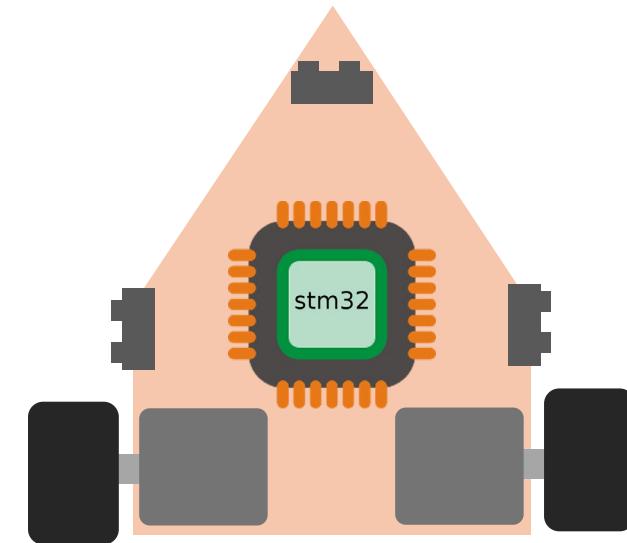
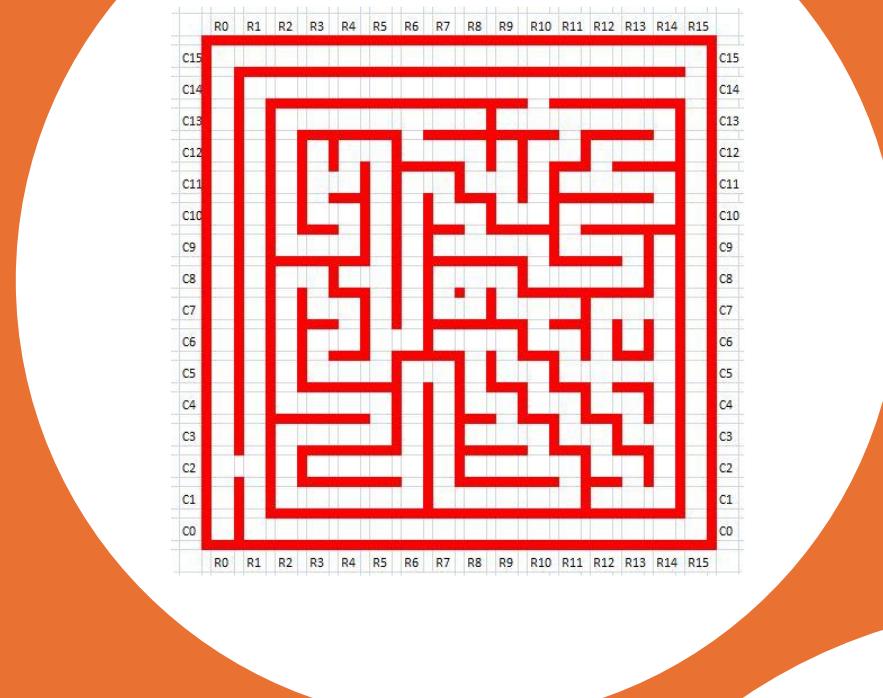
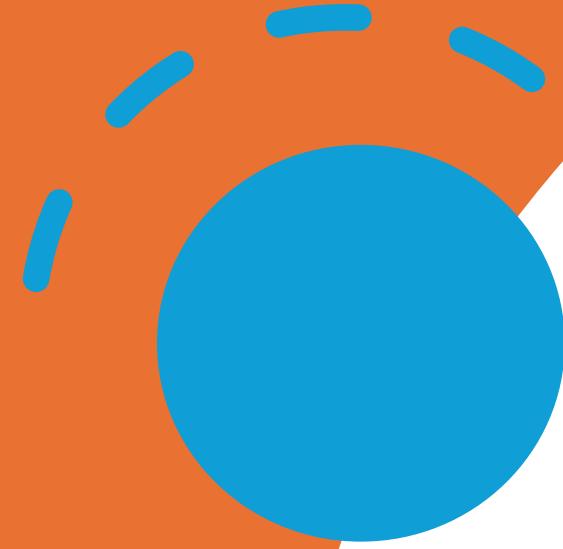


STM32 for Maze-solving Robotics

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

Comprehensive offer of production-ready embedded software and tools

1

Compatibility across the STM32 hardware products

2

Free and business-friendly license

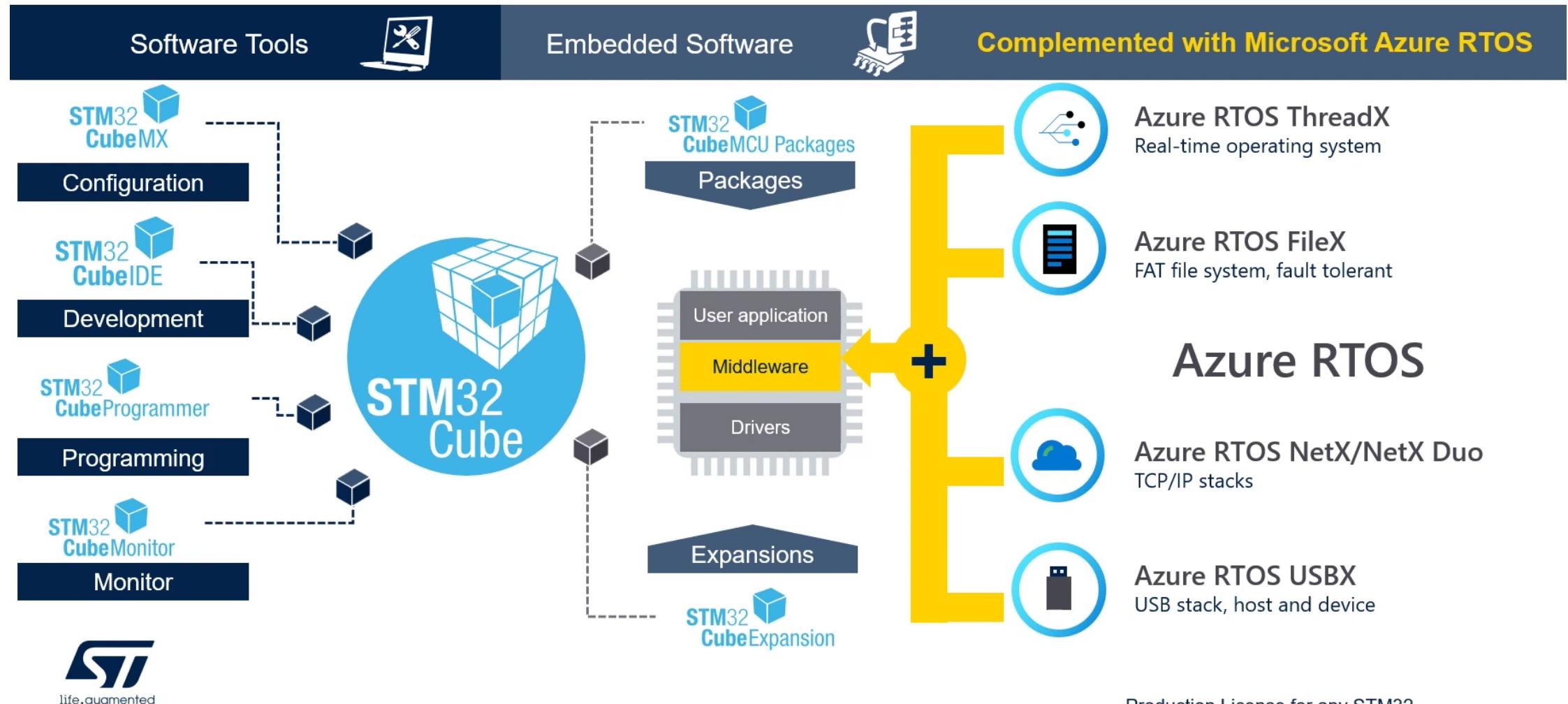
3

Ecosystem of STM32

STM32Cube MCU ecosystem

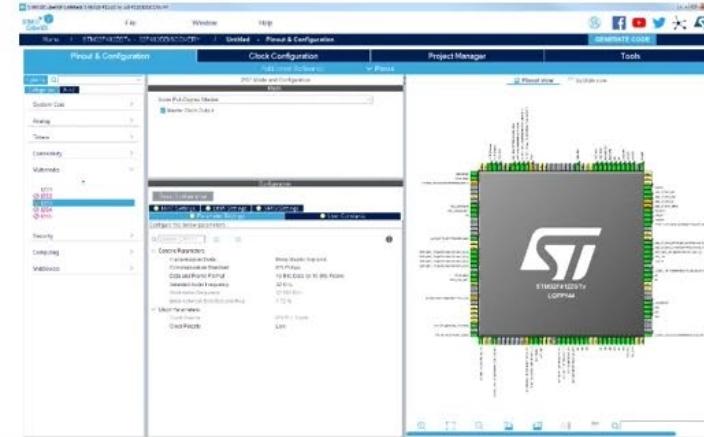


Ecosystem of STM32



Inside the STM32Cube MCU ecosystem (Software Tools)

Ecosystem of STM32



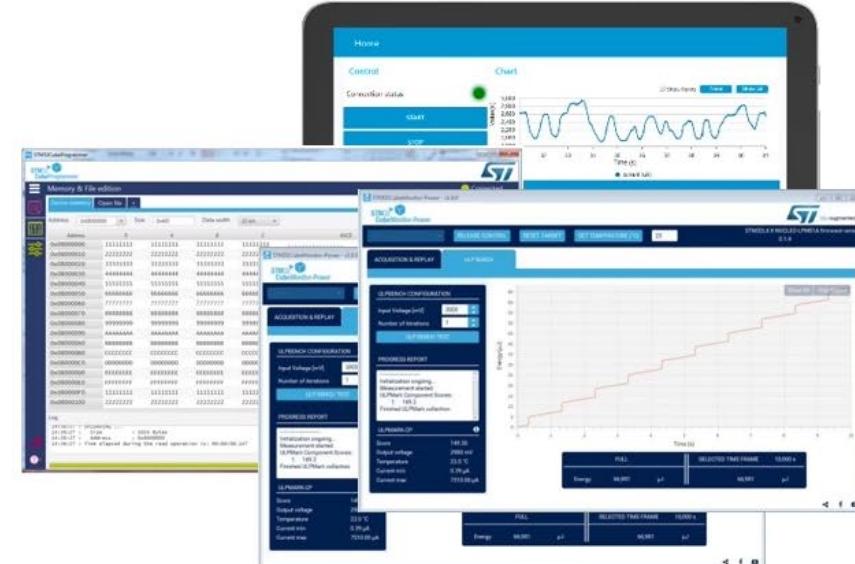
STM32 CubeMX

STM32CubeMX, GUI builders
Configure & generate code



STM32 CubelIDE

ST and Partner IDEs
Compile and debug



STM32 CubeProgrammer

STM32CubeProg/Monitor
Monitor, program & utilities

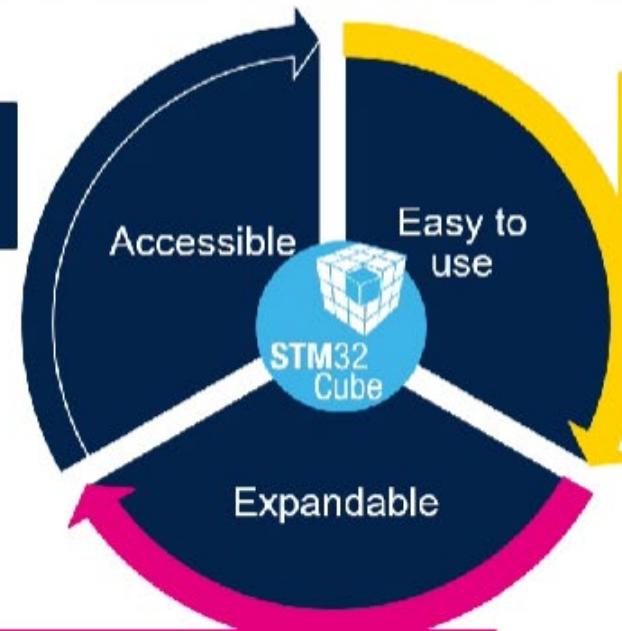
STM32 CubeMonitor

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

Ecosystem of STM32

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

Fits many developer profiles:
from beginner to expert



Easy & fast learning curve to focus
on your competitive advantage

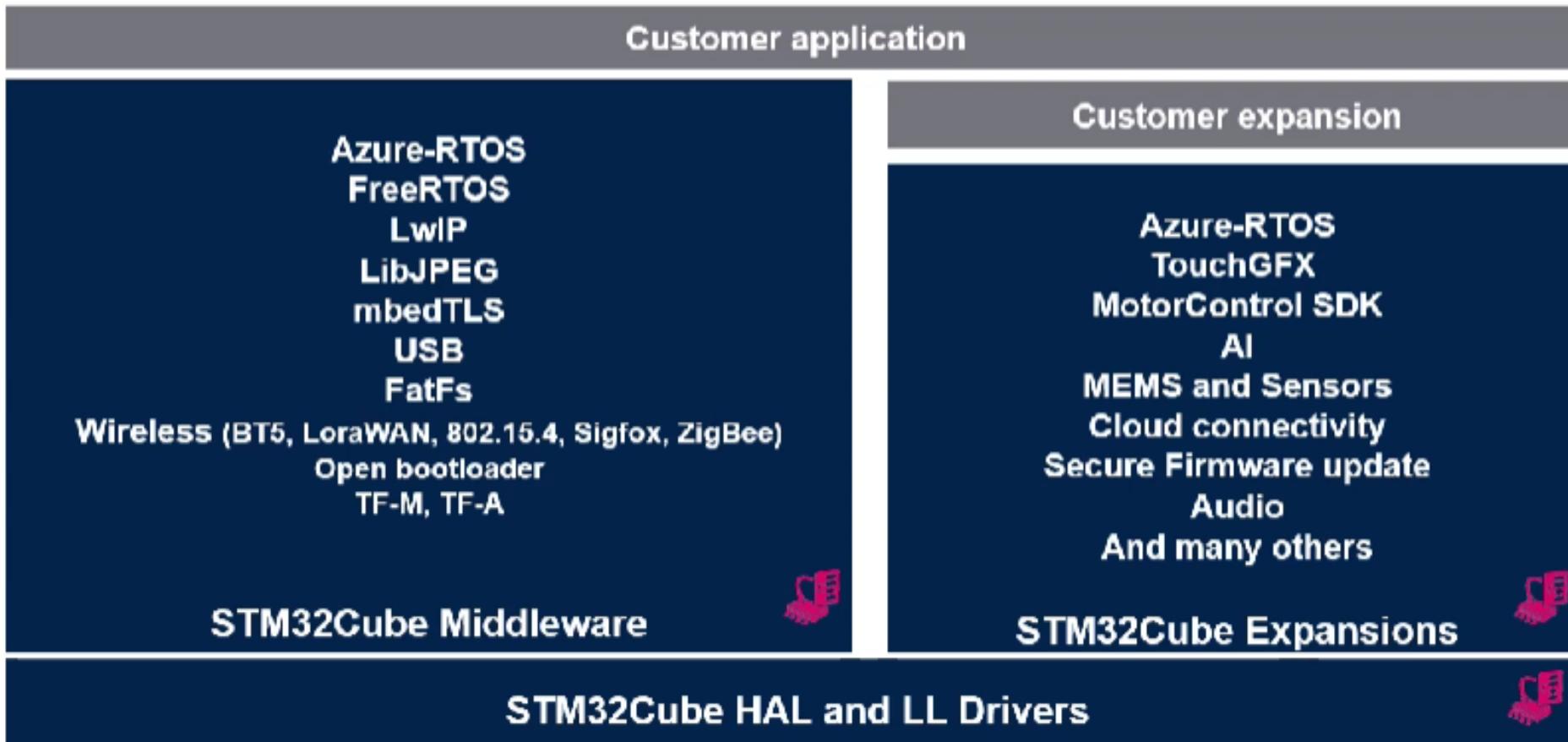
Production-ready to address
many customer use cases



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

Flexible prototyping

www.st.com/stm32nucleo



Discovery kits

Key feature prototyping

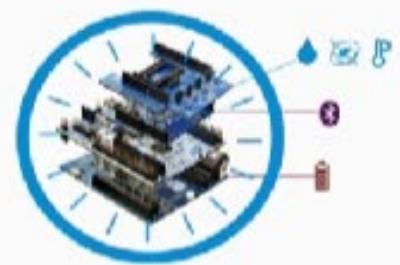
www.st.com/stm32discovery



Evaluation boards

Full feature evaluation

www.st.com/stm32evaltools



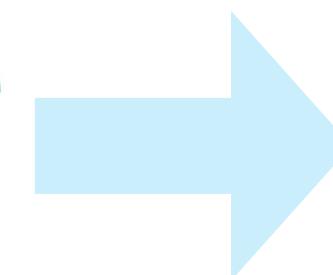
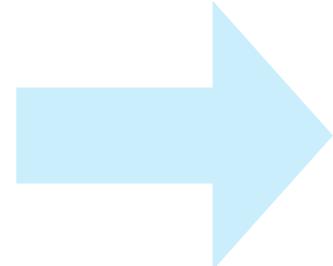
STM32 Nucleo expansion

