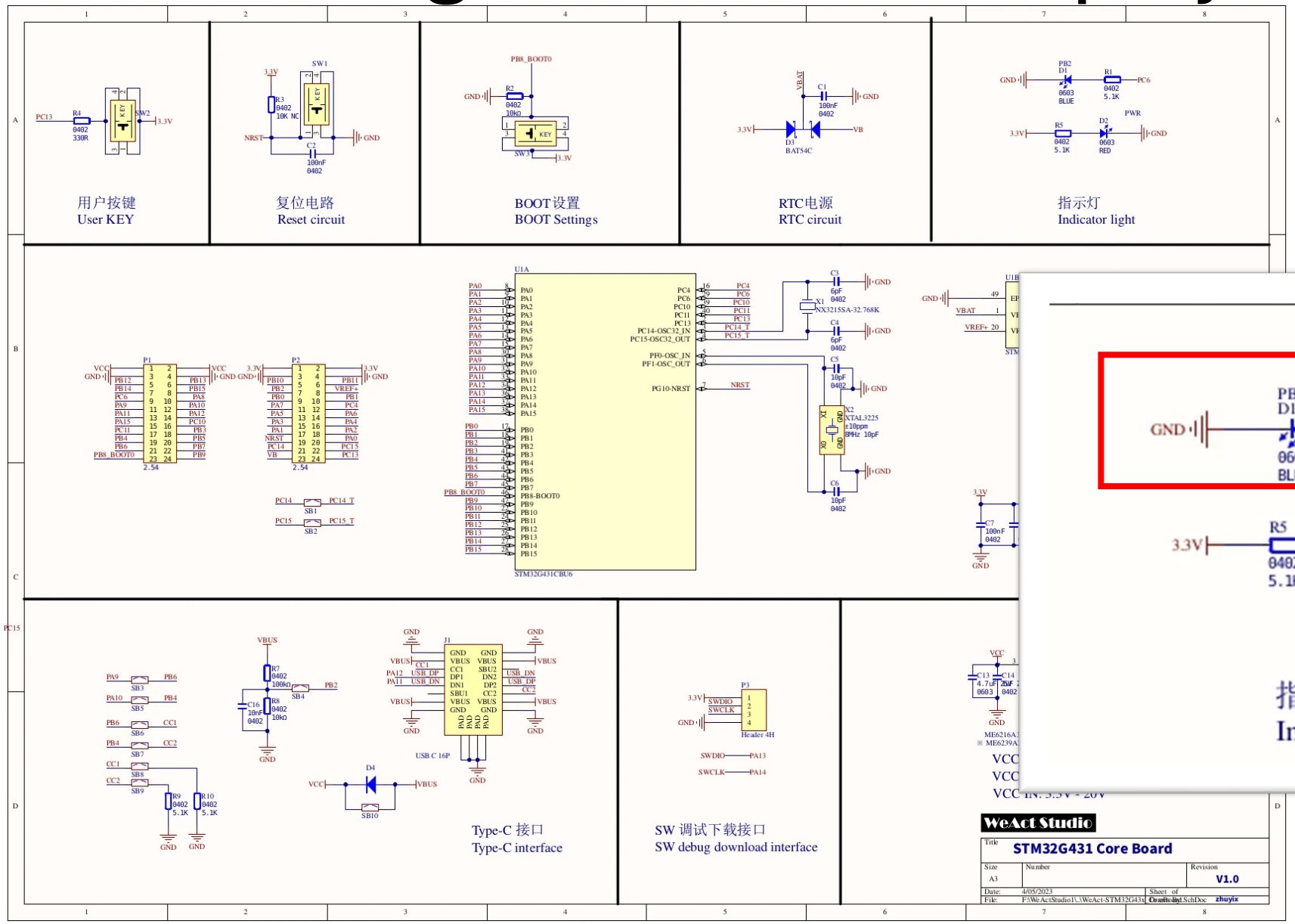
Basics of Embedded Programming

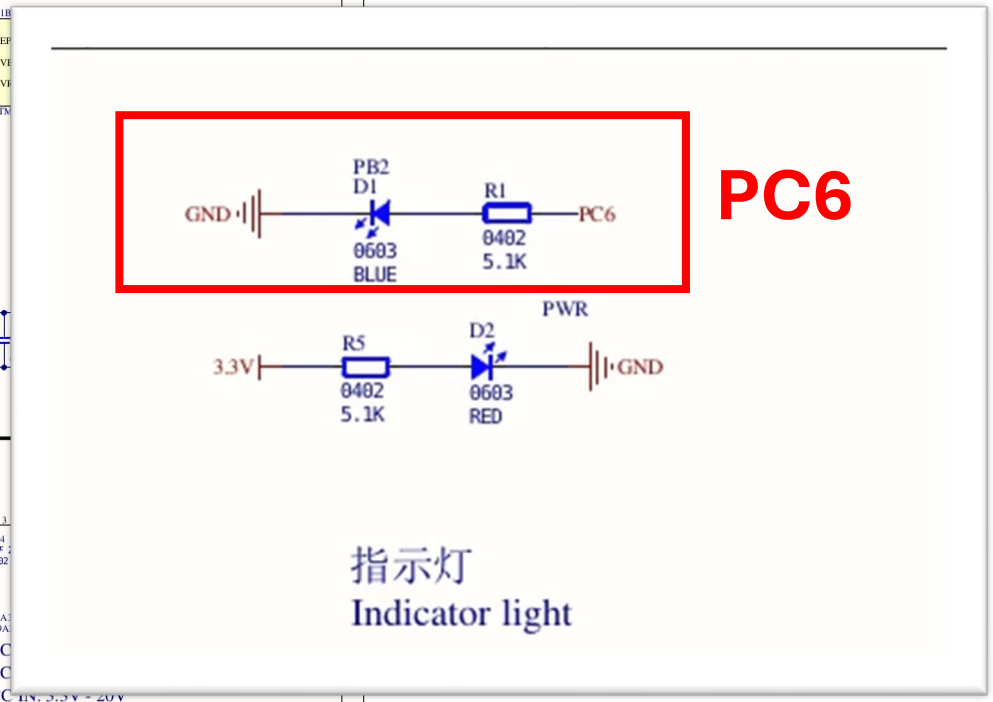
GPIO and your first project in STM32 with LED Blinking

