

1.Introduction

DR.SOMSIN THONGKRAIRAT



life.augmented

ARDUINO

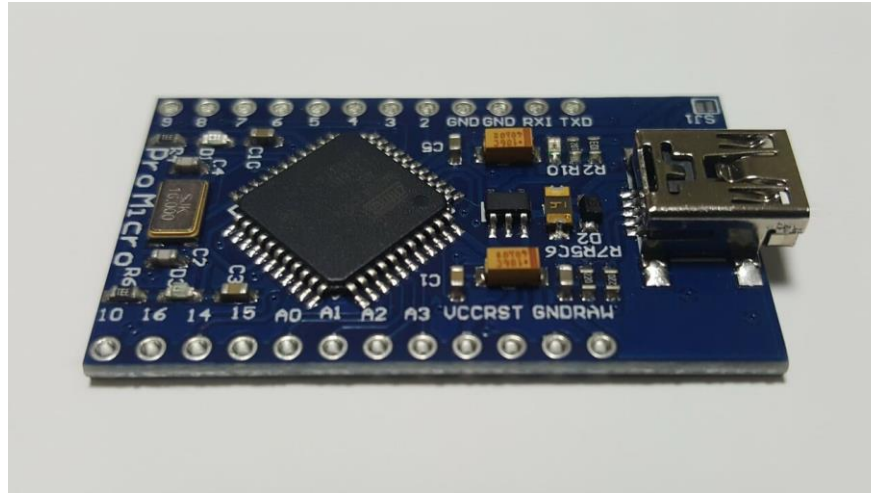


STM32



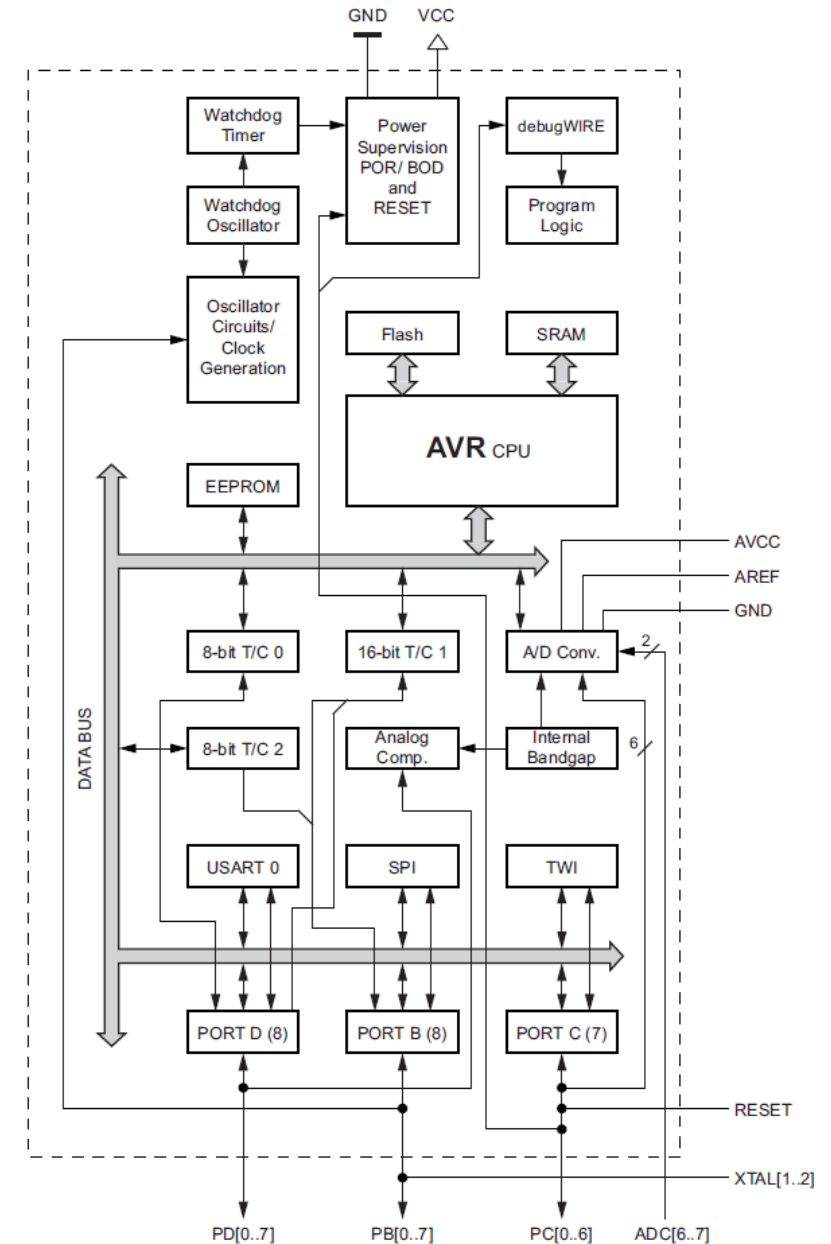
What is MCU (microcontroller unit)

- น่าจะเรียนไปแล้วในหลายๆ วิชาก่อนหน้านี้ (ถ้ายังไม่ลืมนะครับ - -)
- หน่วยประมวลผลขนาดเล็ก มี CPU memory (RAM ROM) และ peripheral