Functionality add-on

www.st.com/x-nucleo

Ecosystem of STM32

Easy PCB design in STM32 family



Design the specific
PCB for system

Order the PCB
production

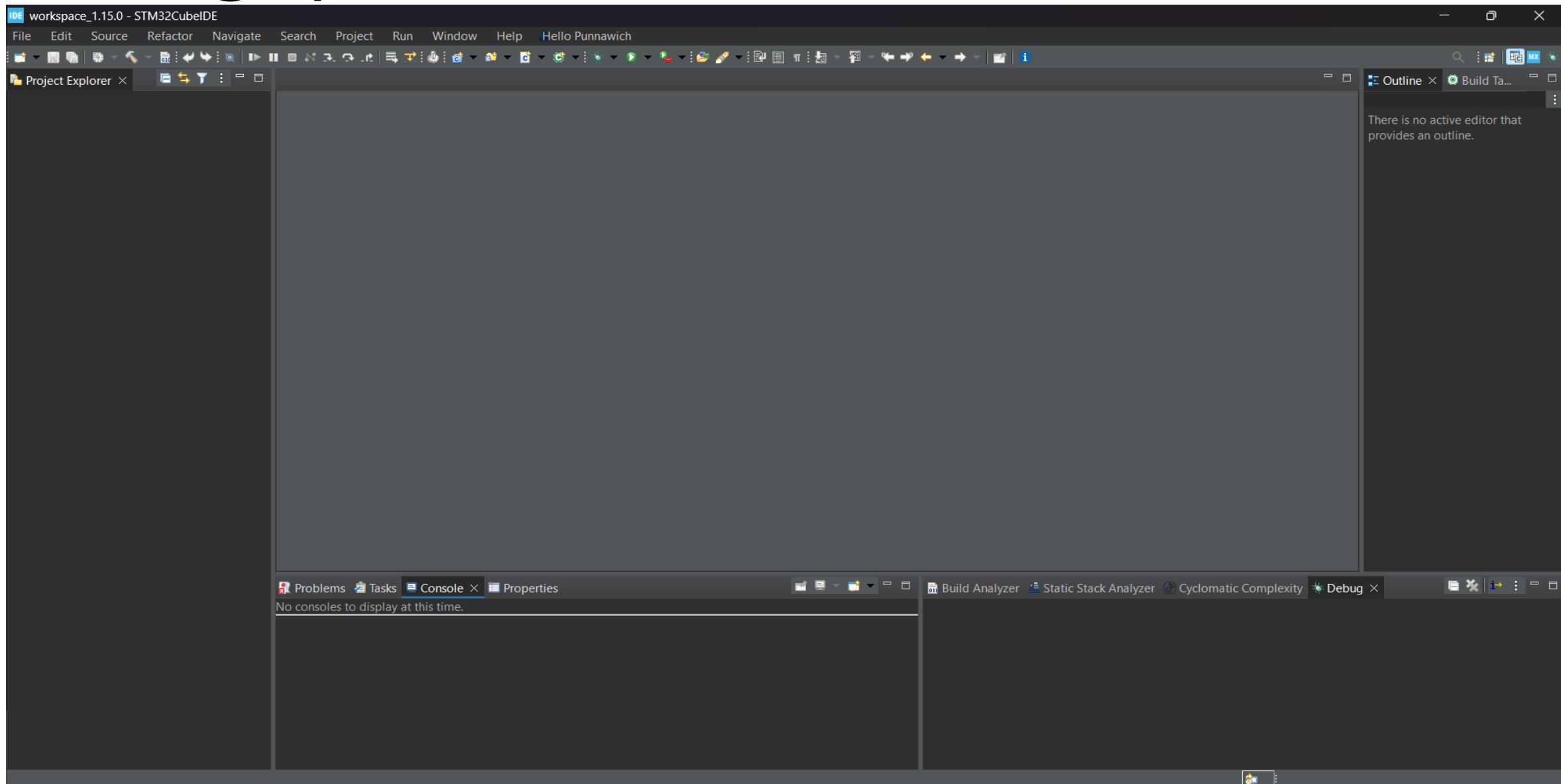
Programming the
PCB with STM32 IDE



Setting up the environment for STM32 IDE

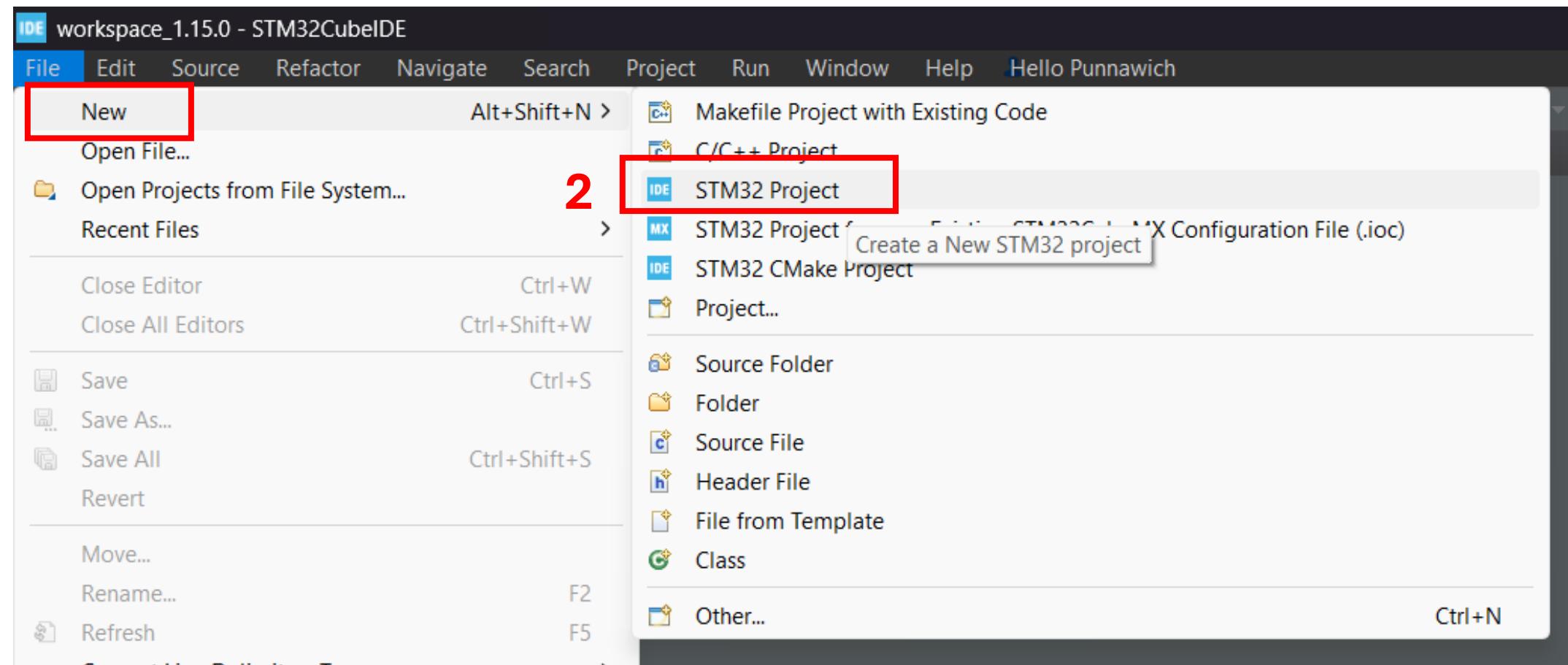
How to start your STM32 project and first project in
STM32

Setting up the environment



Setting up the environment

Creating the STM32 Project



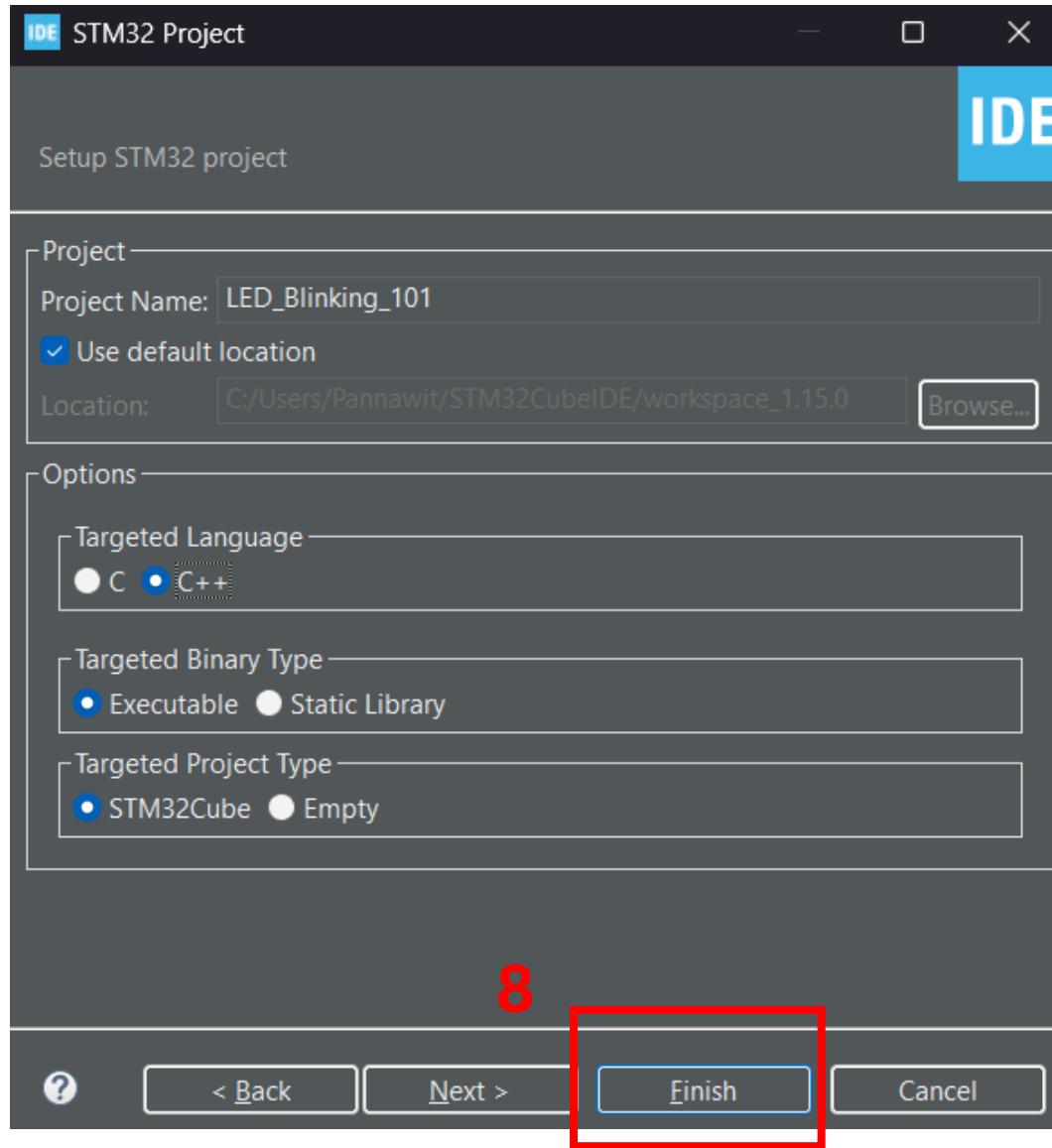
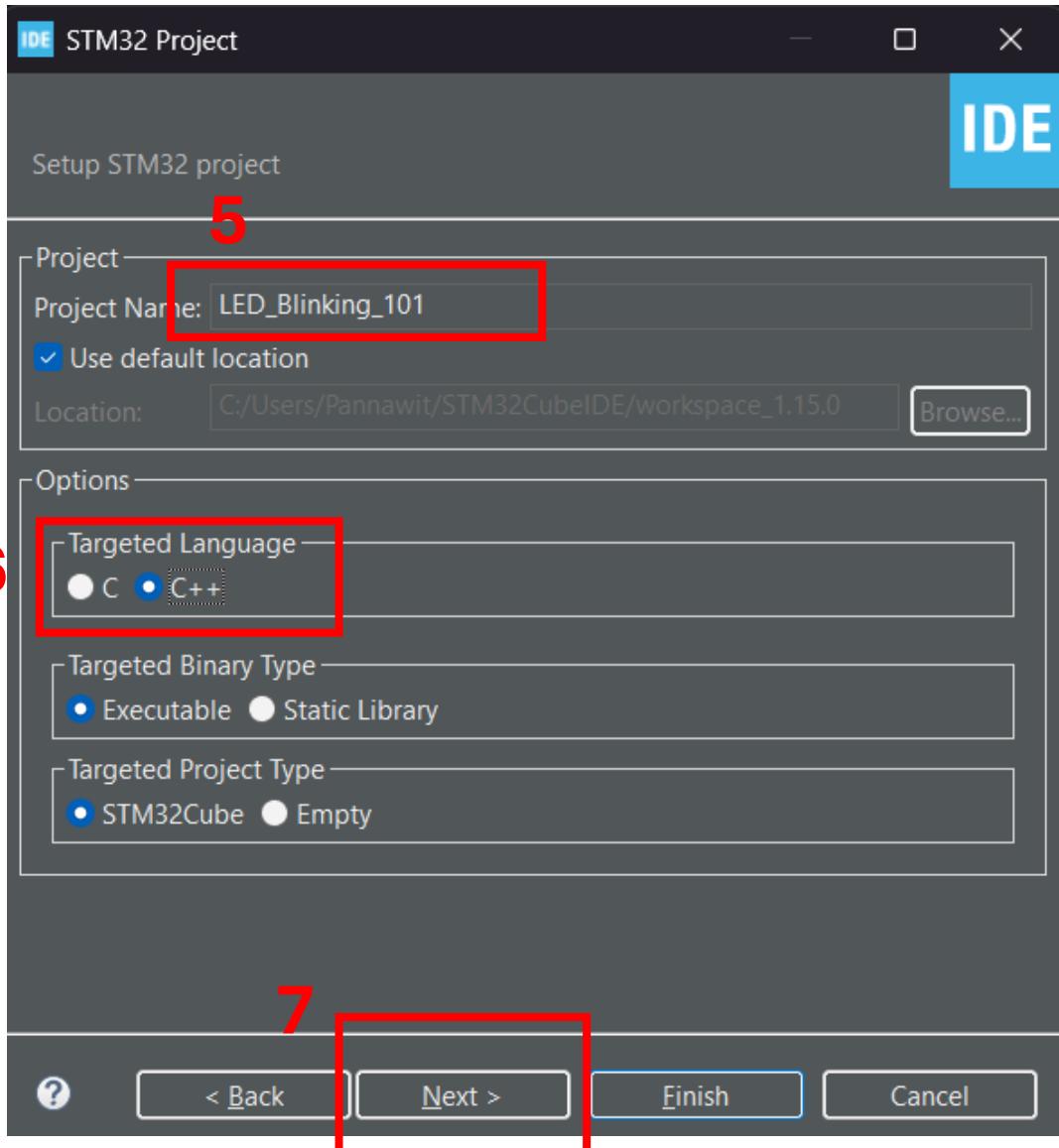
Setting up the environment

The screenshot shows the 'STM32 Project' window with the 'Target Selection' tab active. The 'MCU/MPU Selector' tab is selected. In the 'MCU/MPU Filters' section, the 'Commercial Part Number' field is highlighted with a red box and contains the value 'STM32G431CBU6'. A large red annotation '3' is placed over this field. Below it, a red annotation 'Typing the correct board number' covers the 'PRODUCT INFO' section. In the 'MCUs/MPUs List' section, a table shows two items, both of which are highlighted with a red box. A red annotation '4' is placed over the first item in the list. The table data is as follows:

*	Comme...	Part ...	Referen...	Marke...	X	Unit P...	X	Board	X	Packa...	X	Flash	X	RAM	X	I/O	X	Frequ...	X
★	STM32...	STM32...		Active		2.2781				UFQFP...	128 kB...	32 kBt...	42		170	MHz			
★	STM32...	STM32...		Active		2.2781				UFQFP...	128 kB...	32 kBt...	42		170	MHz			

At the bottom of the dialog, there are buttons for '?', '< Back' (disabled), 'Next >', 'Finish' (disabled), and 'Cancel'.