LED Blinking for first STM32 project



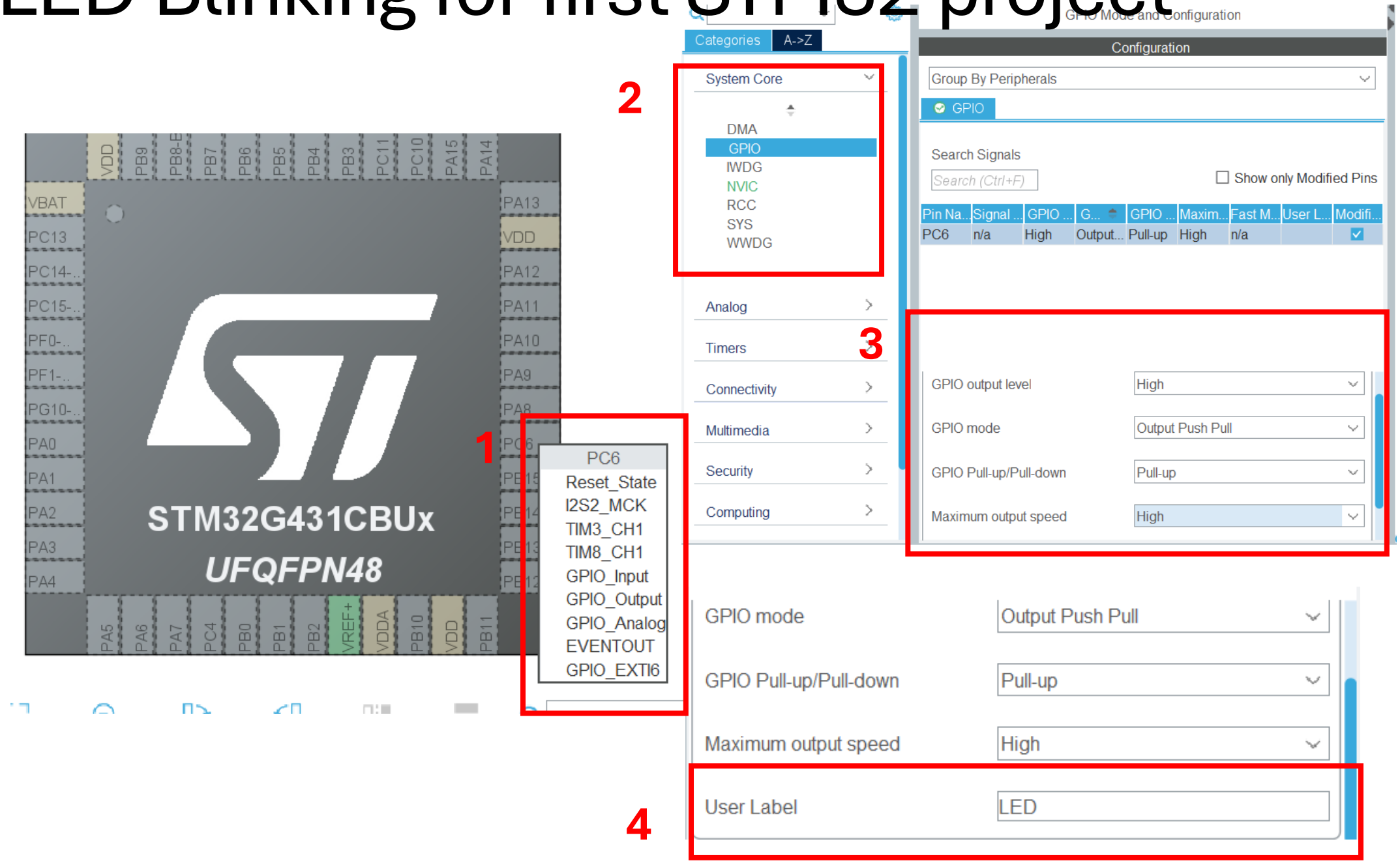
LED Blinking for first STM32 project



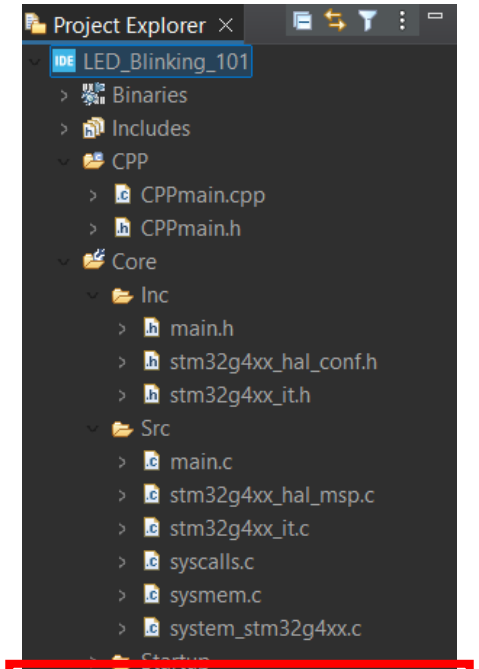


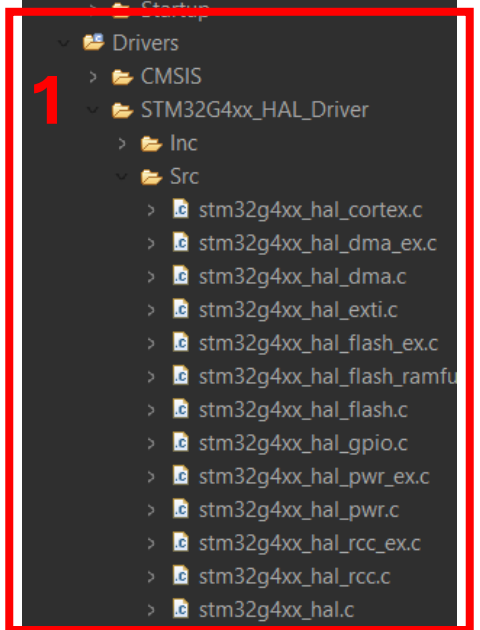
PC6

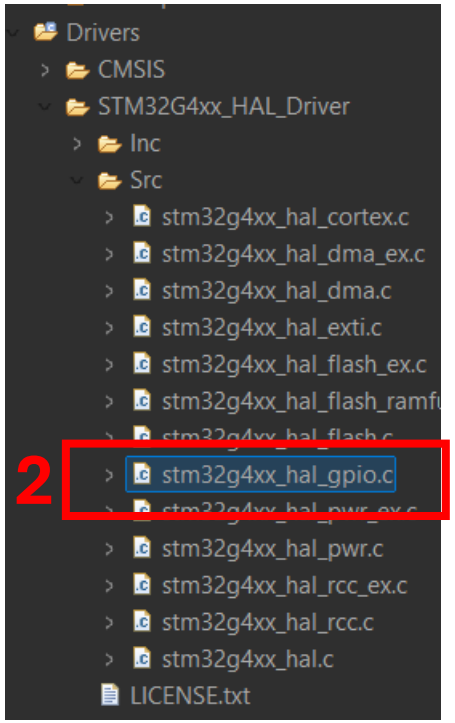
LED Blinking for first STM32 project

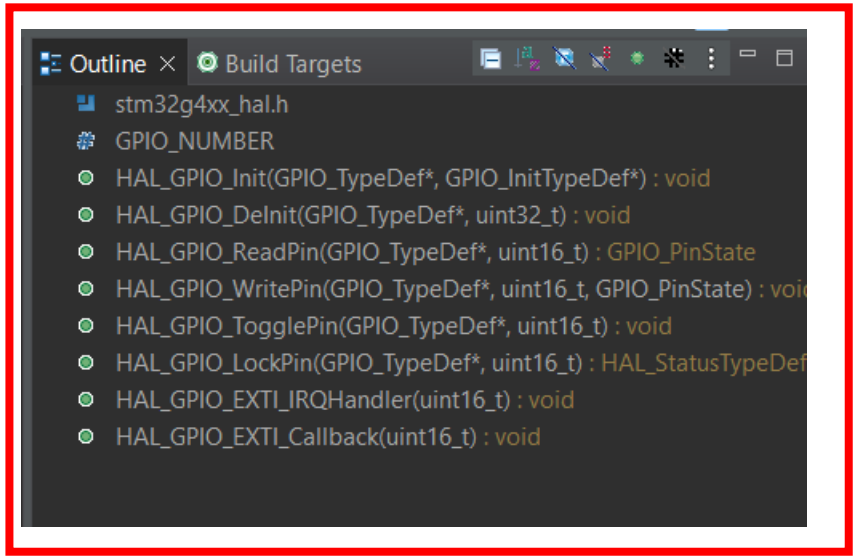


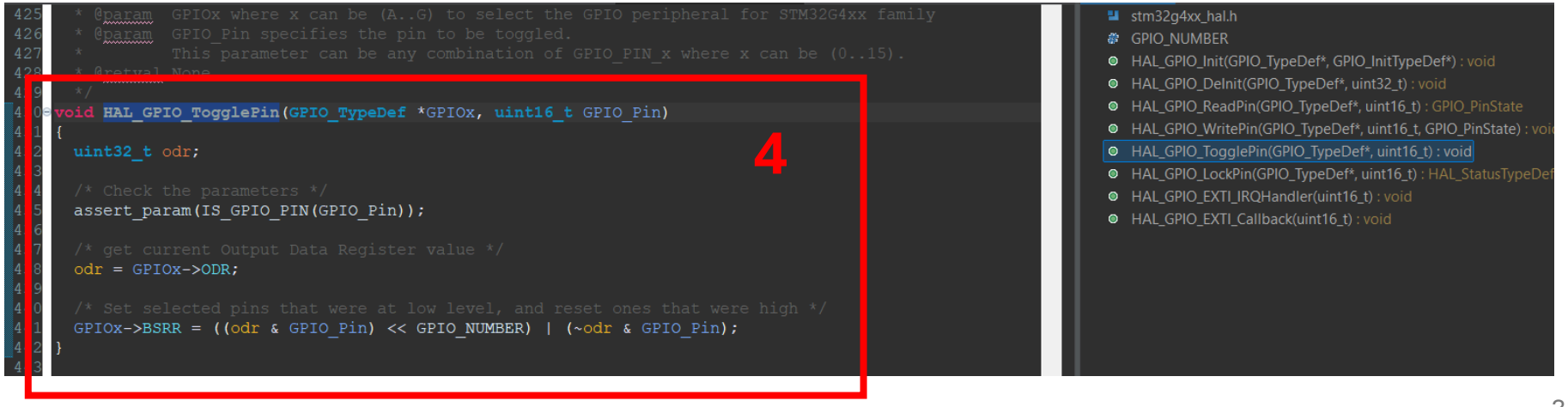
LED Blinking for first STM32 project











LED Blinking for first STM32 project

```
422
423 /**
424   * @brief  Toggle the specified GPIO pin.
425   * @param  GPIOx where x can be (A..G) to select the GPIO peripheral for STM32G4xx family
426   * @param  GPIO_Pin specifies the pin to be toggled.
427   *         This parameter can be any combination of GPIO_PIN_x where x can be (0..15).
428   * @retval None
429   */
430 void HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
431 {
432   uint32_t odr;
433
434   /* Check the parameters */
435   assert_param(IS_GPIO_PIN(GPIO_Pin));
436
437   /* get current Output Data Register value */
438   odr = GPIOx->ODR;
439
440   /* Set selected pins that were at low level, and reset ones that were high */
441   GPIOx->BSRR = ((odr & GPIO_Pin) << GPIO_NUMBER) | (~odr & GPIO_Pin);
442 }
443
```

LED Blinking for first STM32 project

LED_Blinking_101.ioc | main.c | main.h | CPPmain.h | *CPPmain.cpp | stm32g4xx_hal_gpio.c

```
/*
 * CPPmain.cpp
 *
 *  Created on: Dec 10, 2024
 *      Author: Pannawit
 */

#include "CPPmain.h"
#include "stdio.h"

void setup()
{

}

void loop()
{
    HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);
}
```

main.h

```
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define LED_Pin GPIO_PIN_6
#define LED_GPIO_Port GPIOC

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */
```

LED Blinking for first STM32 project

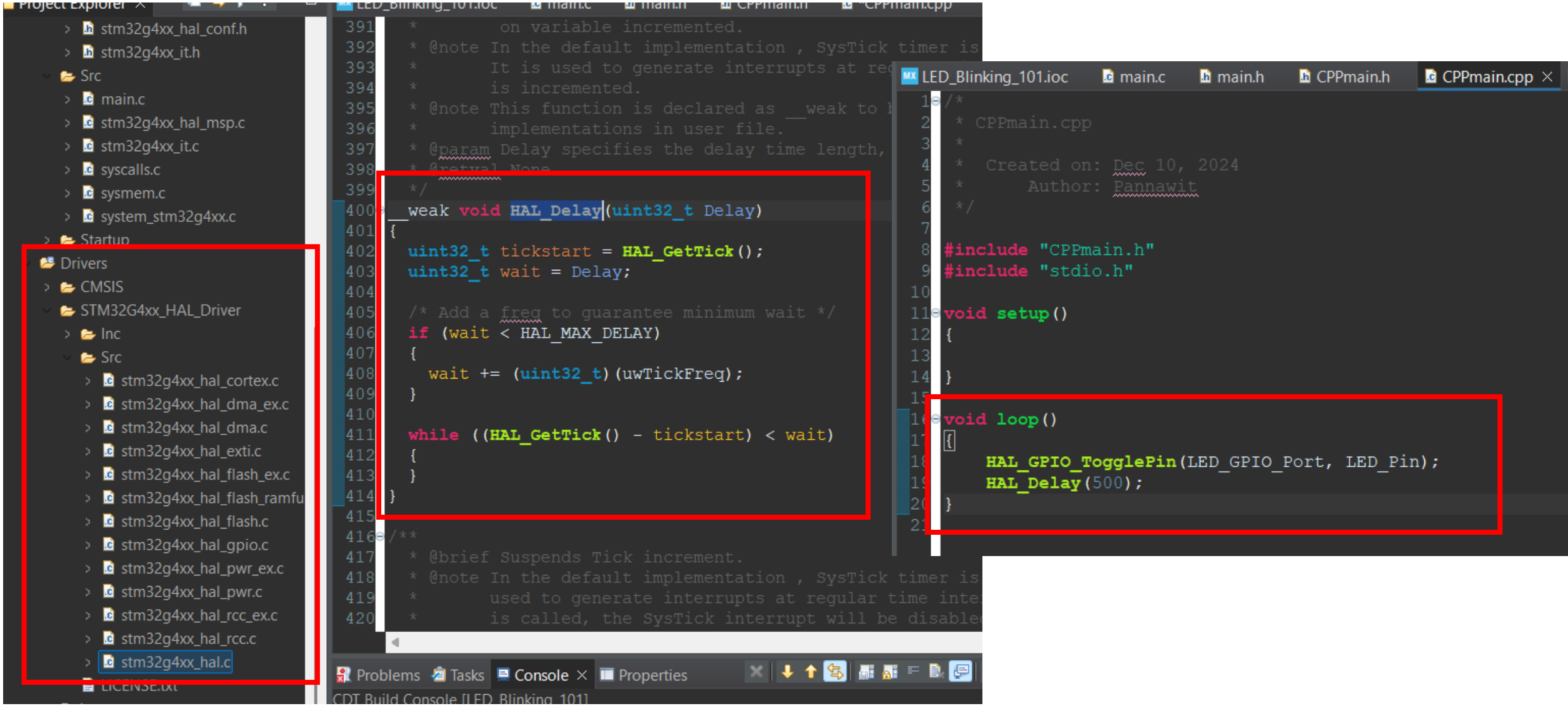
LED_Blinking_101.ioc | main.c | main.h | CPPmain.h

```
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ----------
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines ------------------------
#define LED_Pin GPIO_PIN_6
#define LED_GPIO_Port GPIOC

/* USER CODE BEGIN Private defines */
```

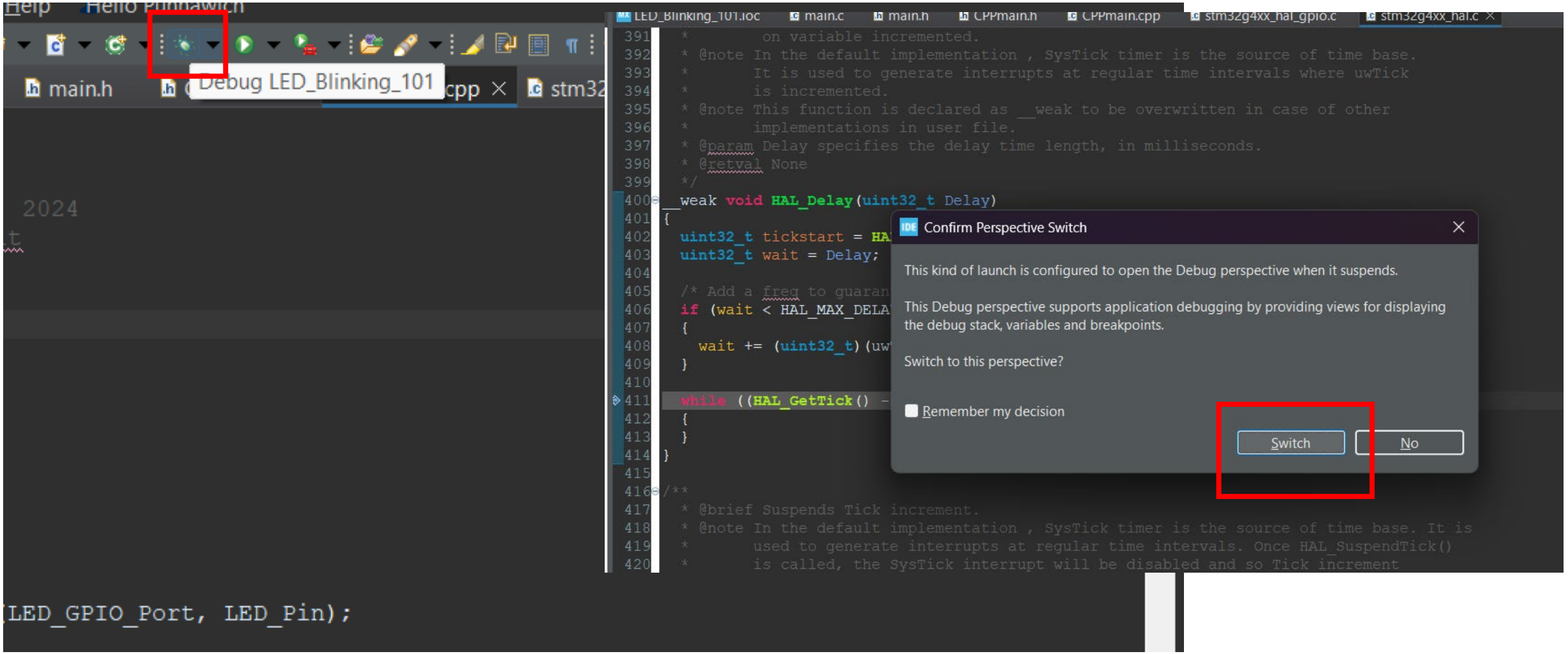
LED_Blinking_101.ioc | main.c | main.h | CPPmain.h | CPPmain.cpp

```
/*
 * CPPmain.cpp
 *
 *  Created on: Dec 10, 2024
 *      Author: Pannawit
 */

#include "CPPmain.h"
#include "stdio.h"

void setup()
{

}

void loop()
{
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
}
```