Setting up the environment



Setting up the environment

LED_Blinking_101.ioc ×

LED_Blinking_101.ioc - Pinout & Configuration

Pinout & Configuration Clock Configuration Project Manager Tools

Software Packs Pinout

Pinout view System view

Categories A-Z

System Core >

Analog >

Timers >

Connectivity >

Multimedia >

Security >

Computing >

Middleware and Software Pac... >

Utilities >

VBAT VDD PA13
PC13 PB9 PA12
PC14... PB8... PA11
PC15... PB7 PA10
PF0... PB6 PA9
PF1... PB5 PA8
PG10... PB4 PC6
PA0 PB3 PB15
PA1 PC11 PB14
PA2 PC10 PB13
PA3 PA15 PB12
PA4 PA14
PA5 PA6 PB11
PA7 PC4 PB10
PC8 PB0 VREF+
PB1 PB1 VDDA
PB2 PB2 VDD
VDDA PB10 VDD
VDD PB11

STM32G431CBUx
UFQFPN48

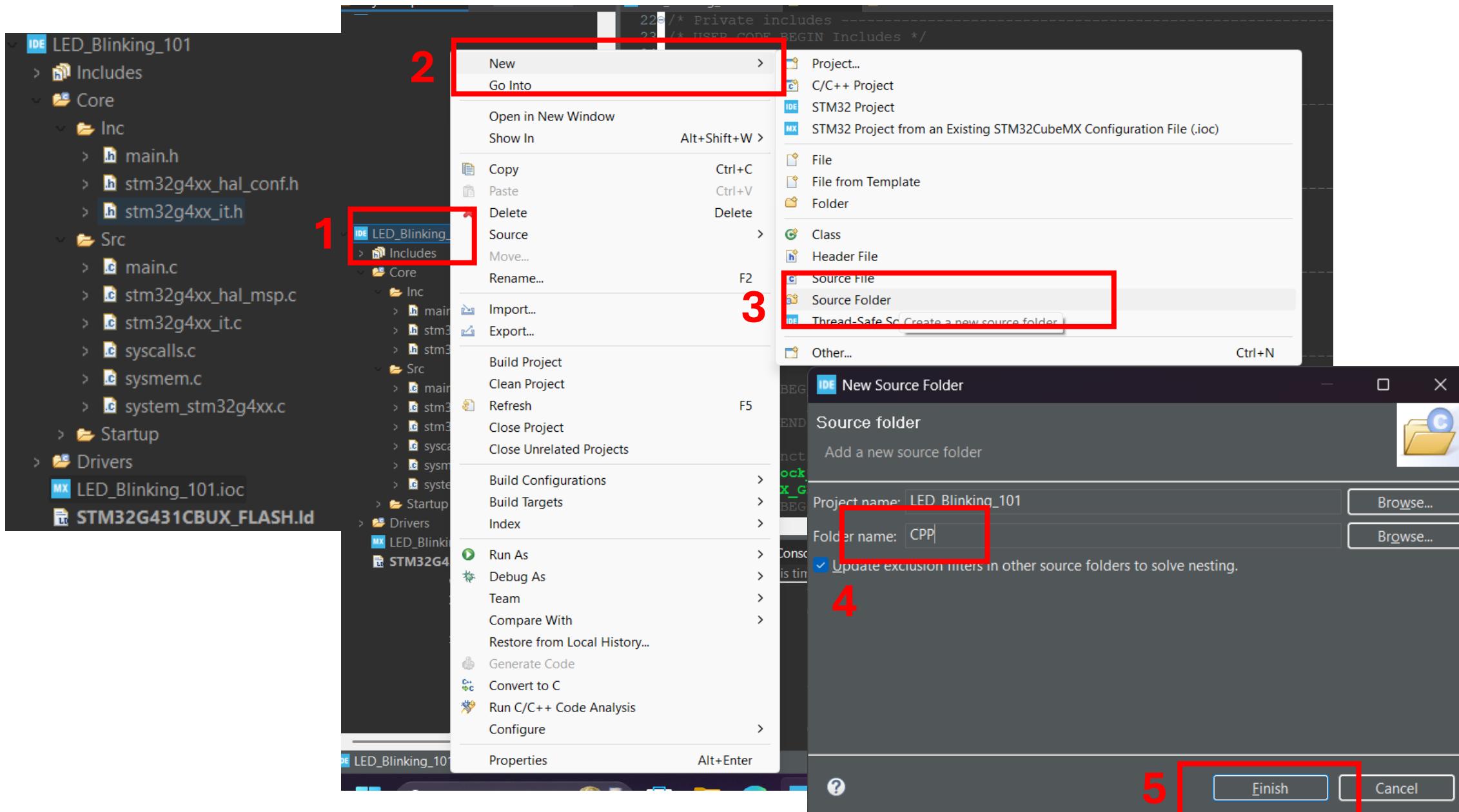
PA5 PA6 PA7 PC4 PB0 PB1 PB2 VREF+ VDDA VDD PB11

Console ×

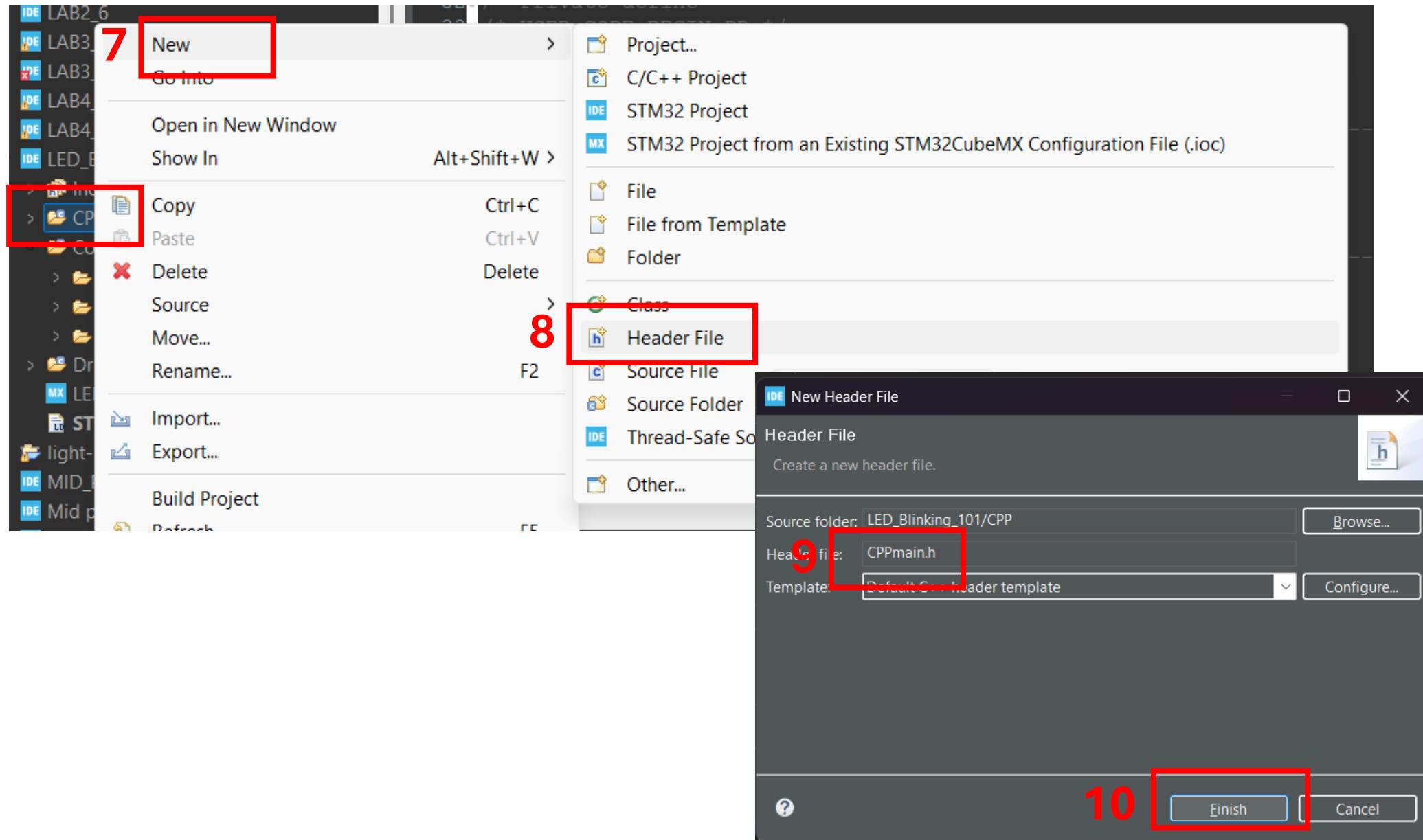
No consoles to display at this time.

17

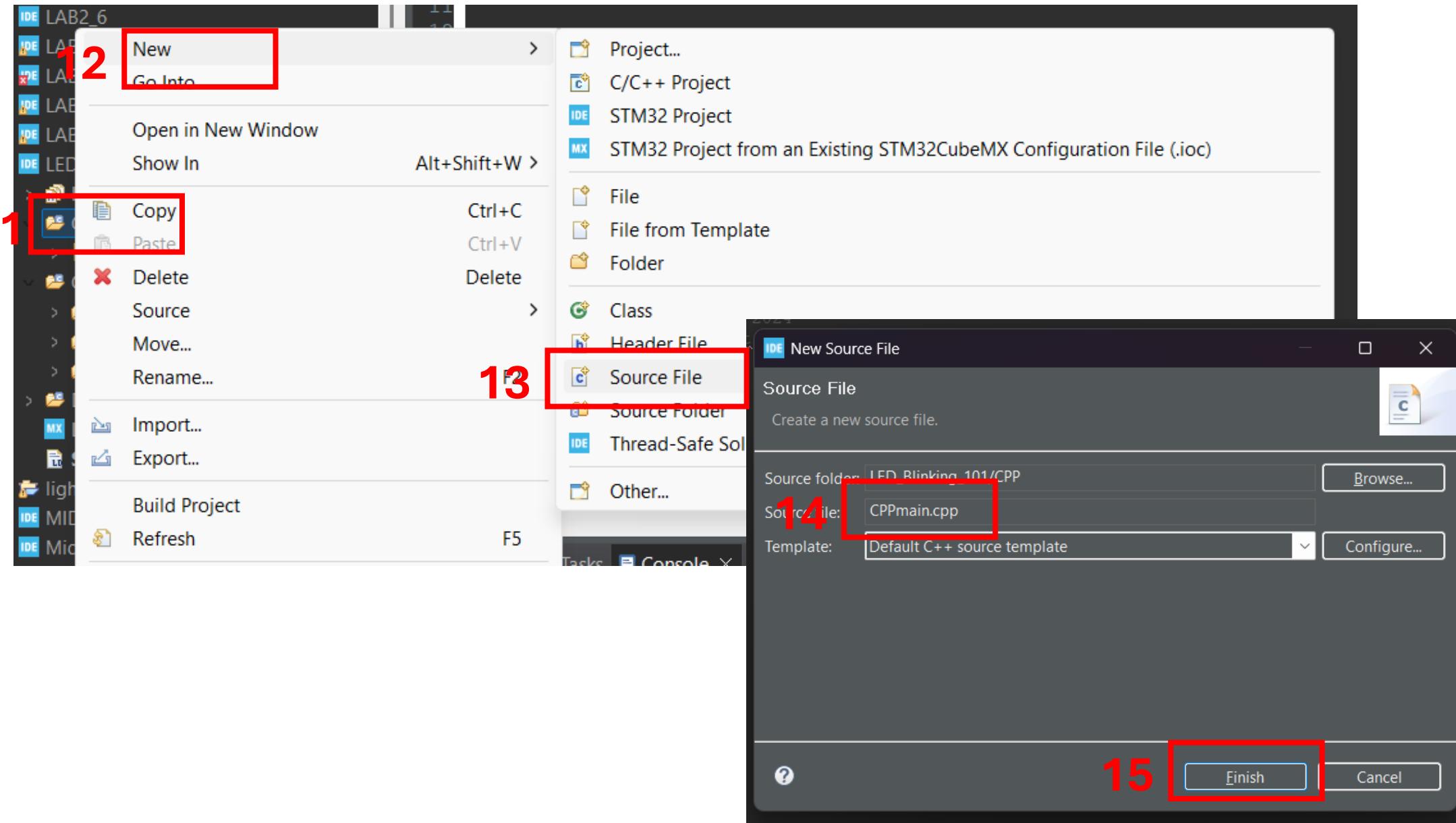
Setting up the environment



Setting up the environment



Setting up the environment



Setting up the environment

The image shows two side-by-side code editors, both titled "LED_Blinking_101.ioc". The left editor displays a header file (CPPmain.h) with the following content:

```
18 /*  
19 * CPPmain.h  
20 *  
21 * Created on: Dec 10, 2024  
22 * Author: Pannawit  
23 */  
24  
25 #ifndef CPPMAIN_H_  
26 #define CPPMAIN_H_  
27  
28  
29 #endif /* CPPMAIN_H_ */
```

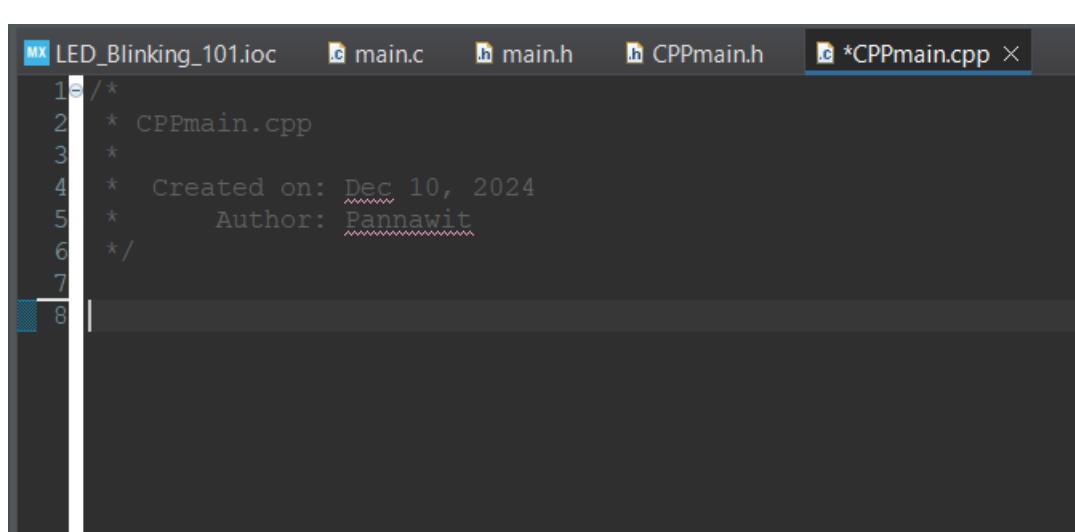
The right editor also displays a header file (CPPmain.h) with the following content, which includes a red box highlighting the main function implementation:

```
18 /*  
19 * CPPmain.h  
20 *  
21 * Created on: Dec 10, 2024  
22 * Author: Pannawit  
23 */  
24  
25 #ifndef CPPMAIN_H_  
26 #define CPPMAIN_H_  
27  
28  
29 #include "main.h"  
30 #ifdef __cplusplus  
31 extern "C" {  
32 #endif  
33  
34 void setup();  
35 void loop();  
36 void gpio_init();  
37  
38 #ifdef __cplusplus  
39 }  
40 #endif  
41  
42  
43 #endif /* CPPMAIN_H_ */
```

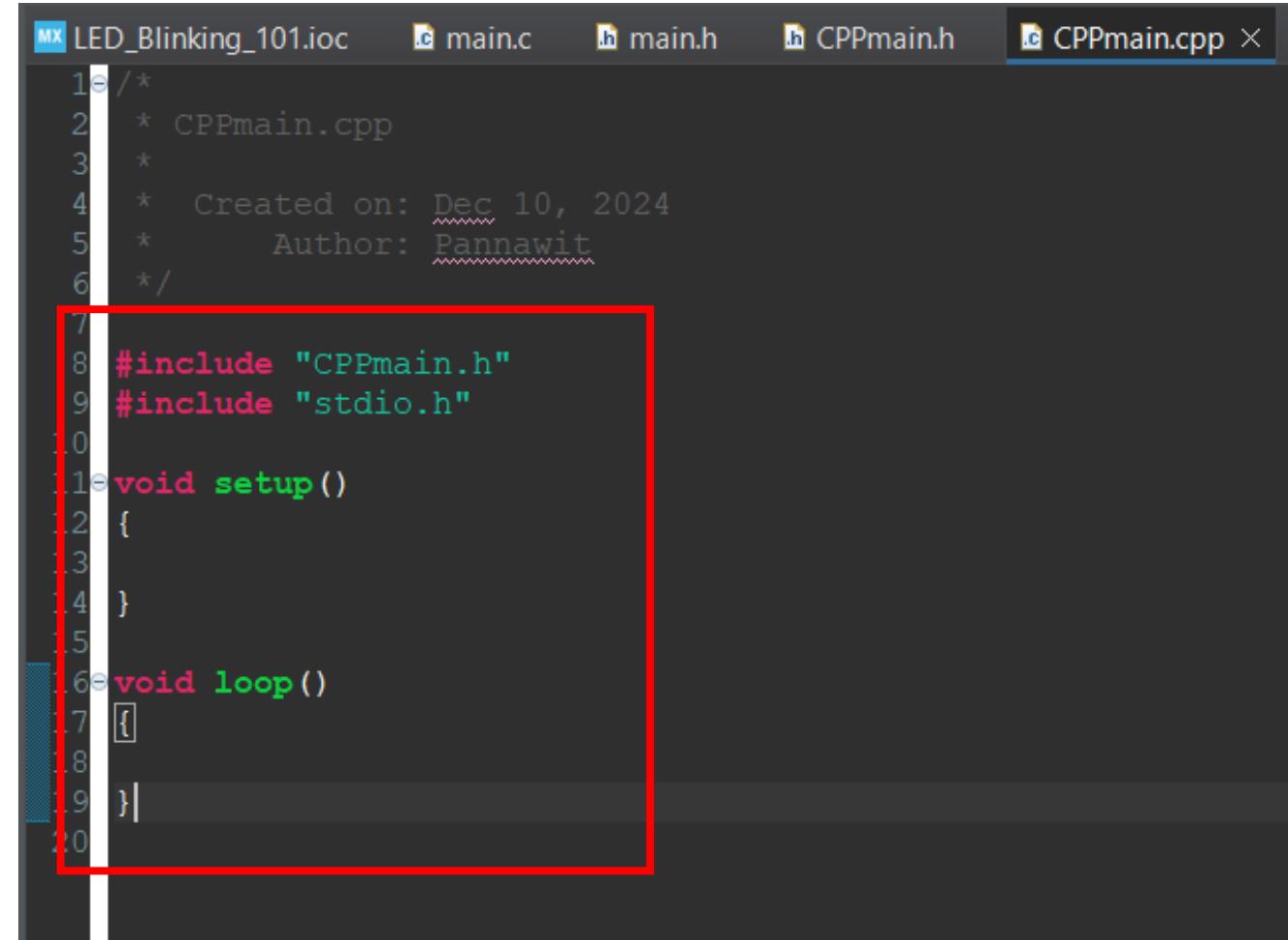
A red rectangular box highlights the following code block in the right editor:

```
30 #ifdef __cplusplus  
31 extern "C" {  
32 #endif  
33  
34 void setup();  
35 void loop();  
36 void gpio_init();  
37  
38 #ifdef __cplusplus  
39 }  
40 #endif  
41  
42  
43 #endif /* CPPMAIN_H_ */
```