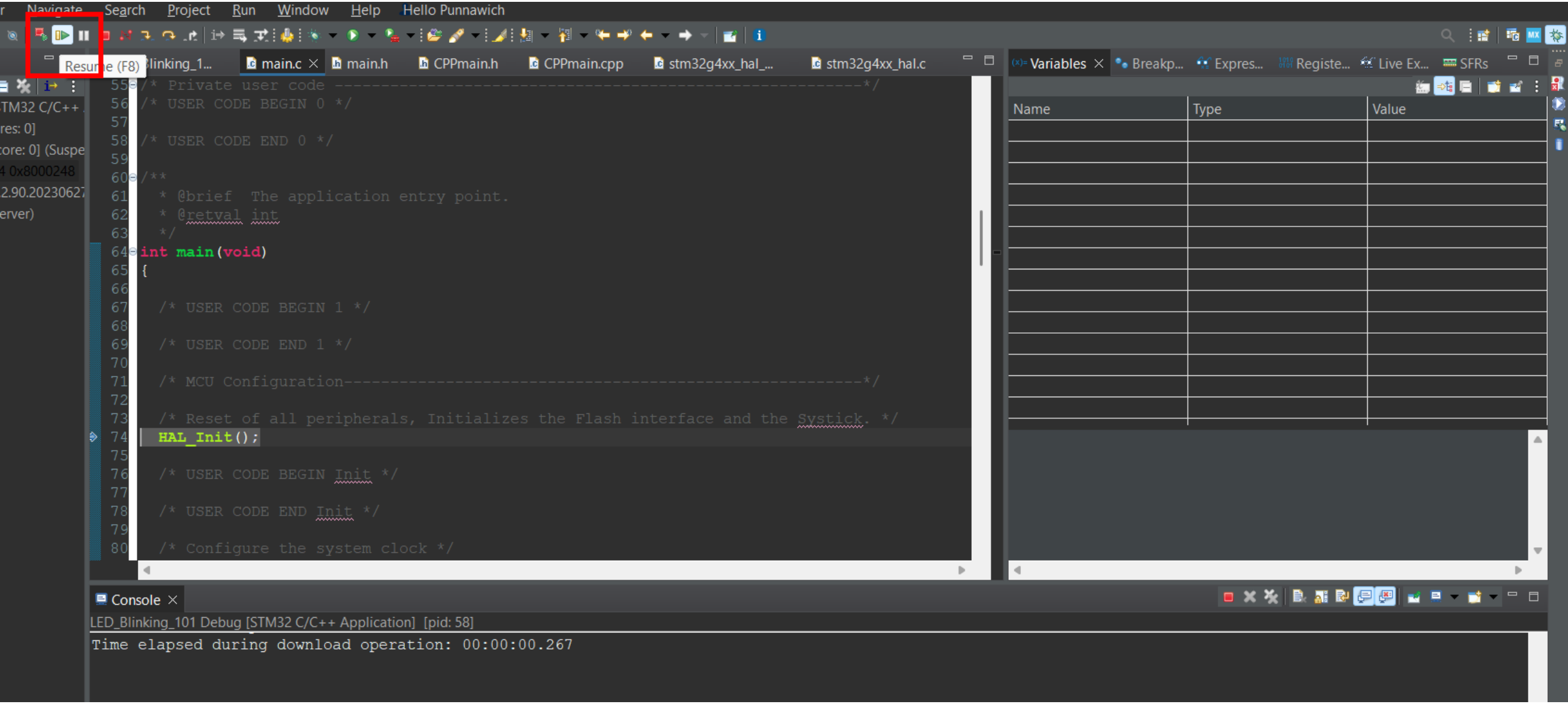
LED Blinking for first STM32 project



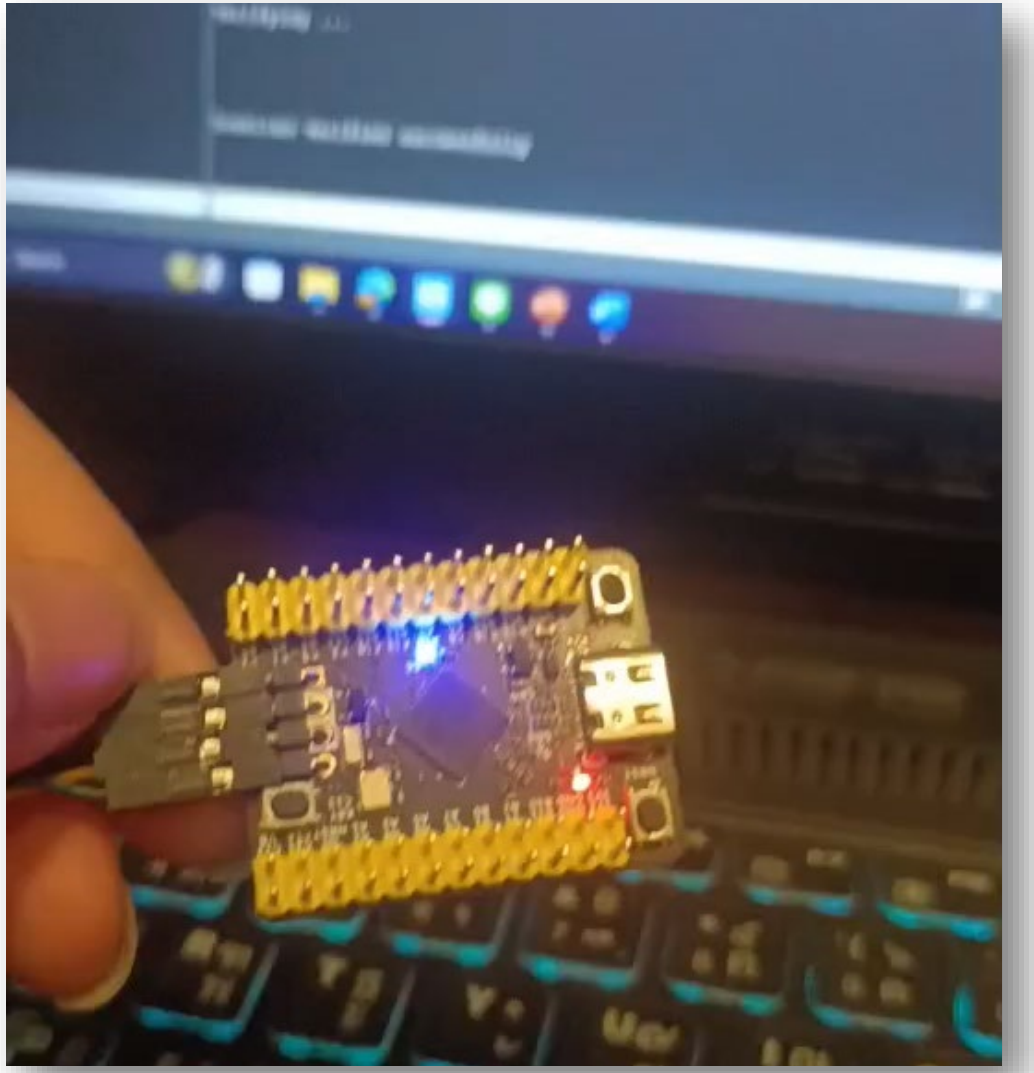
LED Blinking for first STM32 project



LED Blinking for first STM32 project

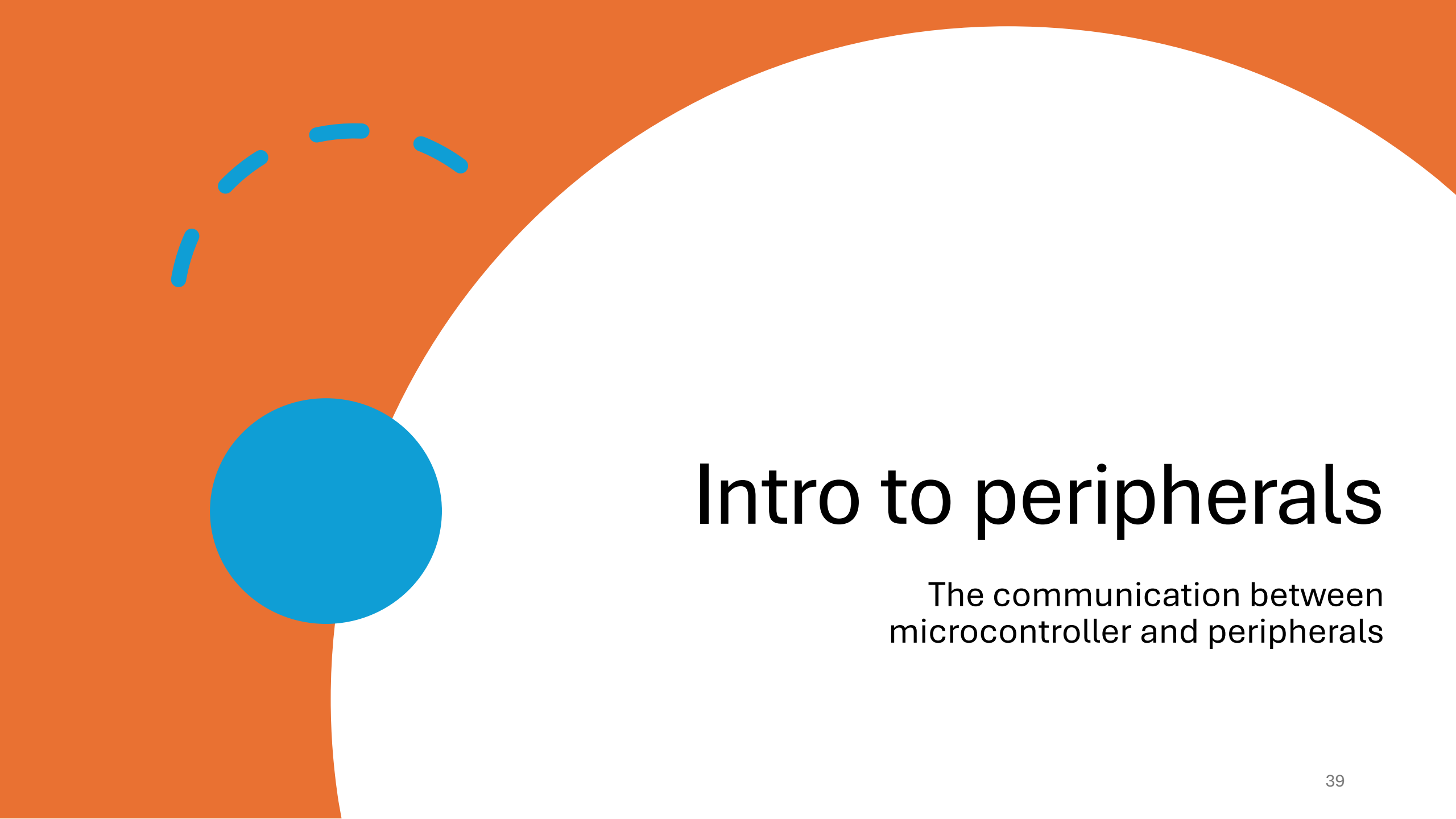


LED Blinking for first STM32 project



Congratulation

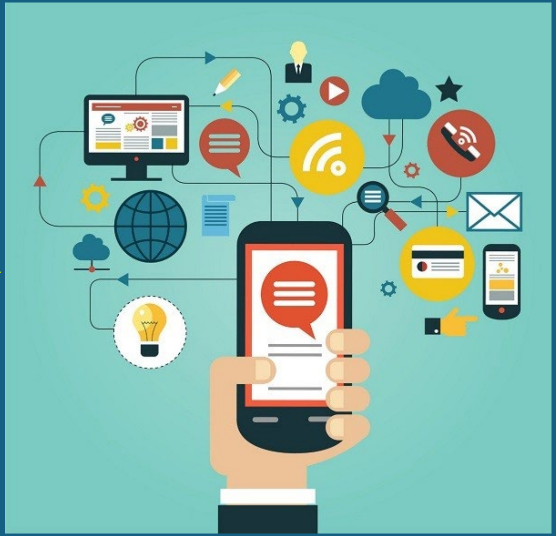
Now, you guys can do the STM32 project



Intro to peripherals

The communication between microcontroller and peripherals

CĂyụuvýśửɔĂẢvừừºửŝĝẻävửĂ Ĵỷwâvĝvựź



Communication is the key

Ò ř ẻ ê Ă È ì Ă Ễ ś ụ Ă g v Ễ ỉ ỷ

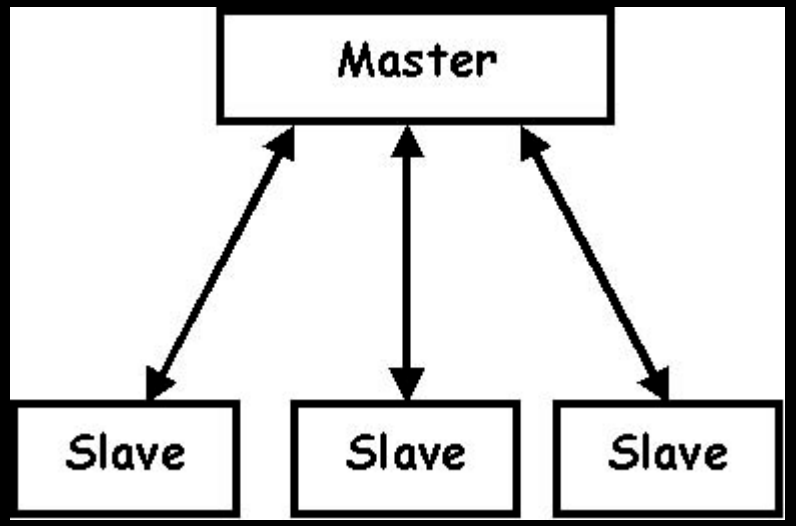
- NJĂKŇĂ9ĂNJĻĂKŇ
- Ề Ả
- Ļ Ĵ Ề
- ẢĂĠĂ ỉ Ů ỷ 7

SOME WORDs

Duplex

Simplex Half Duplex Full Duplex

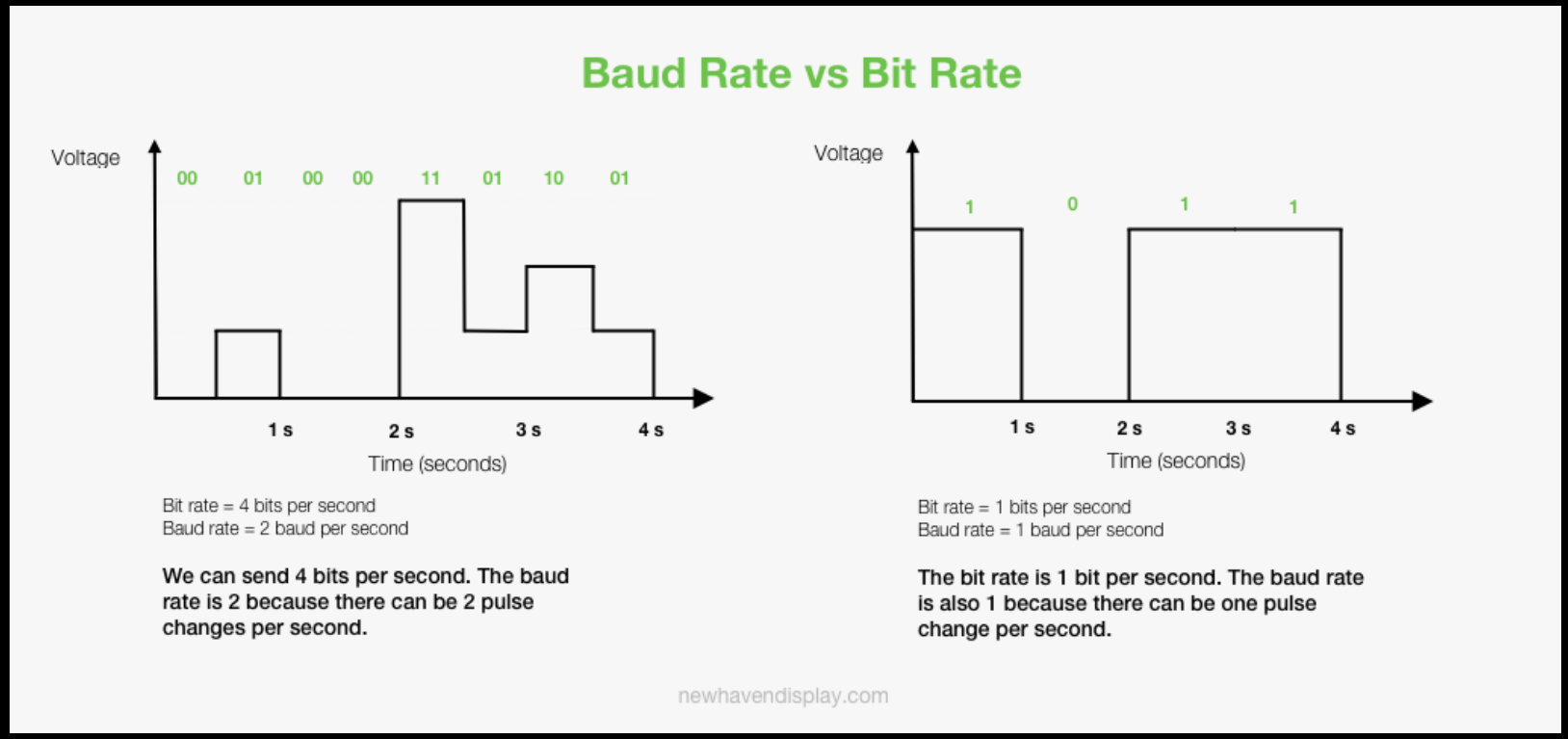
Master-slave



SOME WORDs

"Baud rate"

= is the measure of the number of ***changes*** to the signal (per second) that propagate through a transmission medium



UART / USART

Universal Synchronous/Asynchronous Receiver/Transmitter

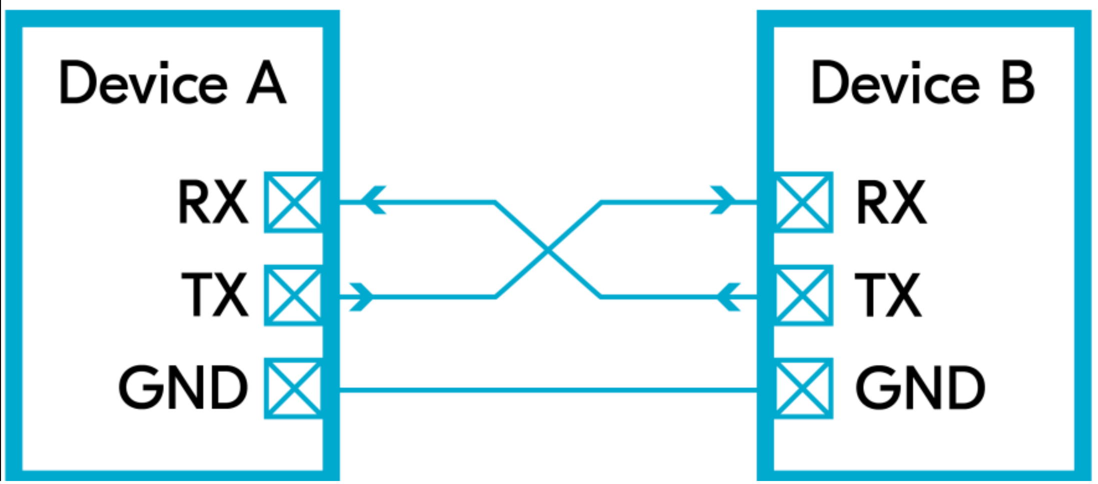
Basics Serial communication, full-duplex, point-to-point.

Key parameters: baud rate, data bits, stop bits, parity.

Applications

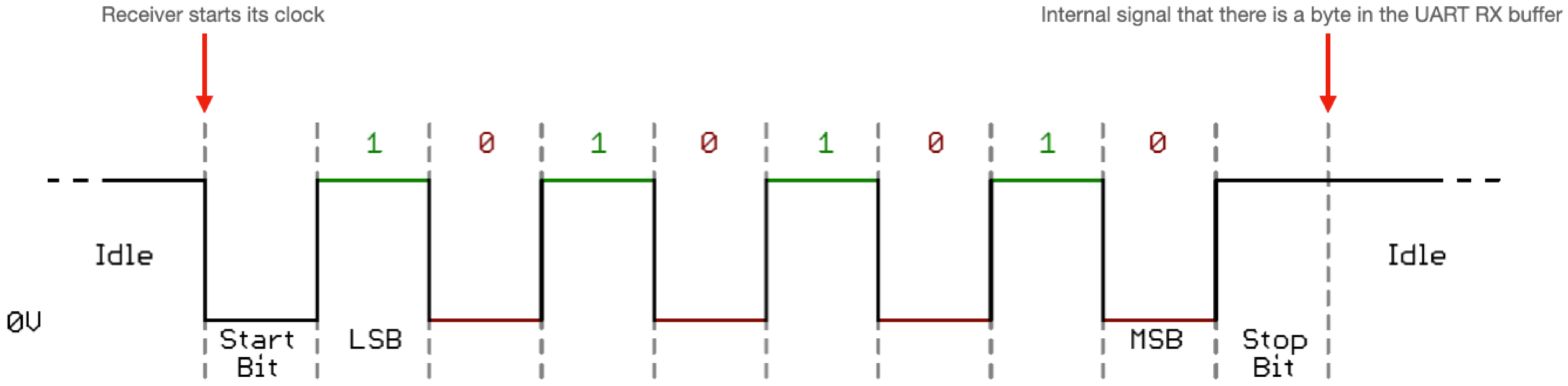
Debugging (e.g., sending data to a terminal),

communication between a microcontroller and a peripheral.

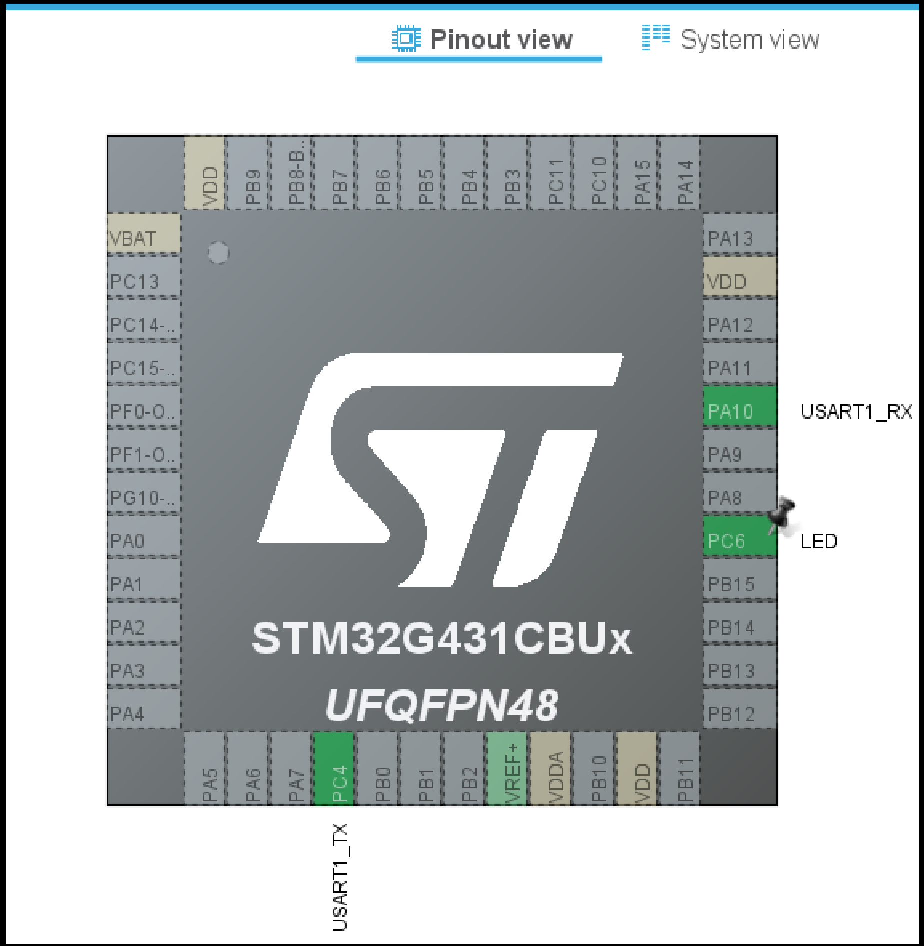


Between 2 devices their baud rate must be the same!

Ļĉẻửĩ ẻỷĩ Ăğẻºĩ Ăỷẻâ
—??6Ă#\ \??6Ă#—¯??6ĂX—\??6Ă--??6Ă##-¯??Ăẻửĩ Ăư vỷì



Mini task




```c
#include "CPPmain.h"
#include "stdio.h"

extern UART_HandleTypeDef huart1;

uint8_t data[10];
uint8_t time = 0;

void setup()
{
	HAL_UART_Receive_IT(&huart1, data, sizeof(data));
}

void loop()
{
	time = (data[0] << 8) | data[1];

	HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
	HAL_Delay(time);

	HAL_UART_Receive(&huart1, data, sizeof(data), 1000);
}
```

Ňýẻửźừ sŕĂŕ ì Ăừ ì Ă ì ụẻÝĂv Ăì ôĂŕ ì Ăg ụẻửừ ửoĂ ị ì ýśvî

I2C

Inter-Integrated Circuit

Basics

Two-wire protocol (SDA and SCL).

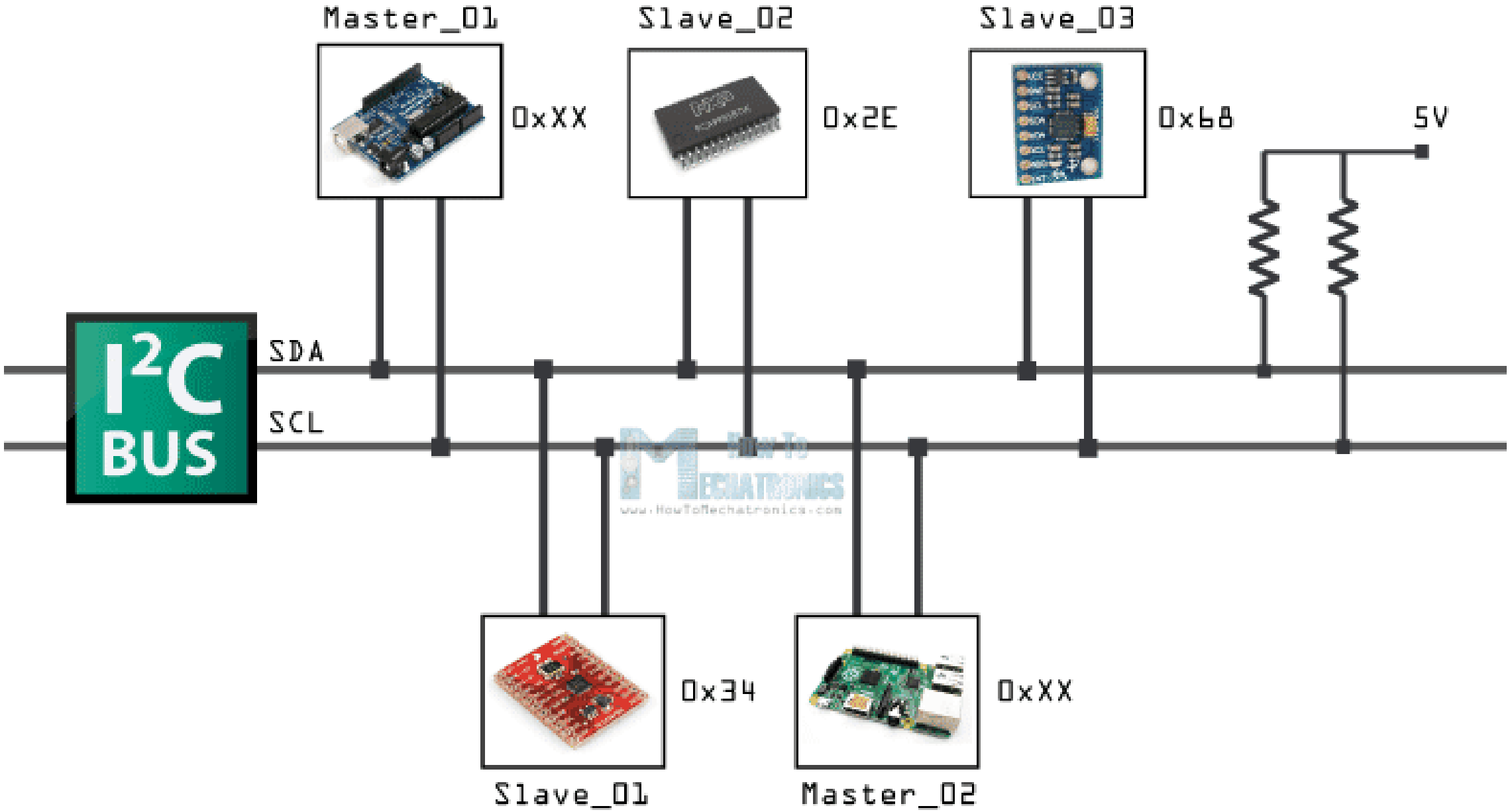
Addressing mechanism.

How I2C Works

Start and stop conditions, acknowledgment, and data transfer.

Applications in Robotics

Sensor data acquisition, interfacing IMUs or temperature sensors



https://www.google.com/url?sa=i&url=https%3A%2F%2Fai.thestempedia.com%2Fdocs%2Fevive%2Fevive-technical-specifications%2Farduino-core-interface%2Fi2c-communication%2F&psig=AOvVaw1JgjS6_62YiE4mU7gi8J7e&ust=1734295687173000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCKCGkISRqIoDFQAAAAAdAAAAABAS

Mini task

Try to communicate with this sensor

ROHM SEMICONDUCTOR

Technical Note

Ambient Light Sensor IC Series

Digital 16bit Serial Output Type Ambient Light Sensor IC

BH1750FVI No.09046EBT01

●Descriptions

BH1750FVI is an digital Ambient Light Sensor IC for I²C bus interface. This IC is the most suitable to obtain the ambient light data for adjusting LCD and Keypad backlight power of Mobile phone. It is possible to detect wide range at High resolution. (1 - 65535 lx).

●Features

1) I²C bus Interface (f / s Mode Support)
2) Spectral responsibility is approximately human eye response
3) Illuminance to Digital Converter
4) Wide range and High resolution. (1 - 65535 lx)
5) Low Current by power down function
6) 50Hz / 60Hz Light noise reject-function
7) 1.8V Logic input interface
8) No need any external parts
9) Light source dependency is little. (ex. Incandescent Lamp. Fluorescent Lamp. Halogen Lamp. White LED. Sun Light)
10) It is possible to select 2 type of I²C slave-address.
11) Adjustable measurement result for influence of optical window (It is possible to detect min. 0.11 lx, max. 100000 lx by using this function.)
12) Small measurement variation (+/- 20%)
13) The influence of infrared is very small.