Setting up the environment

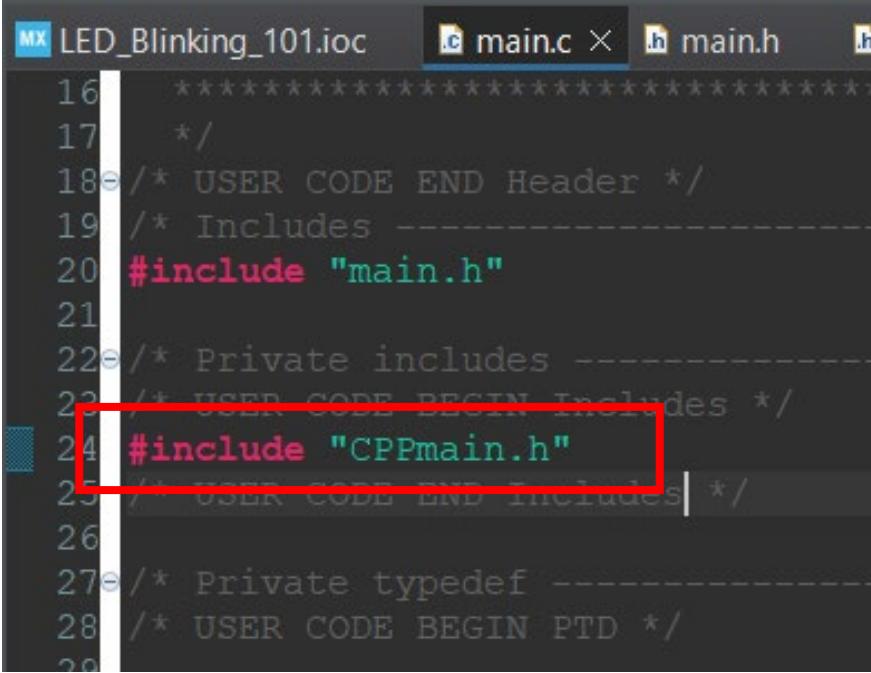


```
MX LED_Blinking_101.ioc    main.c    main.h    CPPmain.h    *CPPmain.cpp X
1  /*
2   * CPPmain.cpp
3   *
4   * Created on: Dec 10, 2024
5   * Author: Pannawit
6   */
7
8
```



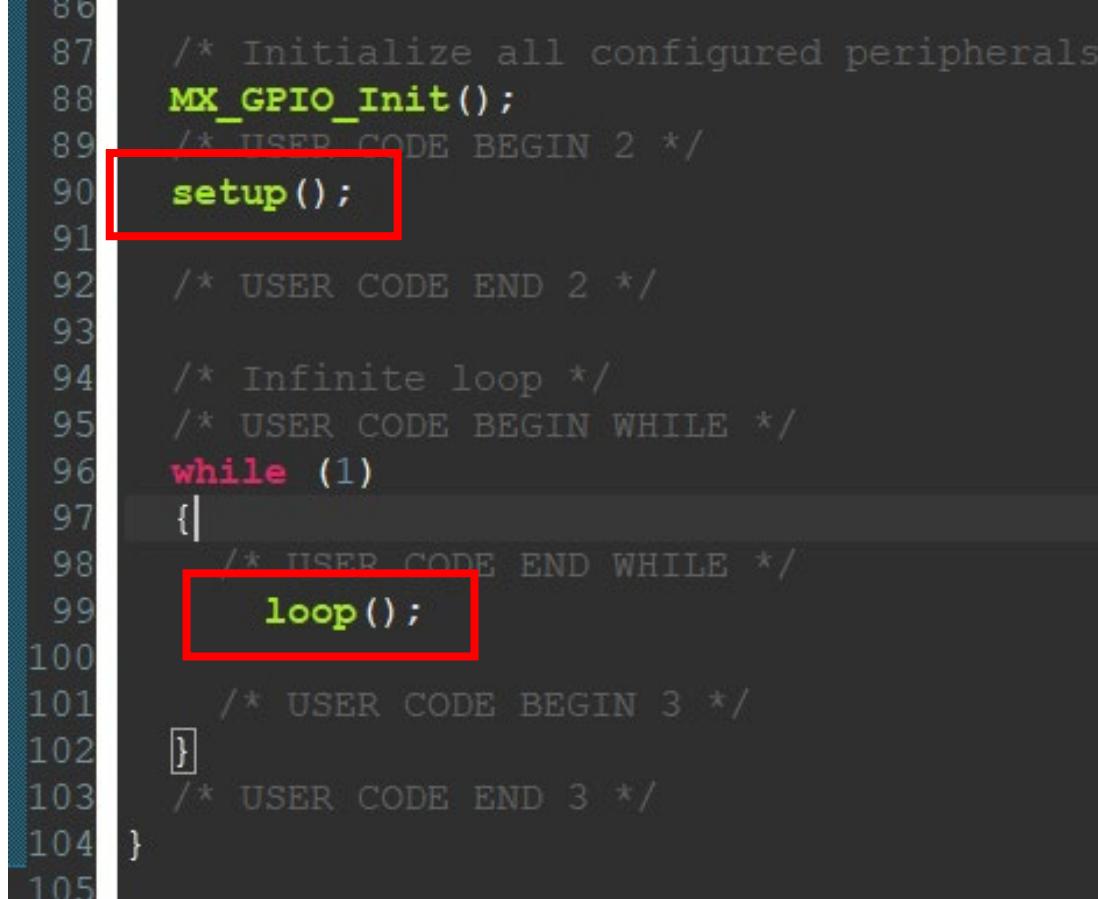
```
MX LED_Blinking_101.ioc    main.c    main.h    CPPmain.h    CPPmain.cpp X
1  /*
2   * CPPmain.cpp
3   *
4   * Created on: Dec 10, 2024
5   * Author: Pannawit
6   */
7
8  #include "CPPmain.h"
9  #include "stdio.h"
10
11 void setup()
12 {
13 }
14
15 void loop()
16 {
17 }
18
19
20
```

Setting up the environment



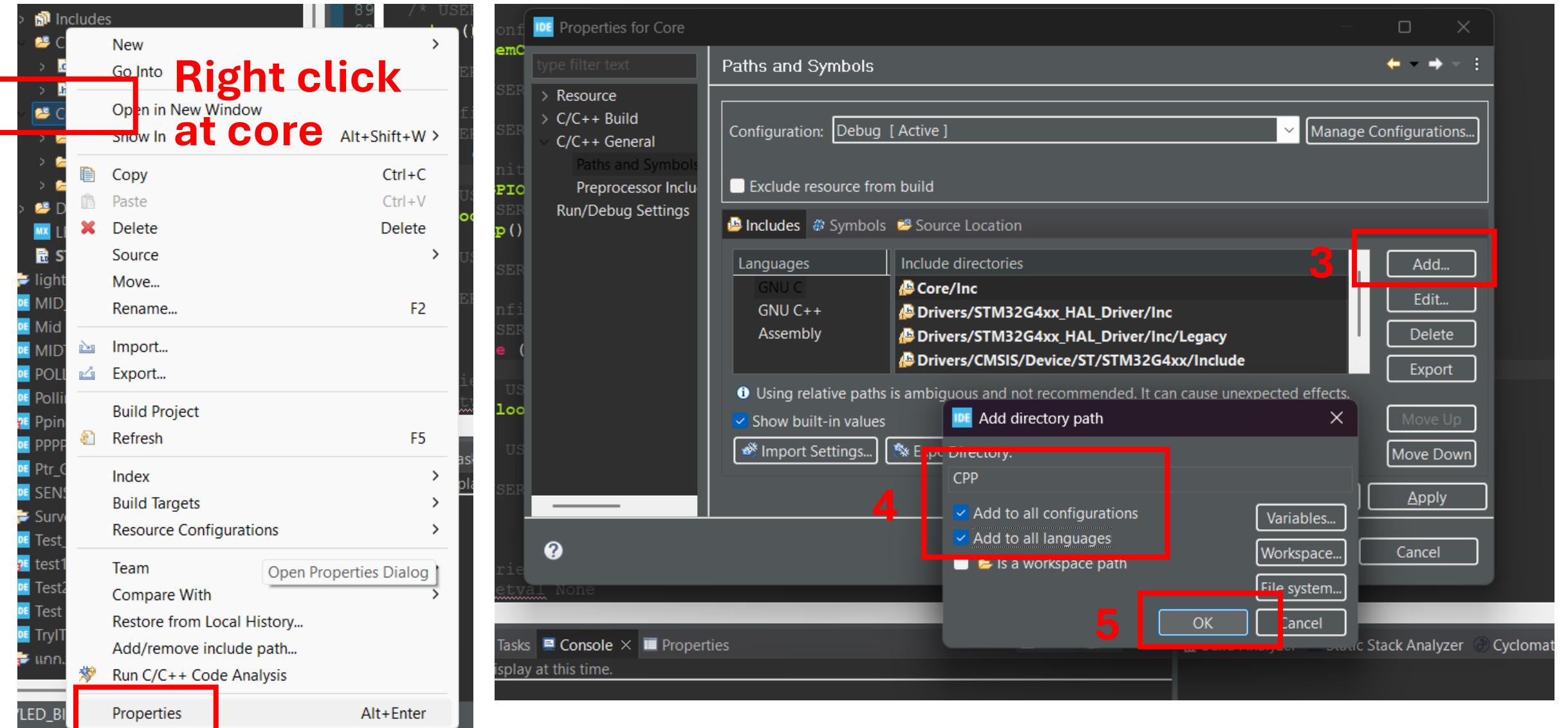
MX LED_Blinking_101.ioc main.c X main.h .h

```
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 */
29
```

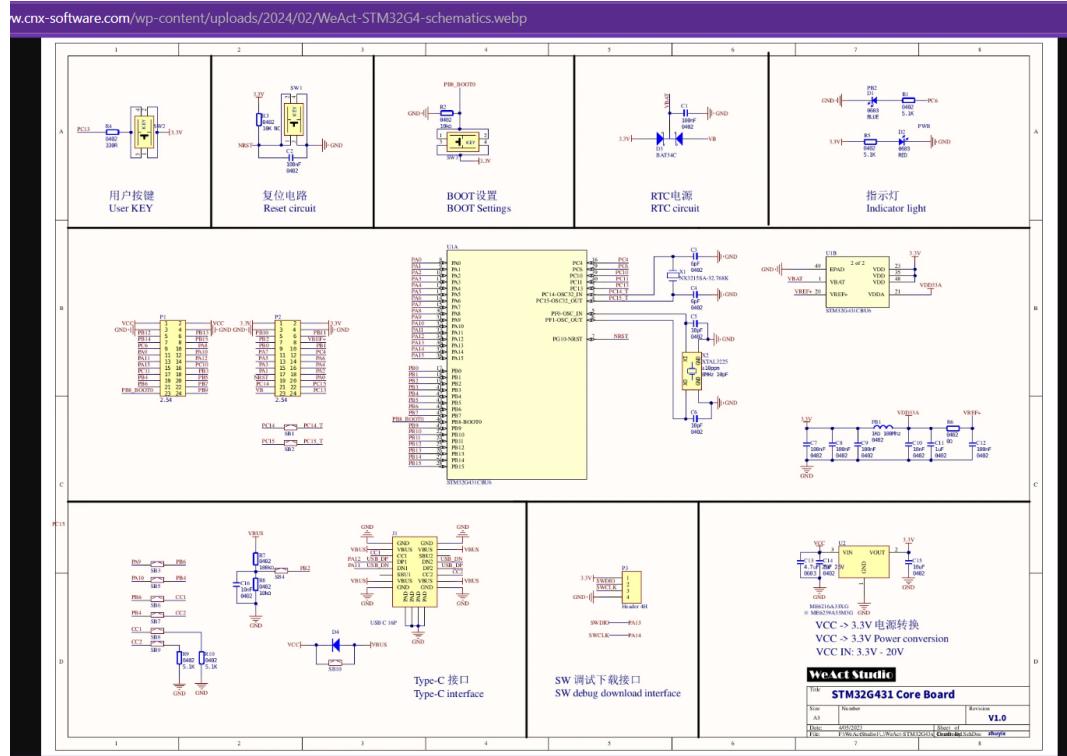


```
86
87     /* Initialize all configured peripherals
88     MX_GPIO_Init();
89     /* USER CODE BEGIN 2 */
90     setup();
91
92     /* USER CODE END 2 */
93
94     /* Infinite loop */
95     /* USER CODE BEGIN WHILE */
96     while (1)
97     {
98         /* USER CODE END WHILE */
99         loop();
100
101        /* USER CODE BEGIN 3 */
102    }
103    /* USER CODE END 3 */
104}
105
```

Setting up the environment



Important thing for embedded system



**STM32G431x6 STM32G431x8
STM32G431xB**

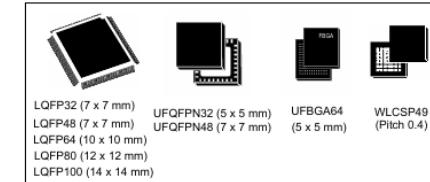
Arm® Cortex®-M4 32-bit MCU+FPU, 170 MHz /213 DMIPS,
up to 128 KB Flash, 32 KB SRAM, rich analog, math accelerator

Datasheet - production data

Features

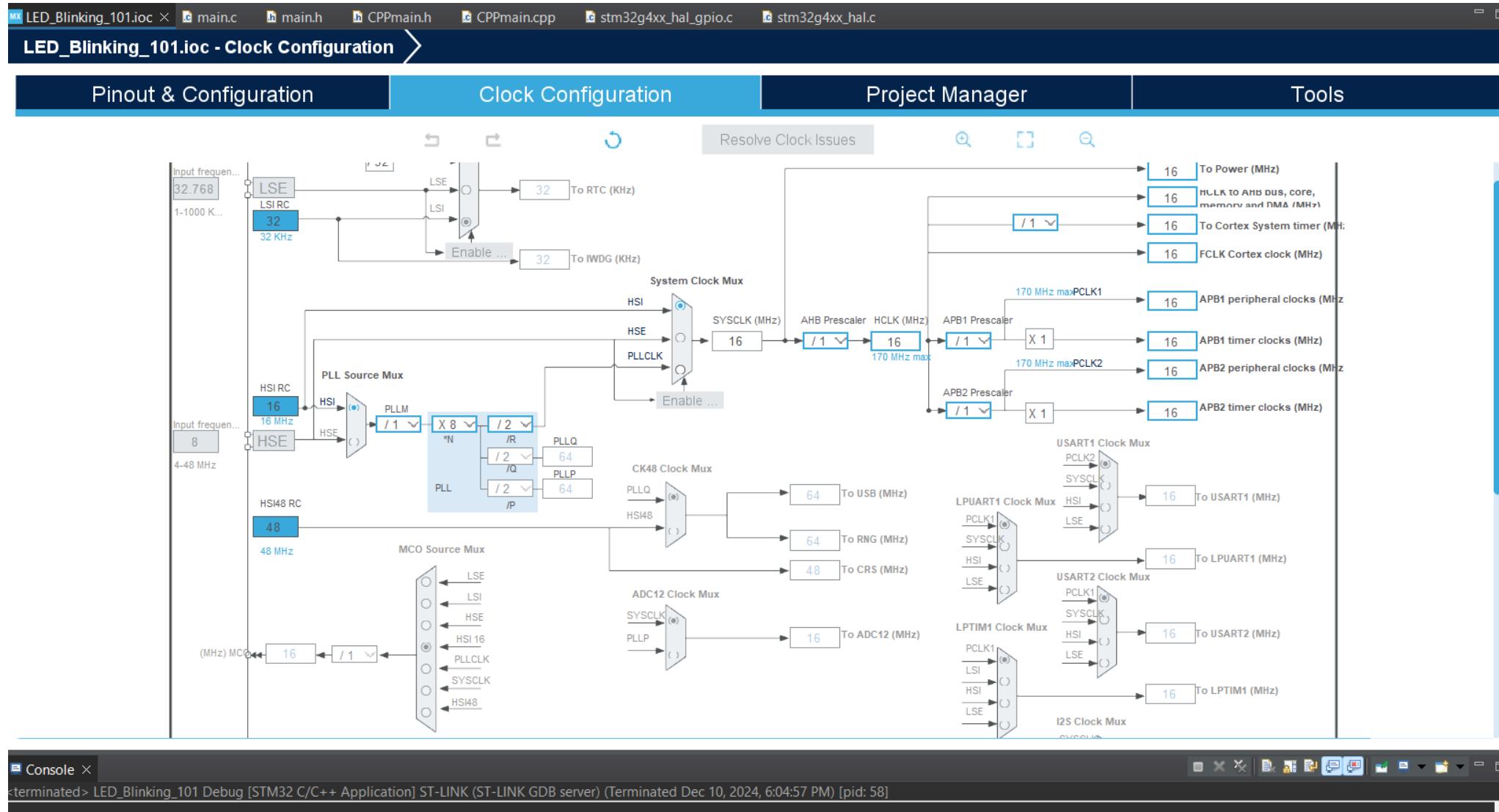
Includes ST state-of-the-art patented technology