●Applications

Mobile phone, LCD TV, NOTE PC, Portable game machine, Digital camera, Digital video camera, Car navigation, PDA, LCD display

Mini task

CPP
- BH1750.cpp
- BH1750.h

Ňŕ śźĂğvî ì ĂśźĂŭửĂŕ ì Ă Í Ĵ Ă

```cpp
#ifndef BH1750_H_
#define BH1750_H_

#include "main.h"

class BH1750
{
public:

    void BH1750_Init(void);

    void BH1750_Start(void);

    uint16_t BH1750_ReadLight(void);

};

#endif /* BH1750_H_ */
```

Êì ểî ì ýĂ ụ

```cpp
#include "BH1750.h"

extern I2C_HandleTypeDef hi2c1;

#define BH1750_ADDRESS (0x23 << 1)
#define BH1750_POWER_ON 0x01
#define BH1750_CONT_H_RES_MODE 0x10

void BH1750::BH1750_Init(void)
{
    uint8_t power = BH1750_POWER_ON;
    HAL_I2C_Master_Transmit(&hi2c1, BH1750_ADDRESS, &power, 1, HAL_MAX_DELAY);
}

void BH1750::BH1750_Start(void)
{
    uint8_t start = BH1750_CONT_H_RES_MODE;

    HAL_I2C_Master_Transmit(&hi2c1, BH1750_ADDRESS, &start, 1, HAL_MAX_DELAY);
}

uint16_t BH1750::BH1750_ReadLight(void)
{
    uint8_t data[2];

    HAL_I2C_Master_Receive(&hi2c1, BH1750_ADDRESS, data, 2, HAL_MAX_DELAY);

    uint16_t lux = (data[0] << 8) | data[1];

    return lux / 1.2;
}
```

Ļvºýğì Ặ ụ

SPI

Serial Peripheral Interface

Basics

Four-wire protocol (MISO, MOSI, SCLK, SS).

Full-duplex communication, master-slave architecture.

How SPI Works

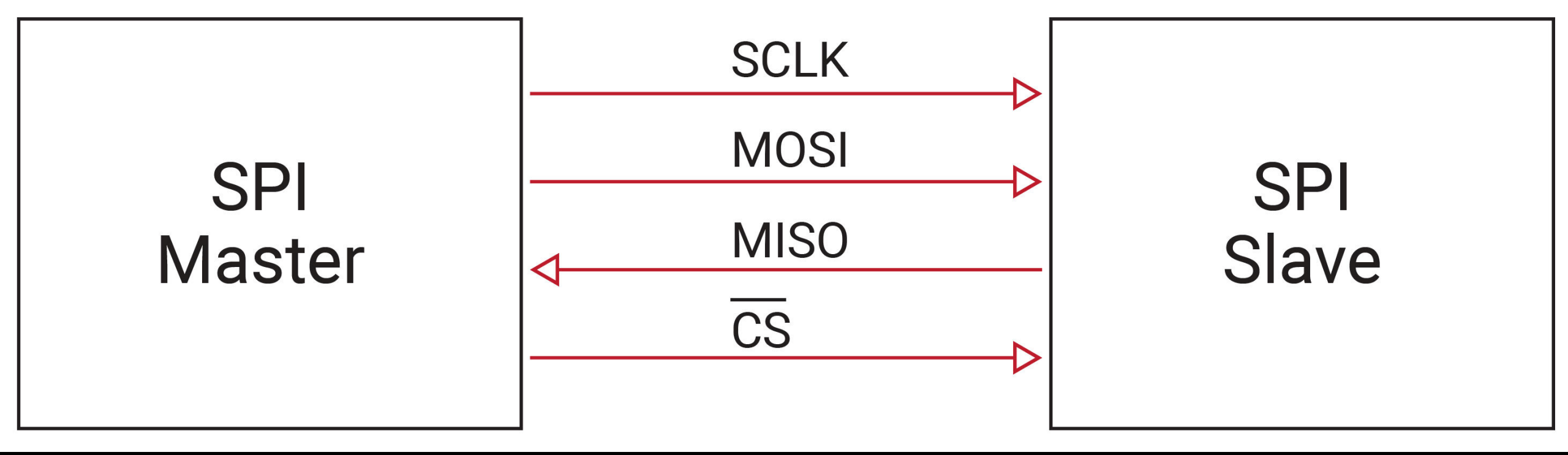
Clock-driven data exchange, roles of master and slave devices.

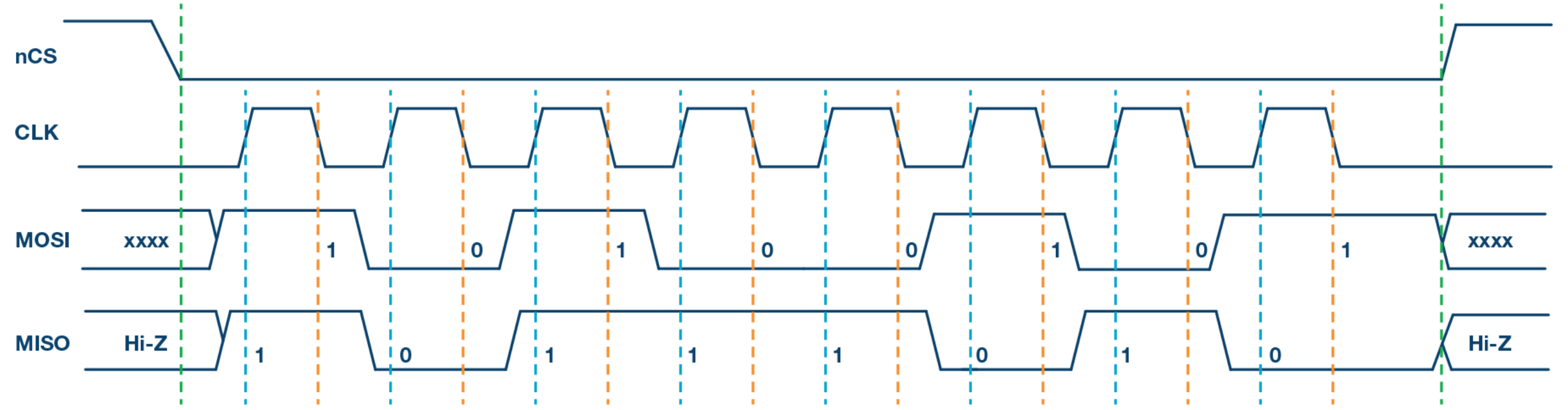
Advantages and Limitations

High speed, simplicity vs. limited multi-device support.

Applications in Robotics

Connecting high-speed peripherals like SD cards, OLED displays.


SPI
Master
SCLK
MOSI
MISO
$\overline{CS}$
SPI
Slave


nCS
CLK
MOSI
xxxx
1
0
1
0
0
1
0
1
xxxx
MISO
Hi-Z
1
0
1
1
1
0
1
0
Hi-Z

CAN

Serial Peripheral Interface

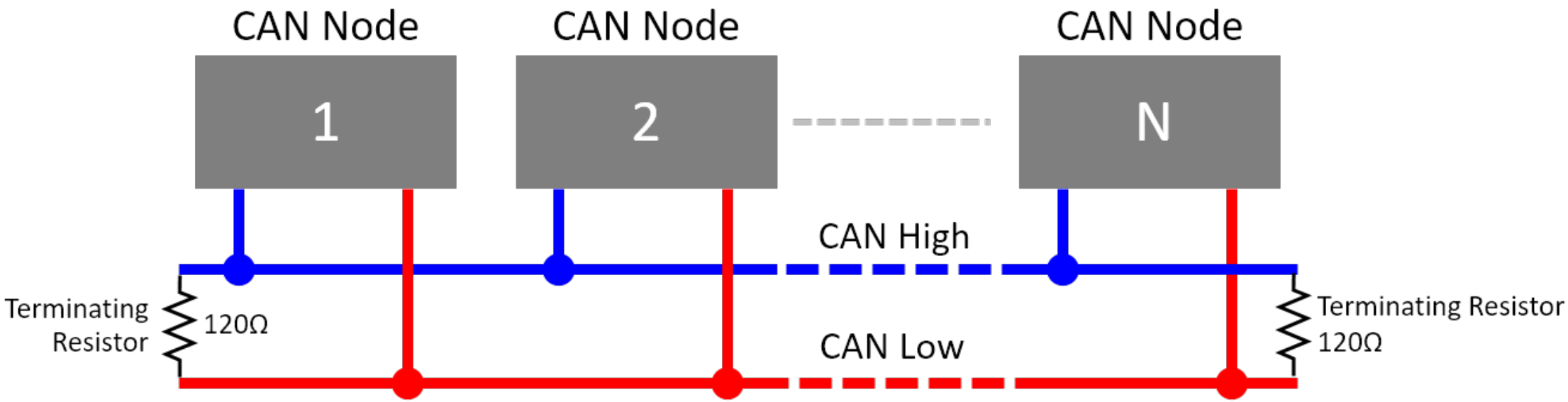
- **Basics**
- Multi-master, message-oriented protocol.
- Designed for high reliability in noisy environments.

Advantages and Use Cases

Robust error-checking, real-time performance, ideal for automotive and industrial robotics.

Applications

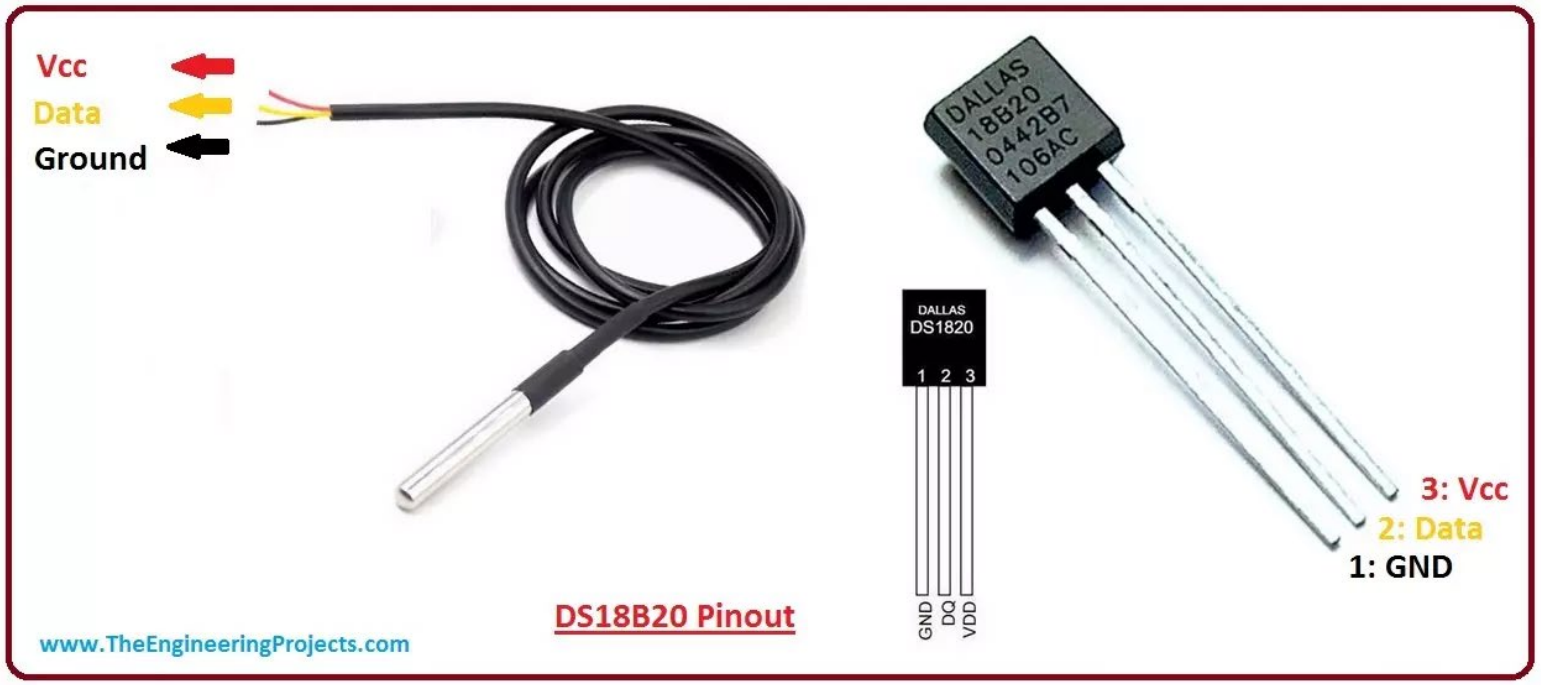
Communication between controllers in mobile robots or AGVs (e.g., ROS nodes).