- 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), secureable 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 ($\pm 1\%$)
 - Internal 32 kHz RC oscillator ($\pm 5\%$)
- 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 μ s (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_101.ioc

LED_Blinking_101.ioc - Pinout & Configuration

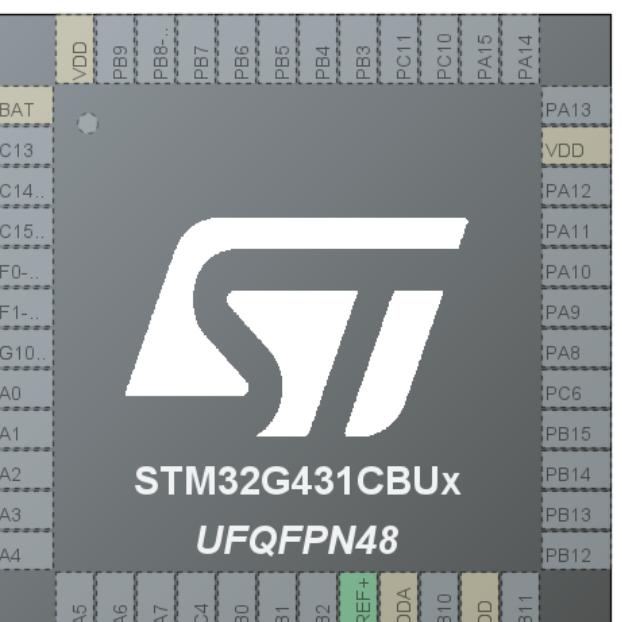
Pinout & Configuration Clock Configuration Project Manager Tools

Software Packs Pinout

Pinout view System view

Categories A-Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware and Software Pac... >
- Utilities >

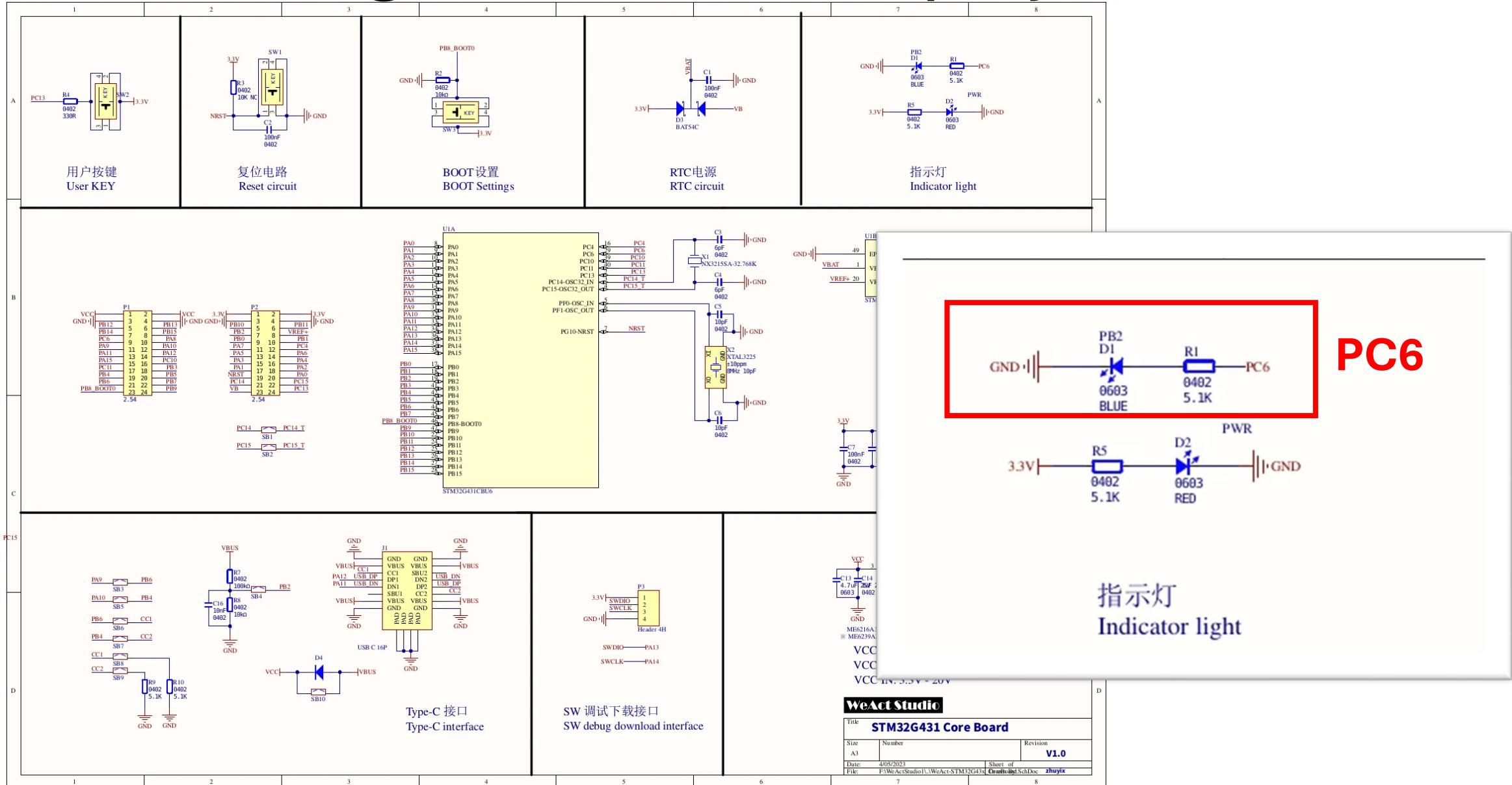


The diagram shows the pinout of the STM32G431CBUx microcontroller in a UFQFPN48 package. The pins are arranged in a grid. The top row contains VDD, PB9, PB8, PB7, PB6, PB5, PB4, PB3, PC11, PC10, PA15, and PA14. The bottom row contains PA5, PA6, PA7, PC4, PB0, PB1, PB2, VREF+, VDDA, PB10, VDD, and PB11. The left column contains VBAT, PC13, PC14.., PC15.., PF0-.., PF1-.., PG10.., PA0, PA1, PA2, PA3, and PA4. The right column contains PA13, VDD, PA12, PA11, PA10, PA9, PA8, PC6, PB15, PB14, PB13, and PB12. The central area features the ST logo and the text "STM32G431CBUx" and "UFQFPN48".

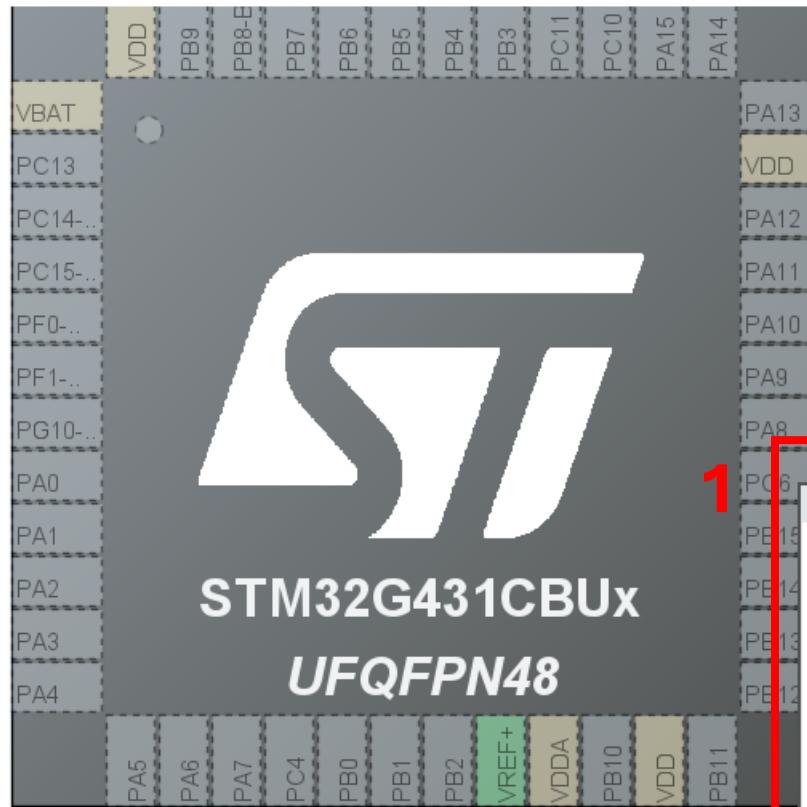
Search Zoom Filter Import Export Console

No consoles to display at this time.

LED Blinking for first STM32 project



LED Blinking for first STM32 project



STM32CubeMX GPIO Mode and Configuration

Categories A-Z

System Core

- DMA
- GPIO**
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Search Signals Search (Ctrl+F) Show only Modified Pins

Pin Na...	Signal ...	GPIO ...	G...	GPIO ...	Maxim...	Fast M...	User L...	Modifi...
PC6	n/a	High	Output...	Pull-up	High	n/a		<input checked="" type="checkbox"/>

1 PC6

2 System Core

3 Configuration

4 User Label

Configuration

Group By Peripherals

GPIO

Search Signals Search (Ctrl+F)

Pin Na... Signal ... GPIO ... G... GPIO ... Maxim... Fast M... User L... Modifi...

PC6 n/a High Output... Pull-up High n/a

GPIO output level High

GPIO mode Output Push Pull

GPIO Pull-up/Pull-down Pull-up

Maximum output speed High

GPIO mode Output Push Pull

GPIO Pull-up/Pull-down Pull-up

Maximum output speed High

User Label LED

LED Blinking for first STM32 project

1 Project Explorer view showing the project structure and files.

2 Drivers/STM32G4xx_HAL_Driver/Src folder containing source files. The file `stm32g4xx_hal_gpio.c` is highlighted with a red box.

3 Outline view showing the symbols defined in the `stm32g4xx_hal.h` header file. The symbol `HAL_GPIO_TogglePin` is highlighted with a red box.

4 Code editor view showing the implementation of the `HAL_GPIO_TogglePin` function in the `stm32g4xx_hal_gpio.c` file. The code implements the logic to toggle the specified GPIO pin based on its current state.

```

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 }

```

5 Outline view showing the symbols defined in the `stm32g4xx_hal.h` header file. The symbol `HAL_GPIO_TogglePin` is highlighted with a red box.

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 }
```

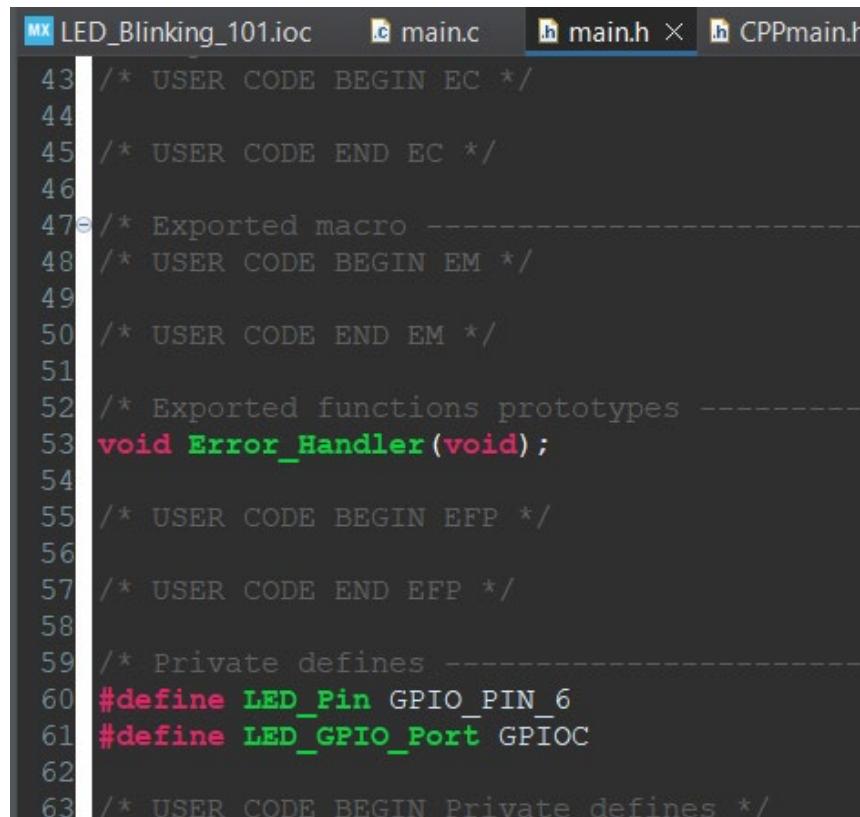
LED Blinking for first STM32 project

```
MX LED_Blinking_101.ioc  c main.c  h main.h  h CPPmain.h  c *CPPmain.cpp ×  c stm32g4xx_hal_gpio.c
18/*
19 * CPPmain.cpp
20 *
21 * Created on: Dec 10, 2024
22 * Author: Pannawit
23 */
24
25#include "CPPmain.h"
26#include "stdio.h"
27
28void setup()
29{
30}
31
32void loop()
33{
34    HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);
35}
```

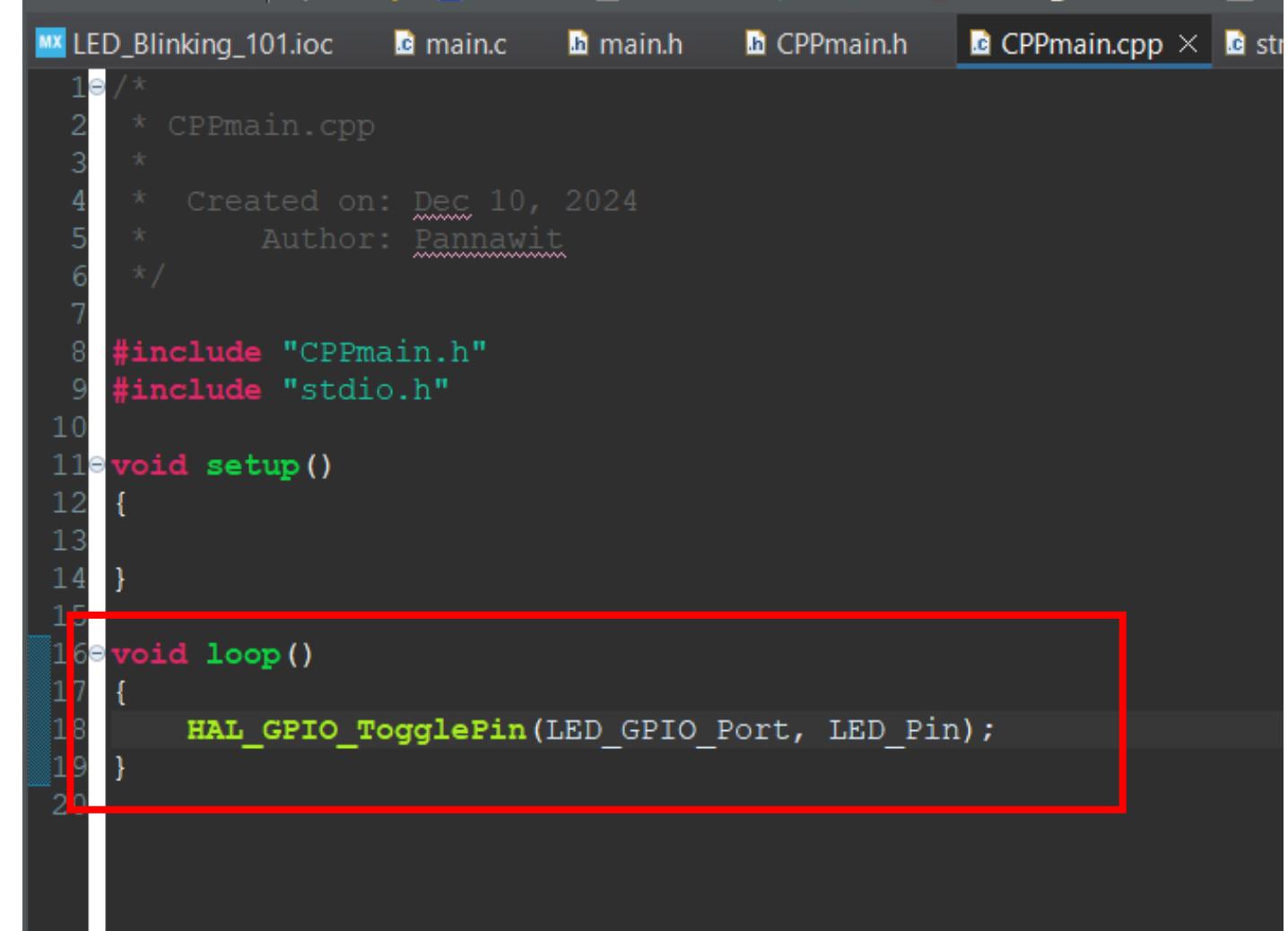
```
MX LED_Blinking_101.ioc  c main.c  h main.h ×  h CPPmain.h  c *CPPmain.cpp  c stm32g4xx_hal_gpio.c
43/* USER CODE BEGIN EC */
44
45/* USER CODE END EC */
46
47/* Exported macro */
48/* USER CODE BEGIN EM */
49
50/* USER CODE END EM */
51
52/* Exported functions prototypes */
53void Error_Handler(void);
54
55/* USER CODE BEGIN EFP */
56
57/* USER CODE END EFP */
58
59/* Private defines */
60#define LED_Pin GPIO_PIN_6
61#define LED_GPIO_Port GPIOC
62
63/* USER CODE BEGIN Private defines */
64
65/* USER CODE END Private defines */
66
```

main.h

LED Blinking for first STM32 project



```
MX LED_Blinking_101.ioc main.c main.h CPPmain.h
43 /* USER CODE BEGIN EC */
44
45 /* USER CODE END EC */
46
47 /* Exported macro */
48 /* USER CODE BEGIN EM */
49
50 /* USER CODE END EM */
51
52 /* Exported functions prototypes */
53 void Error_Handler(void);
54
55 /* USER CODE BEGIN EFP */
56
57 /* USER CODE END EFP */
58
59 /* Private defines */
60 #define LED_Pin GPIO_PIN_6
61 #define LED_GPIO_Port GPIOC
62
63 /* USER CODE BEGIN Private defines */
```



```
MX LED_Blinking_101.ioc main.c main.h CPPmain.h CPPmain.cpp str
1 /* CPPmain.cpp
2 *
3 * Created on: Dec 10, 2024
4 * Author: Pannawit
5 */
6
7
8 #include "CPPmain.h"
9 #include "stdio.h"
10
11 void setup()
12 {
13 }
14
15 void loop()
16 {
17     HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
18 }
19
20
```

LED Blinking for first STM32 project

The screenshot shows the development environment for an STM32 project. The Project Explorer on the left lists various source files and header files, including those from the STM32G4xx_HAL_Driver. A red box highlights the `HAL_Delay` implementation in `CPPmain.cpp`. The code uses SysTick to generate interrupts at regular intervals, with a minimum wait time guaranteed by adding a frequency offset. The `loop` function in `CPPmain.h` toggles an LED pin every 500ms using `HAL_Delay(500)`.

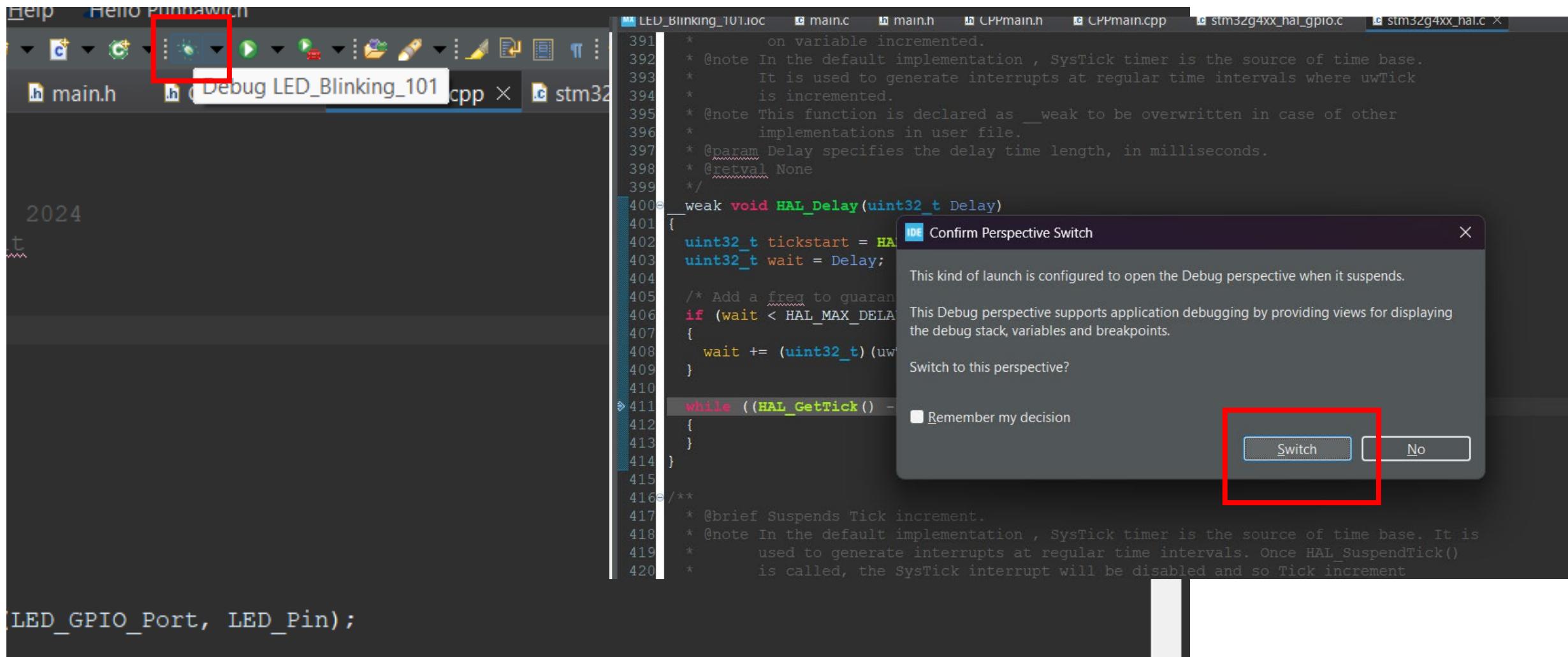
```
Project Explorer ▾ . . . LED_Blinking_101.ioc main.c main.h CPPmain.h CPPmain.cpp

391 *      on variable incremented.
392 * @note In the default implementation , SysTick timer is
393 *      It is used to generate interrupts at regular time
394 *      is incremented.
395 * @note This function is declared as __weak to allow
396 *      implementations in user file.
397 * @param Delay specifies the delay time length,
398 * @retval None
399 */
400 __weak void HAL_Delay(uint32_t Delay)
401 {
402     uint32_t tickstart = HAL_GetTick();
403     uint32_t wait = Delay;
404
405     /* Add a freq to guarantee minimum wait */
406     if (wait < HAL_MAX_DELAY)
407     {
408         wait += (uint32_t) (uwTickFreq);
409     }
410
411     while ((HAL_GetTick() - tickstart) < wait)
412     {
413     }
414 }
415
416 /**
417 * @brief Suspends Tick increment.
418 * @note In the default implementation , SysTick timer is
419 *      used to generate interrupts at regular time intervals.
420 *      is called, the SysTick interrupt will be disabled.
```

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

18 /*
20 * CPPmain.cpp
21 *
22 * Created on: Dec 10, 2024
23 * Author: Pannawit
24 */
25
26 #include "CPPmain.h"
27 #include "stdio.h"
28
29 void setup()
30 {
31 }
32
33 void loop()
34 {
35     HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
36     HAL_Delay(500);
37 }
```

LED Blinking for first STM32 project



LED Blinking for first STM32 project

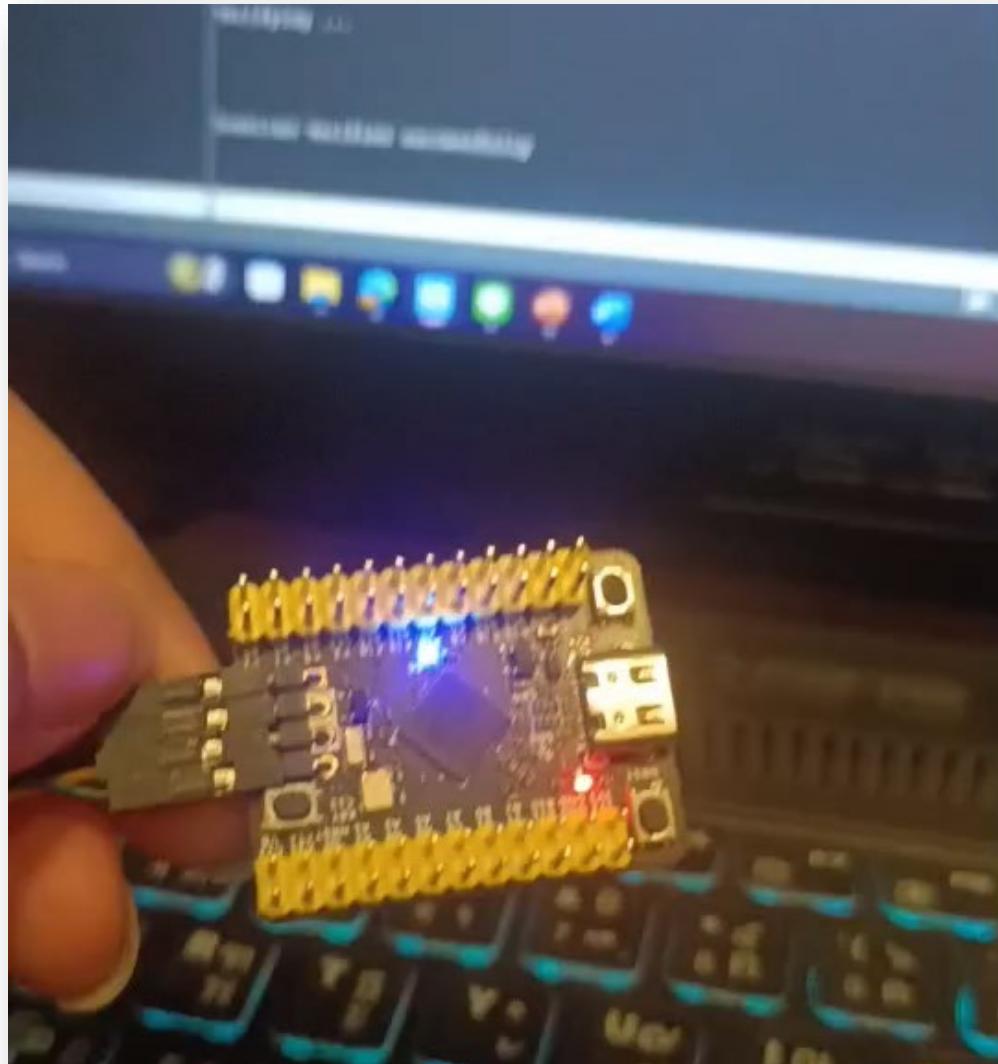
The screenshot shows the STM32CubeIDE interface. The top menu bar includes Navigate, Search, Project, Run, Window, Help, and Hello Punnawich. The toolbar has various icons for file operations, search, and run. A red box highlights the 'Run' button icon (play/pause) in the toolbar. The code editor displays the main.c file with the following code:

```
55 /* Private user code */
56 /* USER CODE BEGIN 0 */
57
58 /* USER CODE END 0 */
59
60 /**
61  * @brief  The application entry point.
62  * @retval int
63  */
64 int main(void)
65 {
66
67  /* USER CODE BEGIN 1 */
68
69  /* USER CODE END 1 */
70
71  /* MCU Configuration-----*/
72
73  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
74  HAL_Init();
75
76  /* USER CODE BEGIN Init */
77
78  /* USER CODE END Init */
79
80  /* Configure the system clock */
```

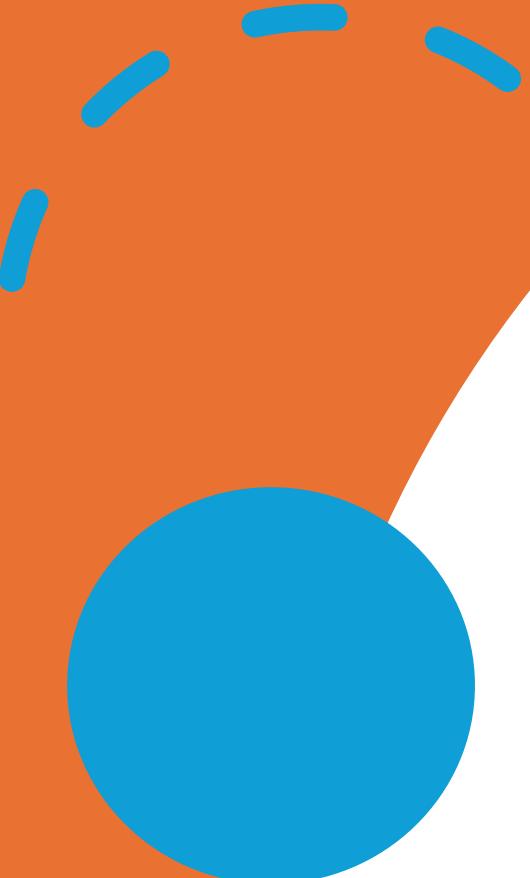
The Variables view on the right shows a table with three columns: Name, Type, and Value, all currently empty.

In the bottom left, the Console tab shows the output: "LED_Blinking_101 Debug [STM32 C/C++ Application] [pid:58]" and "Time elapsed during download operation: 00:00:00.267".