CAN Node
CAN Node
CAN Node
1
2
N
CAN High
CAN Low
Terminating
Resistor
120Ω
Terminating Resistor
120Ω

Special protocol

One-wire: take a look at the example and read the datasheet together.

How to manage with every real sensor



Requirement:

1. Datasheet of your STM32 Board
2. Your STM32 Board's Pinout
3. Datasheet of the DS18B20 temperature sensor

Example in one-wire temperature sensor of DS18B20

*If you are working on basic programming. No need to use datasheet of STM32 board

DS18B20

Click here for production status of specific part numbers.

DS18B20 **Programmable Resolution 1-Wire Digital Thermometer**

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

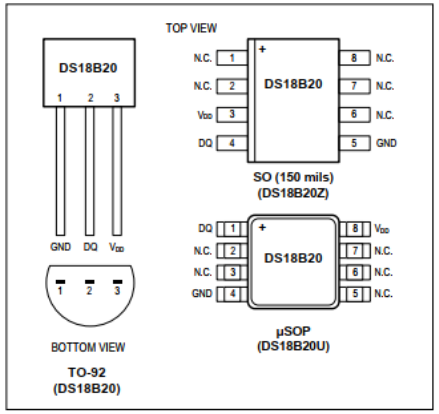
Ordering Information ***appears at end of data sheet.***

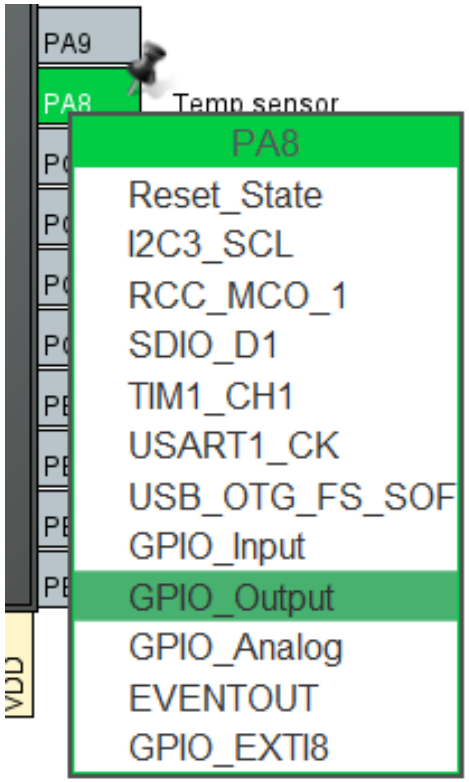
1-Wire is a registered trademark of Maxim Integrated Products, Inc.

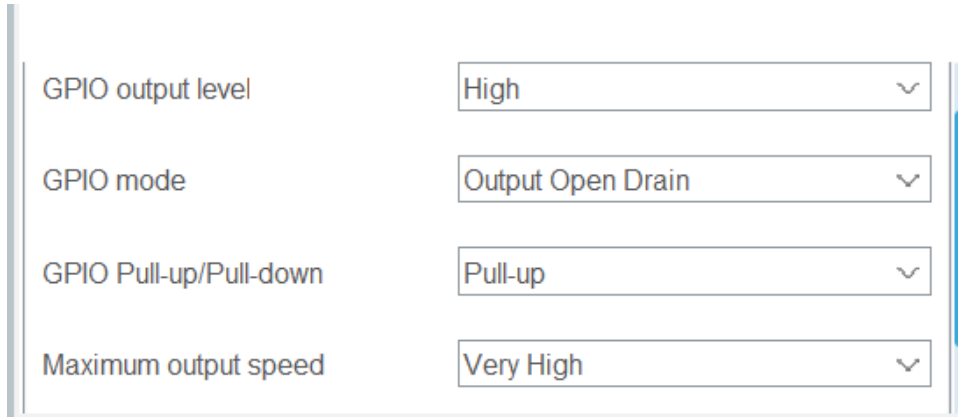
Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations







MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

```
7
8  #ifndef DS18B20_H_
9  #define DS18B20_H_
10
11 #include "main.h"
12
13 #ifdef __cplusplus
14 extern "C" {
15 #endif
16
17 void Delay_us(uint32_t us);
18 void startInitial();
19 void sendByte(uint8_t data);
20 uint8_t ReadBit();
21 uint8_t ReadByte();
22 void Read_ScratchPad(uint8_t* scratchPadPointer);
23 float ConvertToCelsius(uint8_t* tempPointer);
24 float Read_temp();
25
26 extern uint32_t cnt;
27 extern uint8_t scratchPad[9];
28
29 #ifdef __cplusplus
30 }
31 #endif
32
33
34 #endif /* DS18B20_H_ */
35
```

Header file

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

```
7
8  #include "DS18B20.h"
9
10 uint32_t cnt = 0;
11 uint8_t scratchPad[9];
12
13 void Delay_us(uint32_t us)
14 {
15     uint32_t startTime = TIM2->CNT;
16     uint32_t countTime = us*16;
17     while(TIM2->CNT - startTime < countTime);
18 }
19
20 void startInitial()
21 {
22     HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_RESET);
23     Delay_us(500);
24
25     HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_SET);
26
27     while(HAL_GPIO_ReadPin(Temp_sensor_GPIO_Port, Temp_sensor_Pin) == GPIO_PIN_RESET);
28     while(HAL_GPIO_ReadPin(Temp_sensor_GPIO_Port, Temp_sensor_Pin) == GPIO_PIN_SET);
29     uint32_t startTime = TIM2->CNT;
30
31     while(HAL_GPIO_ReadPin(Temp_sensor_GPIO_Port, Temp_sensor_Pin) == GPIO_PIN_RESET);
32     cnt = (TIM2->CNT - startTime) / 16;
33 }
```

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

```
void sendByte(uint8_t data)
{
    // send byte in hex form with LSB
    for(int i=0 ; i<8 ; i++)
    {
        if((data >> i) & 1)
        {
            HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_RESET);
            Delay_us(5);

            HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_SET);
            Delay_us(75);
        }
        else
        {
            HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_RESET);
            Delay_us(65);

            HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_SET);
            Delay_us(15);
        }
    }
}
```

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

```
58
59  uint8_t ReadBit()
60  {
61      uint8_t bit_value;
62
63      HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_RESET);
64      Delay_us(5);
65
66      HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_SET);
67      Delay_us(10);
68
69      bit_value = HAL_GPIO_ReadPin(Temp_sensor_GPIO_Port, Temp_sensor_Pin) == GPIO_PIN_SET;
70      Delay_us(50);
71
72      return bit_value;
73  }
74
75  uint8_t ReadByte()
76  {
77      uint8_t byte_value = 0;
78
79      for (int i=0 ; i<8 ; i++)
80      {
81          byte_value |= (ReadBit() << i);
82      }
83      return byte_value;
84  }
85
```

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

```
85
86  void Read_ScratchPad(uint8_t* scratchPadPointer)
87  {
88      for(int i=0 ; i<9 ; i++)
89      {
90          scratchPadPointer[i] = ReadByte();
91      }
92  }
93
94  // Not endianness safe
95  float ConvertToCelsius(uint8_t* tempPointer)
96  {
97      return *((int16_t*)tempPointer) * 0.0625f;
98  }
99
100 float Read_temp()
101 {
102     Read_ScratchPad(scratchPad);
103     return ConvertToCelsius(scratchPad);
104 }
```

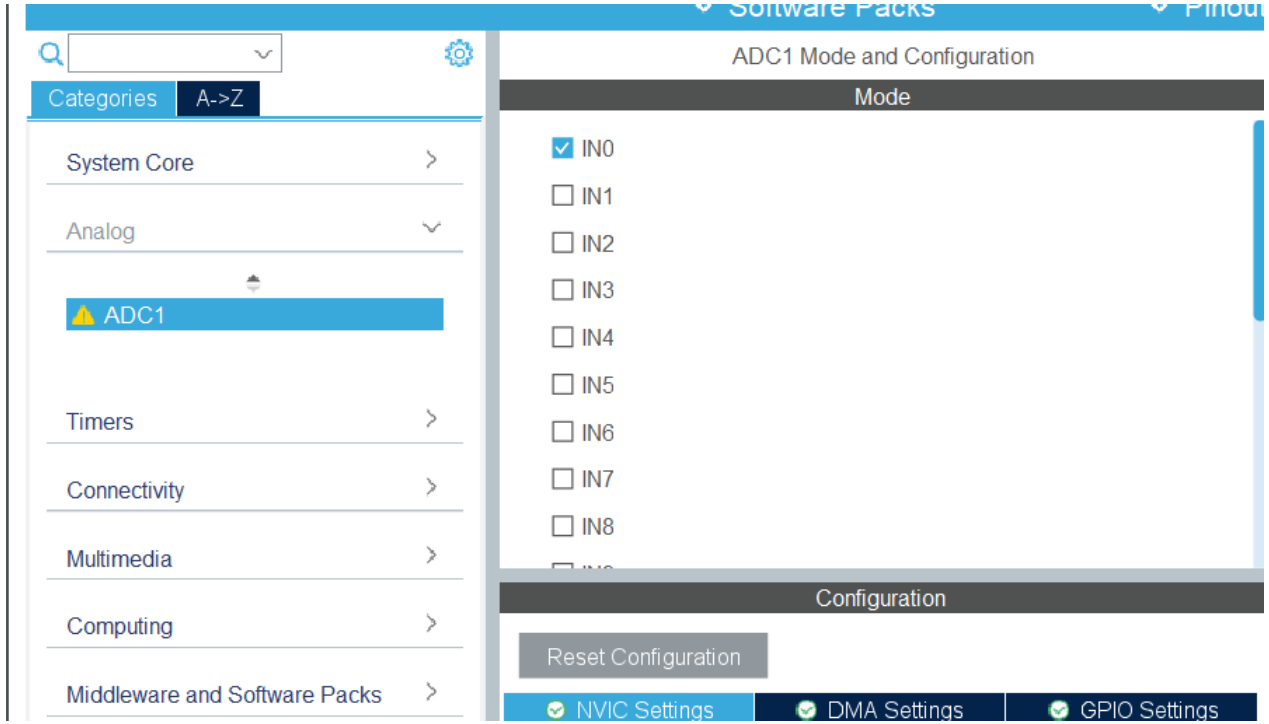
MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

```
void setup()
{
    HAL_TIM_Base_Start(&htim2);
```

```
void loop()
{
    // Temperature
    startInitial();
    sendByte(0xCC);
    sendByte(0x44);
    HAL_Delay(1000);

    startInitial();
    sendByte(0xCC);
    sendByte(0xBE);

    temp = Read_temp();
```