LED Blinking for first STM32 project



Congratulation
Now, you guys can do the STM32 project



Intro to peripherals

The communication between
microcontroller and peripherals

Cặn h̄ụy s̄ụt̄ọ Ạv̄ụr̄ ụ̀ Ọ ụ̉s̄ḡeạ̈v̄ ụ̉Ạ
J̄w̄ậv̄ḡv̄ụ



Communication is the key

Ò ř ēâ Ä
È ì Ä Æ sú Ä
ŷv Ei ý

NĂKŇA
NĂKŇA
LĂKŇ

Ě Á

U È

ÅÄGÄ Ÿyé7



SOME WORDs

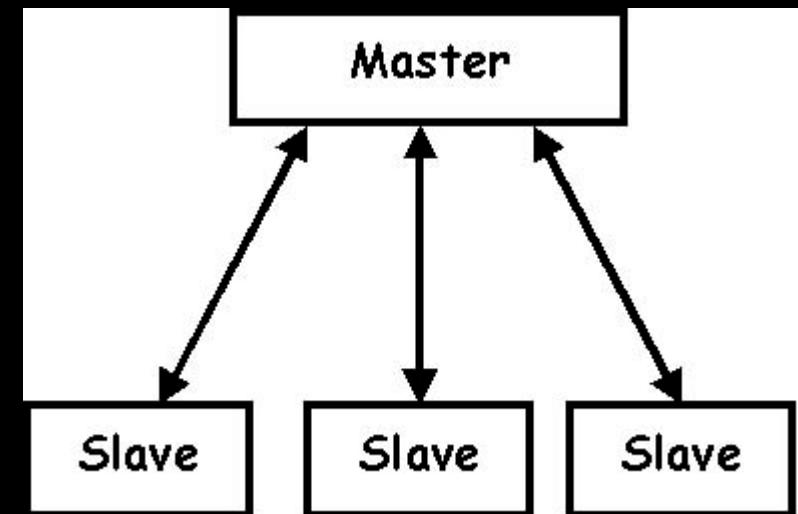
Duplex

Simplex

Half Duplex

Full Duplex

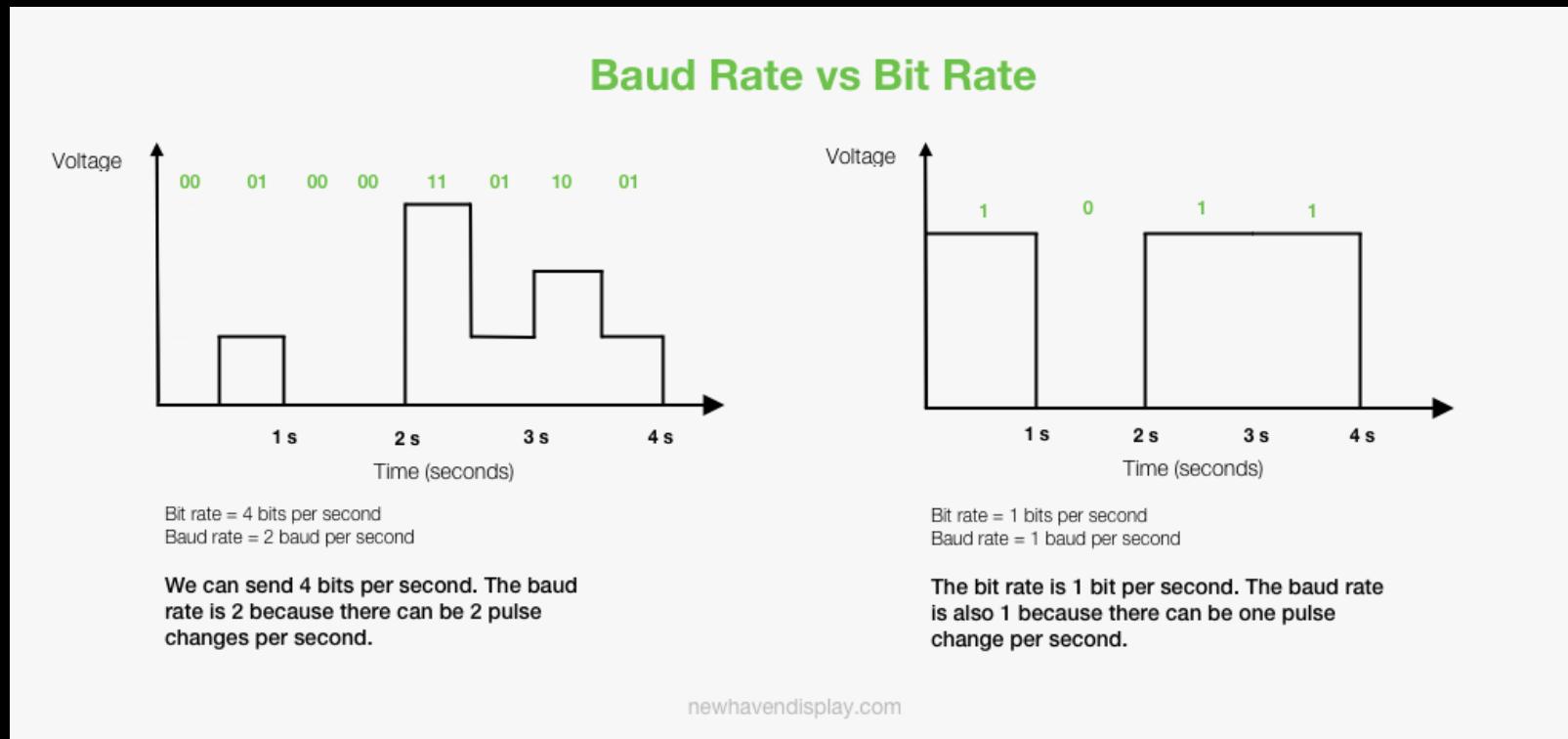
Master-slave



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

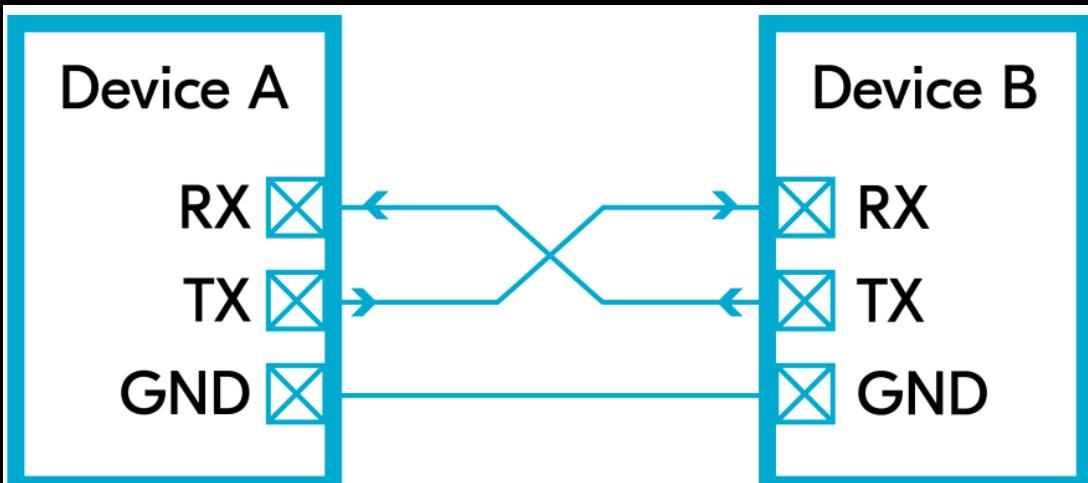
Basics Serial communication, full-duplex, point-to-point.

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.

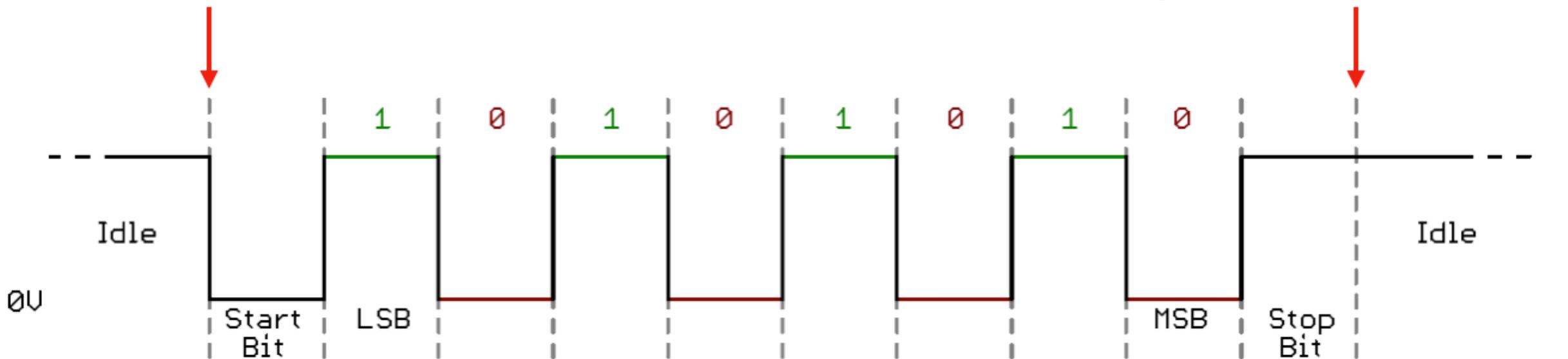




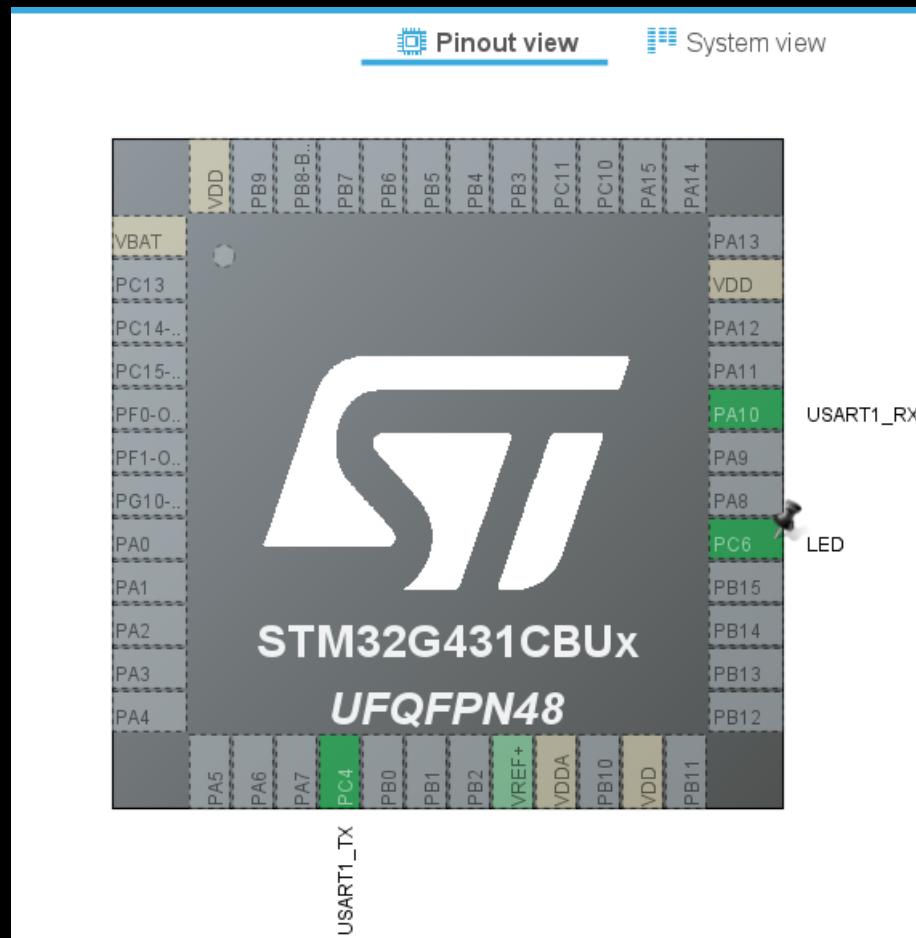
Between 2 devices their baud rate must be the same!

Łaoří eý Čeří Čeřá
—??č#\\" ??č#—??č—??čA— ??č##— ??čúř Čr vý

Receiver starts its clock



Mini task



```
0
1
2
3
4
5
6
7
8 #include "CPPmain.h"
9 #include "stdio.h"
10
11 extern UART_HandleTypeDef huart1;
12
13 uint8_t data[10];
14 uint8_t time = 0;
15
16 void setup()
17 {
18     HAL_UART_Receive_IT(&huart1, data, sizeof(data));
19 }
20
21 void loop()
22 []
23     time = (data[0] << 8) | data[1];
24
25     HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
26     HAL_Delay(time);
27
28     HAL_UART_Receive(&huart1, data, sizeof(data), 1000);
29 }
```

I2C

Inter-Integrated Circuit

Basics

Two-wire protocol (SDA and SCL).

Addressing mechanism.

How I2C Works

Start and stop conditions, acknowledgment, and data transfer.

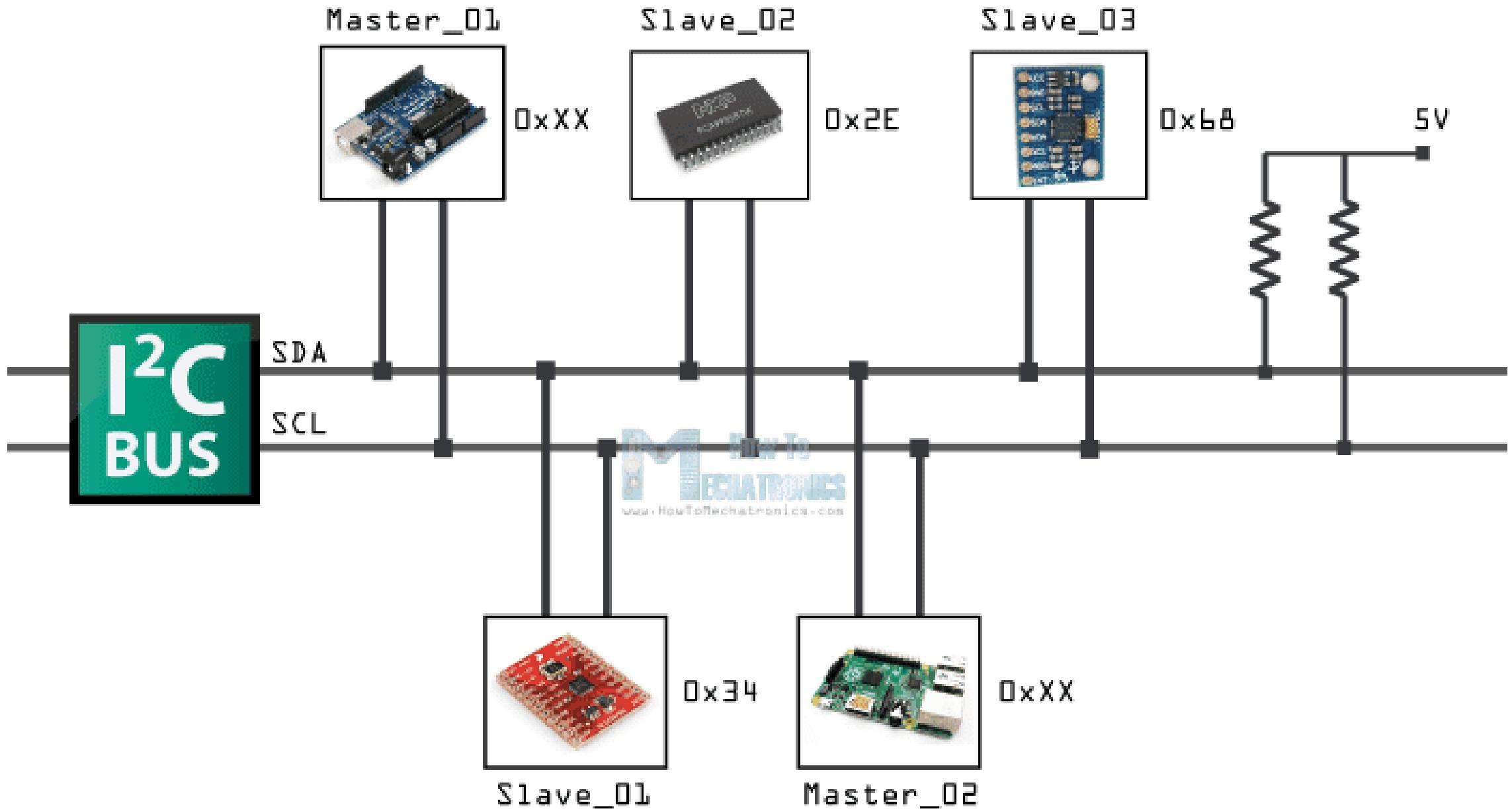
Applications in Robotics

Sensor data acquisition, interfacing IMUs or temperature sensors

I2C EXPLAINED!



robocraze



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=0CBQQjRxqFwoTCKCGkISRqlDFQAAAAAdAAAAAB

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).

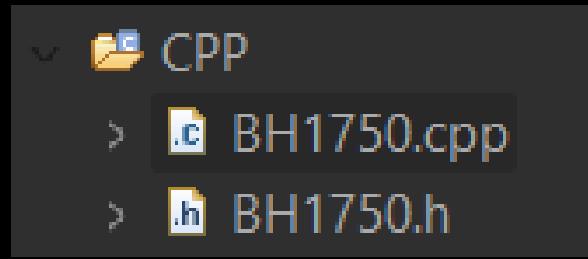
● Features

- 1) I²C bus Interface (f / s Mode Support)
- 2) Spectral responsivity 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 AC 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



Ň ŠÄVÎ Ï ÄŠÄVÄRÌ ÄÍ ÐÄ

```
7  
8 #ifndef BH1750_H_  
9 #define BH1750_H_  
10  
11 #include "main.h"  
12  
13 class BH1750  
14 {  
15 public:  
16  
17     void BH1750_Init(void);  
18  
19     void BH1750_Start(void);  
20  
21     uint16_t BH1750_ReadLight(void);  
22  
23 };  
24  
25 #endif /* BH1750_H_ */  
26 |
```

Êì èî ì ýÄù

```
7  
8 #include "BH1750.h"  
9  
10 extern I2C_HandleTypeDef hi2c1;  
11  
12 #define BH1750_ADDRESS (0x23 << 1)  
13 #define BH1750_POWER_ON 0x01  
14 #define BH1750_CONT_H_RES_MODE 0x10  
15  
16 void BH1750::BH1750_Init(void)  
17 {  
18     uint8_t power = BH1750_POWER_ON;  
19     HAL_I2C_Master_Transmit(&hi2c1, BH1750_ADDRESS, &power, 1, HAL_MAX_DELAY);  
20 }  
21  
22 void BH1750::BH1750_Start(void)  
23 {  
24     uint8_t start = BH1750_CONT_H_RES_MODE;  
25  
26     HAL_I2C_Master_Transmit(&hi2c1, BH1750_ADDRESS, &start, 1, HAL_MAX_DELAY);  
27 }  
28  
29 uint16_t BH1750::BH1750_ReadLight(void)  
30 {  
31     uint8_t data[2];  
32  
33     HAL_I2C_Master_Receive(&hi2c1, BH1750_ADDRESS, data, 2, HAL_MAX_DELAY);  
34  
35     uint16_t lux = (data[0] << 8) | data[1];  
36  
37     return lux / 1.2;  
38 }  
39 |
```

Ľvºýgì Äù

SPI

Serial Peripheral Interface

Basics

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

Full-duplex communication, master-slave architecture.

How SPI Works

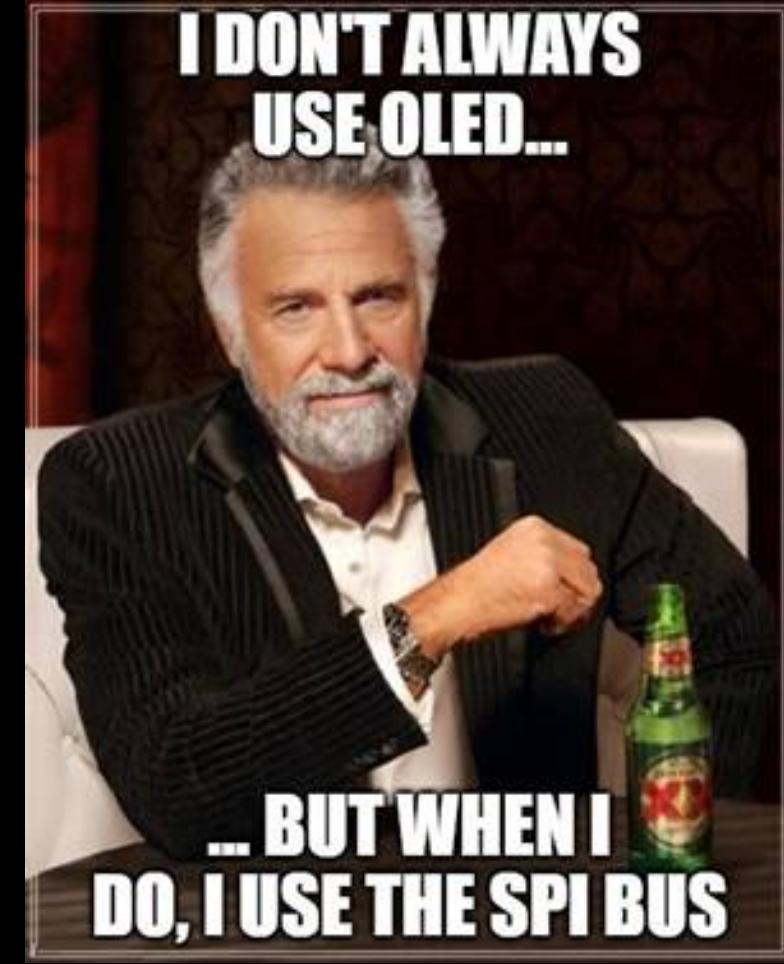
Clock-driven data exchange, roles of master and slave devices.

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**

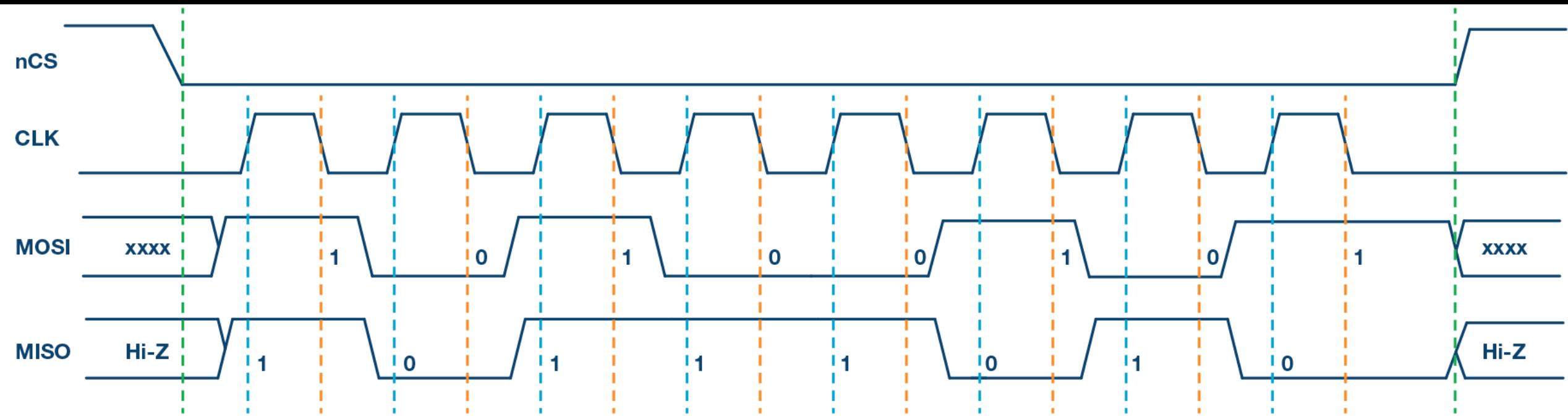
SCLK

MOSI

MISO

\overline{CS}

**SPI
Slave**



CAN

Serial Peripheral Interface

- **Basics**

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

Advantages and Use Cases

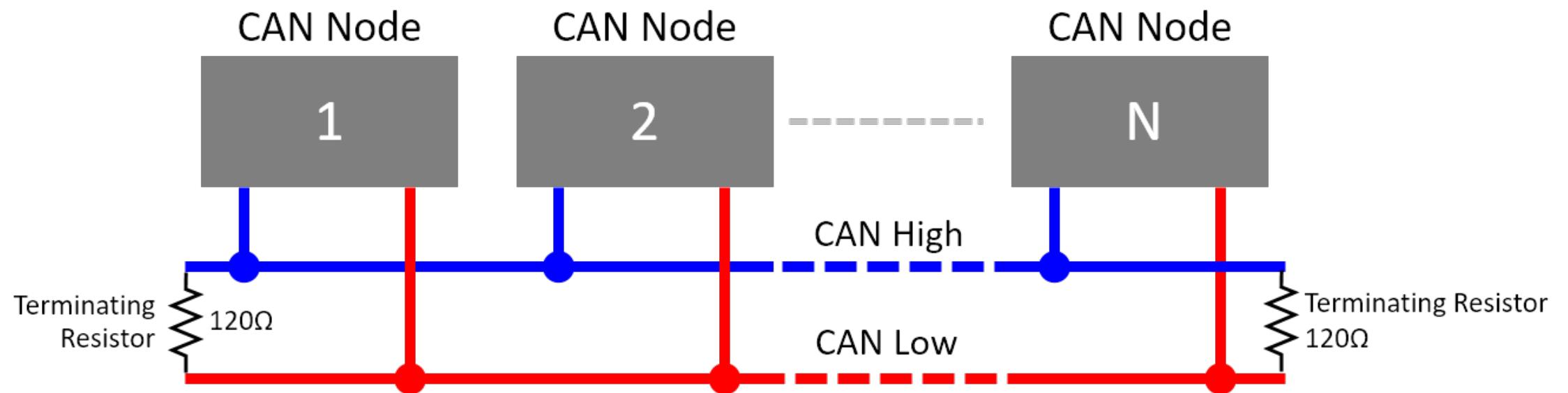
Robust error-checking, real-time performance, ideal for automotive and industrial robotics.

Applications

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



imgflip.com



Special protocol

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

How to manage with every real sensor



Example in one-wire temperature sensor of DS18B20

Requirement:

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

*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.

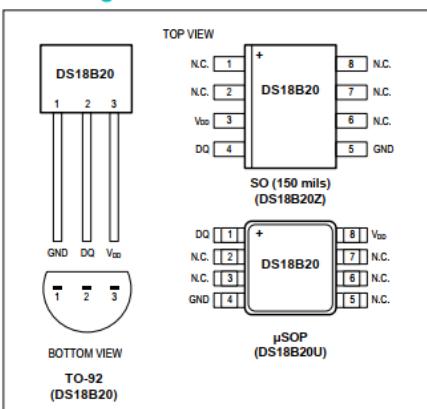
Applications

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

Benefits and Features

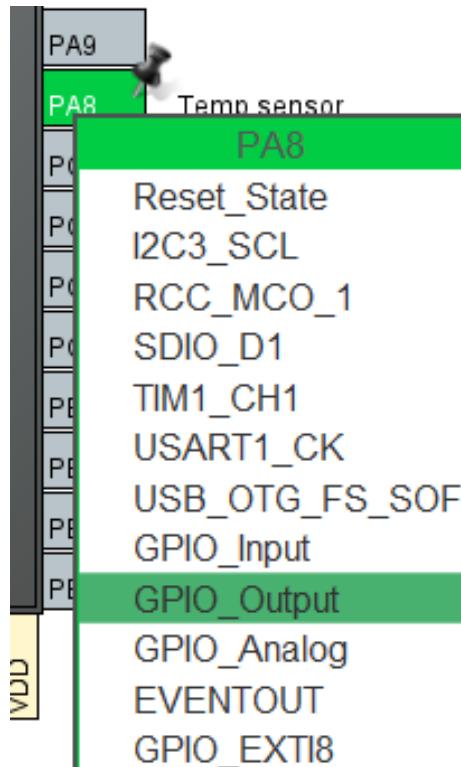
- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
 - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
 - ±0.5°C Accuracy from -10°C to +85°C
 - Programmable Resolution from 9 Bits to 12 Bits
 - No 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



Ordering Information appears at end of data sheet.

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



GPIO output level

GPIO mode

GPIO Pull-up/Pull-down

Maximum output speed

High

Output Open Drain

Pull-up

Very High

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 }
34

```

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.

```

34
35 void sendByte(uint8_t data)
36 {
37     // send byte in hex form with LSB
38     for(int i=0 ; i<8 ; i++)
39     {
40         if((data >> i) & 1)
41         {
42             HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_RESET);
43             Delay_us(5);
44
45             HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_SET);
46             Delay_us(75);
47         }
48         else
49         {
50             HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_RESET);
51             Delay_us(65);
52
53             HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_SET);
54             Delay_us(15);
55         }
56     }
57 }
```

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.

```

57
58 uint8_t ReadBit()
59 {
60     uint8_t bit_value;
61
62     HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_RESET);
63     Delay_us(5);
64
65     HAL_GPIO_WritePin(Temp_sensor_GPIO_Port, Temp_sensor_Pin, GPIO_PIN_SET);
66     Delay_us(10);
67
68     bit_value = HAL_GPIO_ReadPin(Temp_sensor_GPIO_Port, Temp_sensor_Pin) == GPIO_PIN_SET;
69     Delay_us(50);
70
71     return bit_value;
72 }
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 (TH, TL, 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 }
105

```

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	B Eh	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.

```
2 void setup()
3 {
4     HAL_TIM_Base_Start(&htim2);
5
6
7 void loop()
8 {
9     // Temperature
10    startInitial();
11    sendByte(0xCC);
12    sendByte(0x44);
13    HAL_Delay(1000);
14
15    startInitial();
16    sendByte(0xCC);
17    sendByte(0xBE);
18
19    temp = Read_temp();
20}
```

ADC Read

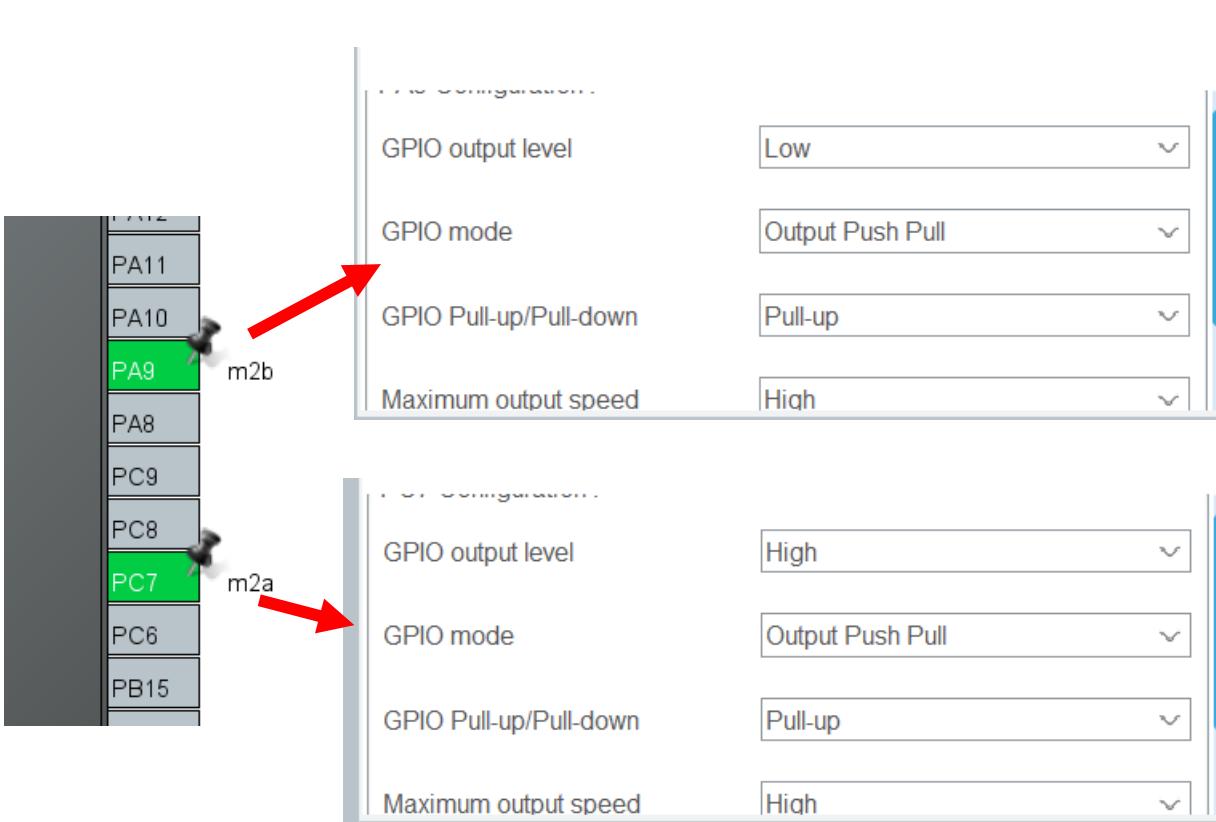
In this workshop, we provide a complete sensors, so you don't need to use the protocol from the previous session.

The screenshot shows the NXP i.MX RT Studio interface. On the left, the navigation pane lists categories like System Core, Analog, Timers, Connectivity, Multimedia, Computing, and Middleware and Software Packs. Under Analog, 'ADC1' is selected and highlighted in blue. The main workspace has two tabs: 'Mode' and 'Configuration'. In the Mode tab, 'IN0' is checked. In the Configuration tab, there is a 'Reset Configuration' button and three checkboxes for NVIC Settings, DMA Settings, and GPIO Settings, all of which are checked. To the right of the interface is a block of C code:

```
7
8 #include "CPPmain.h"
9 #include "stdio.h"
10
11 extern ADC_HandleTypeDef hadc1;
12 uint32_t adc_value;
13
14 void setup()
15 {
16 }
17
18 void loop()
19 {
20     HAL_ADC_Start(&hadc1);
21     if (HAL_ADC_PollForConversion(&hadc1, 1000) == HAL_OK)
22     {
23         adc_value = HAL_ADC_GetValue(&hadc1);
24     }
25 }
26
27
28 }
```

Motor Drive

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



```
7 #include "CPPmain.h"
8 #include "stdio.h"
9
10 extern ADC_HandleTypeDef hadc1;
11 uint32_t adc_value;
12
13
14 void setup()
15 {
16 }
17
18
19 void loop()
20 {
21     HAL_ADC_Start(&hadc1);
22     if (HAL_ADC_PollForConversion(&hadc1, 1000) == HAL_OK)
23     {
24         adc_value = HAL_ADC_GetValue(&hadc1);
25     }
26
27     // Left Motor
28     HAL_GPIO_WritePin(m2a_GPIO_Port, m2a_Pin, GPIO_PIN_SET);
29     HAL_GPIO_WritePin(m2b_GPIO_Port, m2b_Pin, GPIO_PIN_RESET);
30 }
```

Conclusion

3 steps for STM32



DESIGN WHAT
YOU WANT



SETTING UP
IOC

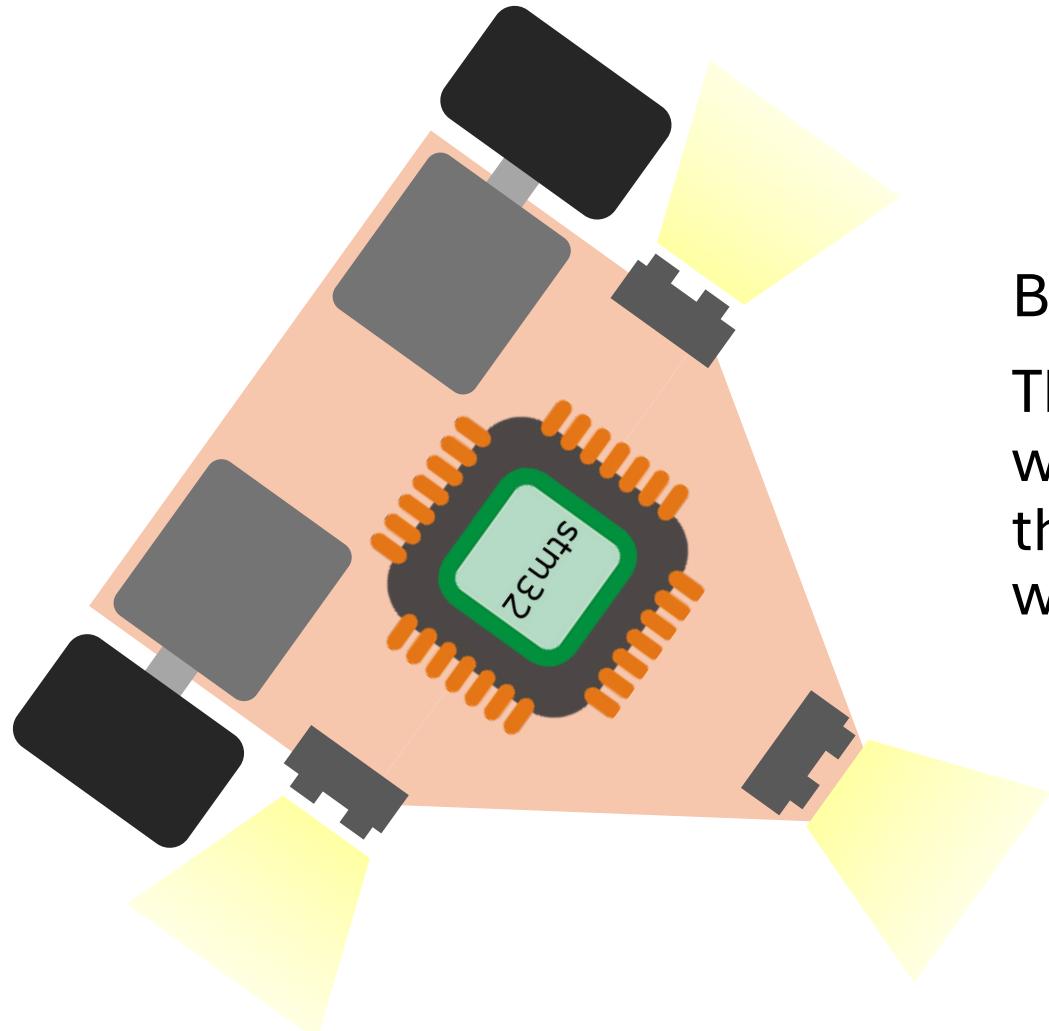


CODING

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