ADC Read

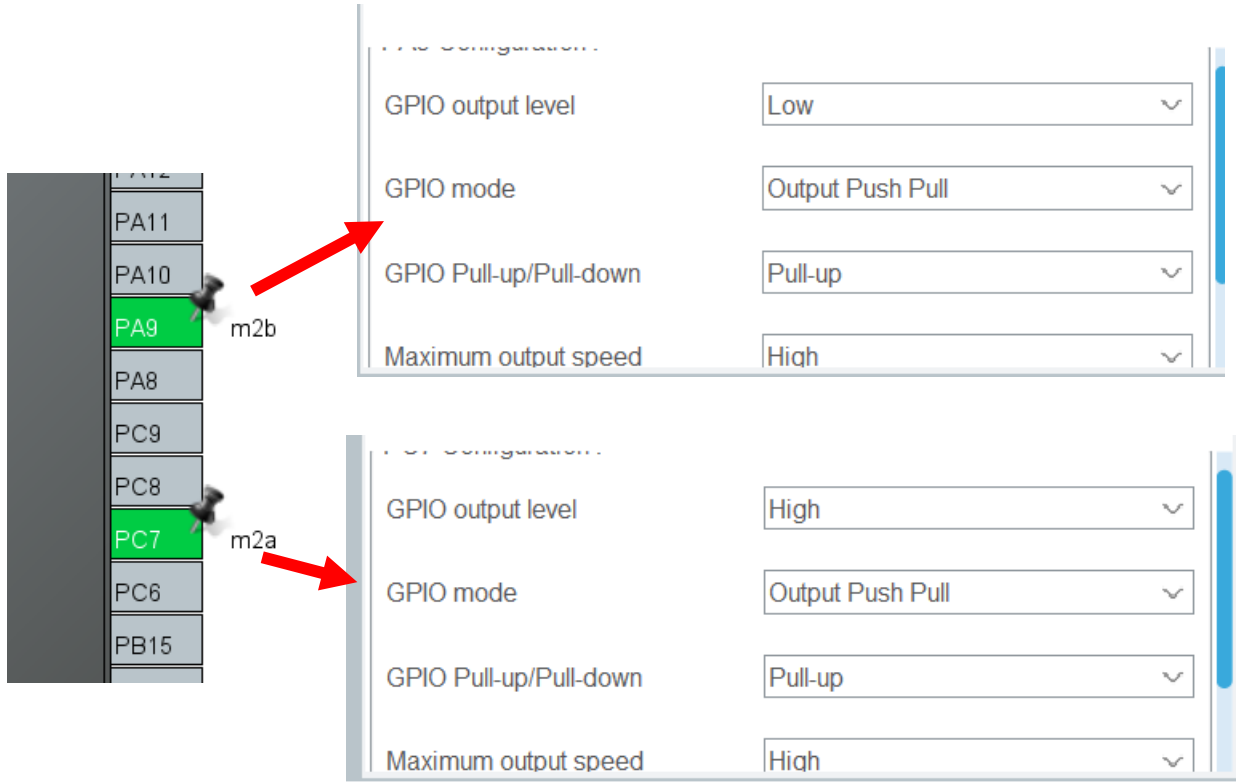
In this workshop, we provide a complete sensors, so you don’t need to use the protocol from the previous session.




```
#include "CPPmain.h"
#include "stdio.h"

extern ADC_HandleTypeDef hadc1;
uint32_t adc_value;

void setup()
{

}

void loop()
{
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, 1000) == HAL_OK)
    {
        adc_value = HAL_ADC_GetValue(&hadc1);
    }

}
```

Motor Drive

You need to use the Maker Drive board to control the motor.




```
#include "CPPmain.h"
#include "stdio.h"

extern ADC_HandleTypeDef hadc1;
uint32_t adc_value;

void setup()
{

}

void loop()
{
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, 1000) == HAL_OK)
    {
        adc_value = HAL_ADC_GetValue(&hadc1);
    }

    // Left Motor
    HAL_GPIO_WritePin(m2a_GPIO_Port, m2a_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(m2b_GPIO_Port, m2b_Pin, GPIO_PIN_RESET);
}
```

Conclusion

3 steps for STM32

DESIGN WHAT YOU WANT

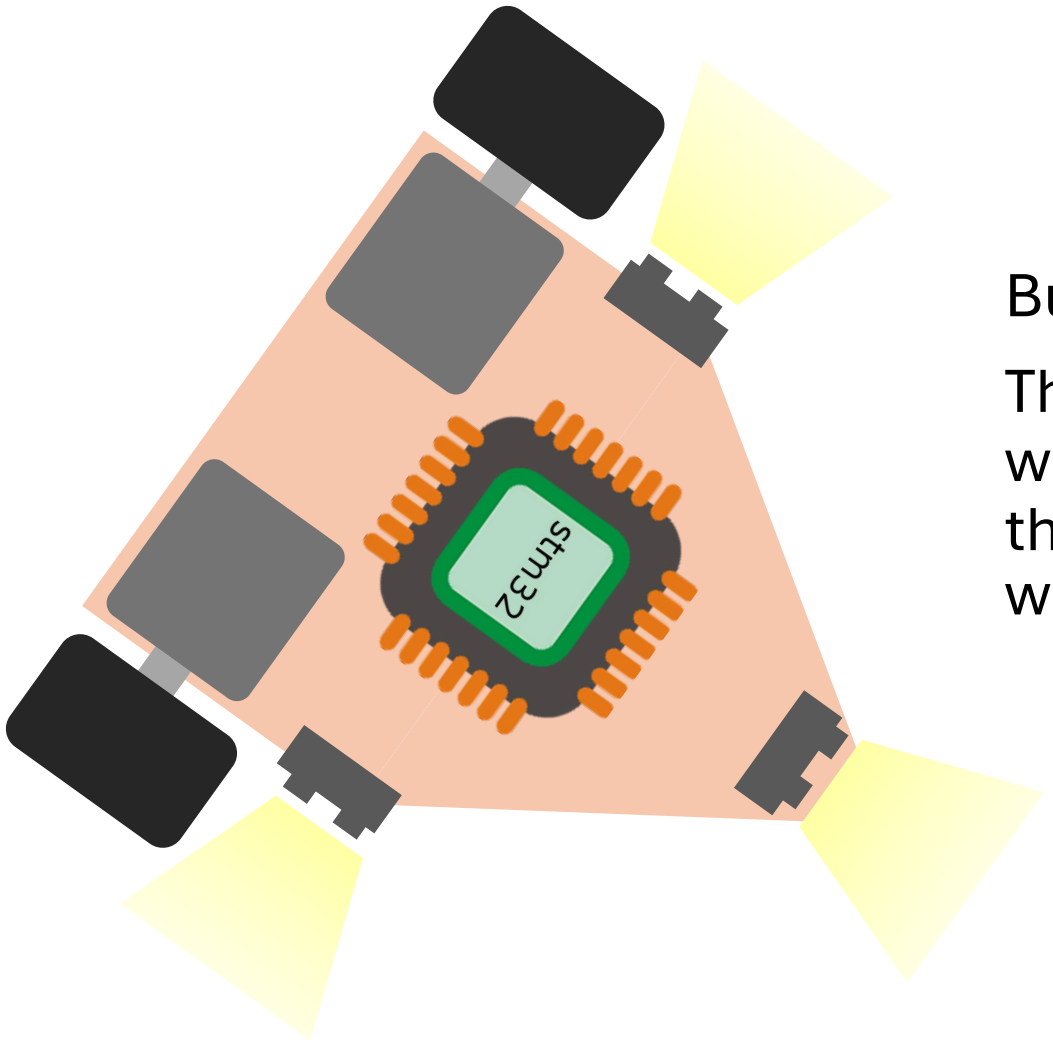
SETTING UP IOC

CODING

Mini Robotics Project

Nothing is more important than the experience

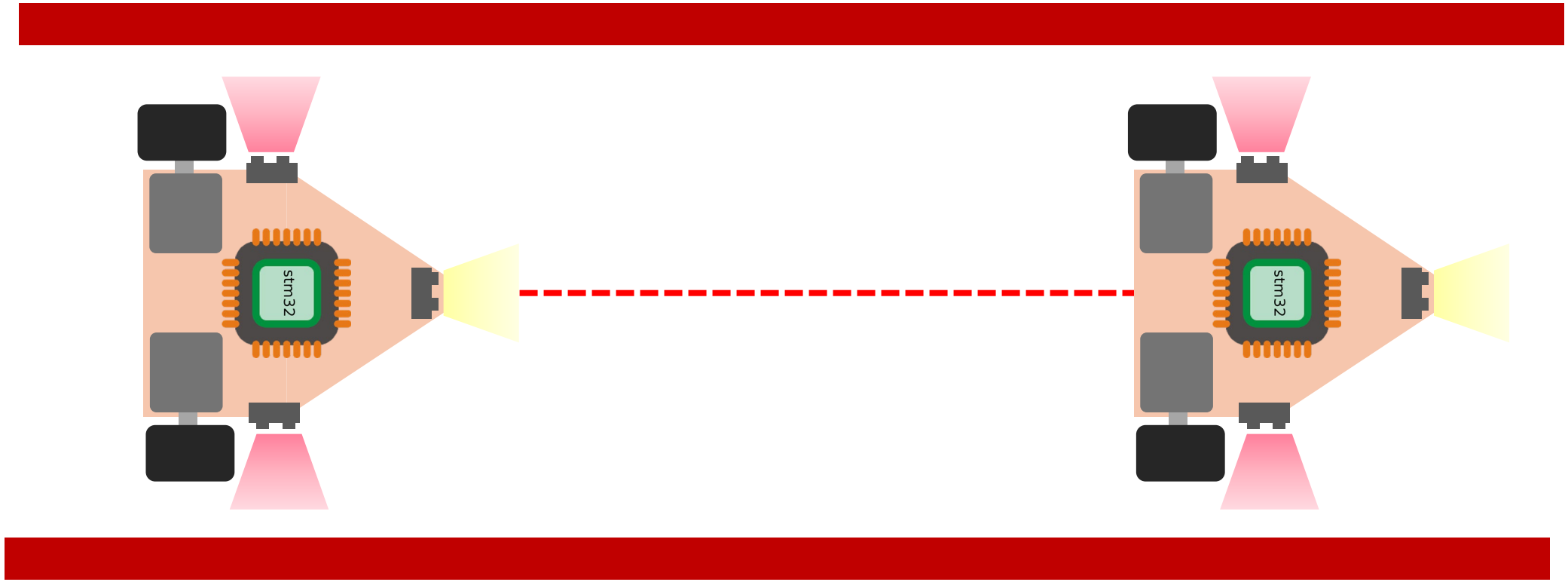
Your task



Building a robot equipped with three IR sensors.

The robot operates automatically: it will turn right when there is no wall on the right, turn left when there is no wall on the left, and move forward when there are two parallel walls.

Your task



Your task