What's inside MCU

- มีอะไรนอกจาก CPU และ memory? บ้าง
- Serial, I2C, SPI, ADC, etc.
- Peripheral

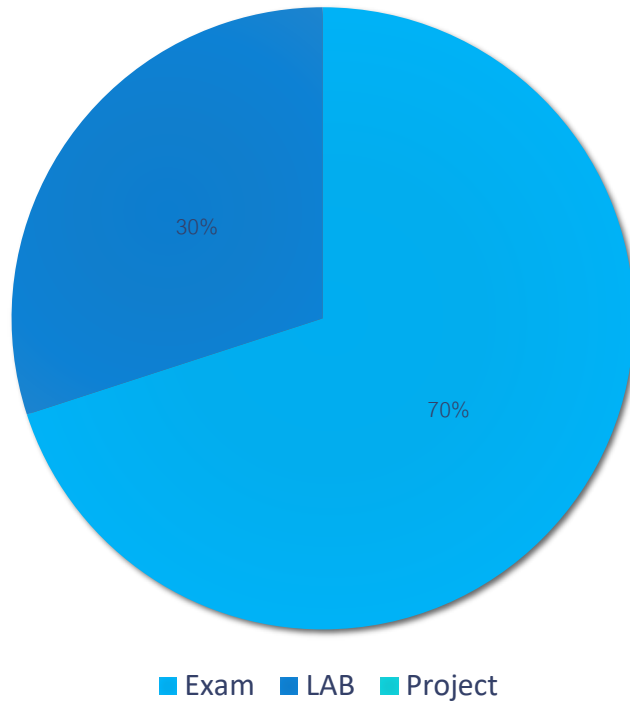


Microcontroller part 1

- เข้าใจการทำงานของ microcomputer เบื้องต้น
- Register, Peripheral, Interrupt คืออะไร
- By STM32 MCU using C language

Scoring

Sales



LAB : in-Class Score [30%]

Exam : midterm and final Exam [70%]

Project : project base Exam [0%] ???

Extra point : Quiz [0-5% (additional)]

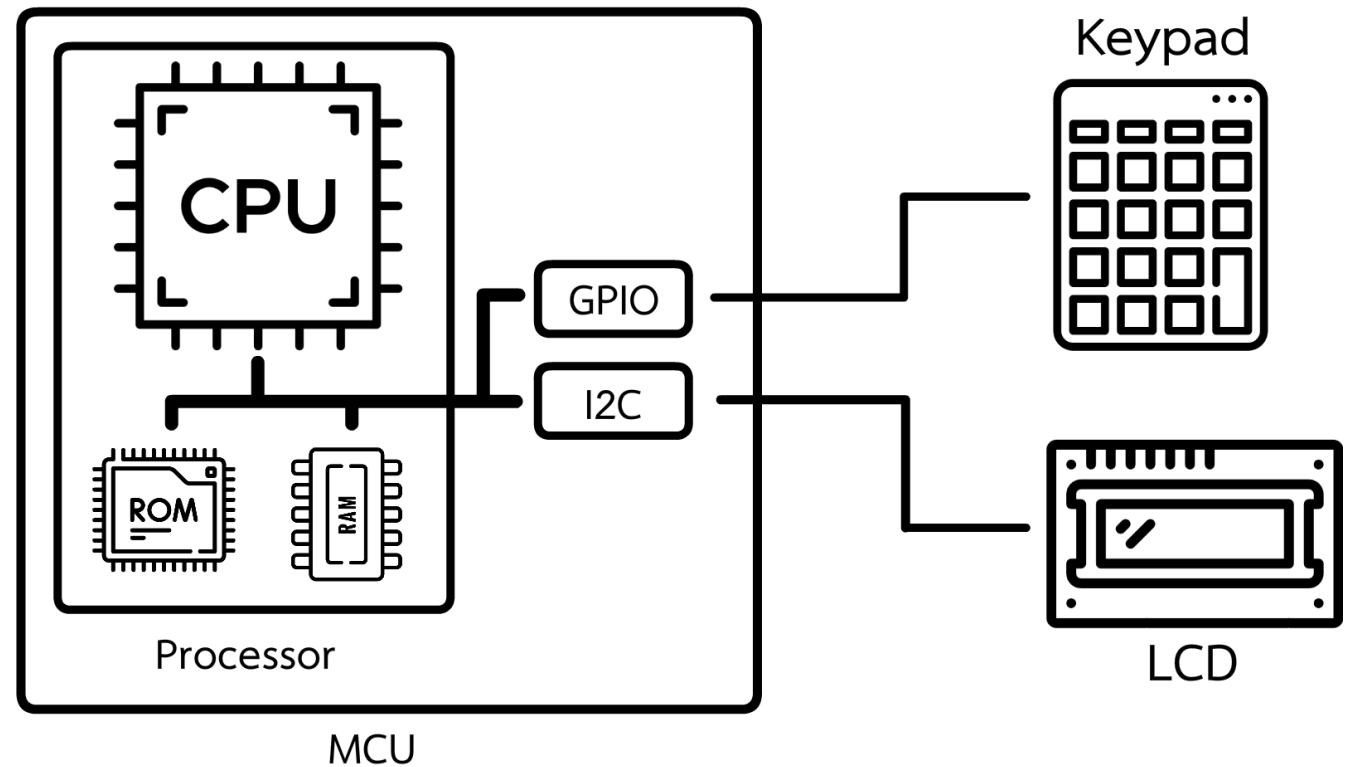
Exam : choice

What is Peripherals

- ส่วนประกอบอื่นๆ ของ ระบบ ที่ช่วยในการทำงาน , ส่งข้อมูล หรือเติมเต็มการทำงานให้กับระบบ
- ไม่จำเป็นต้องมี ระบบก็สามารถประมวลผลได้
- เช่น Serial, timer เป็นต้น
- microcontroller = microprocessor + Peripherals

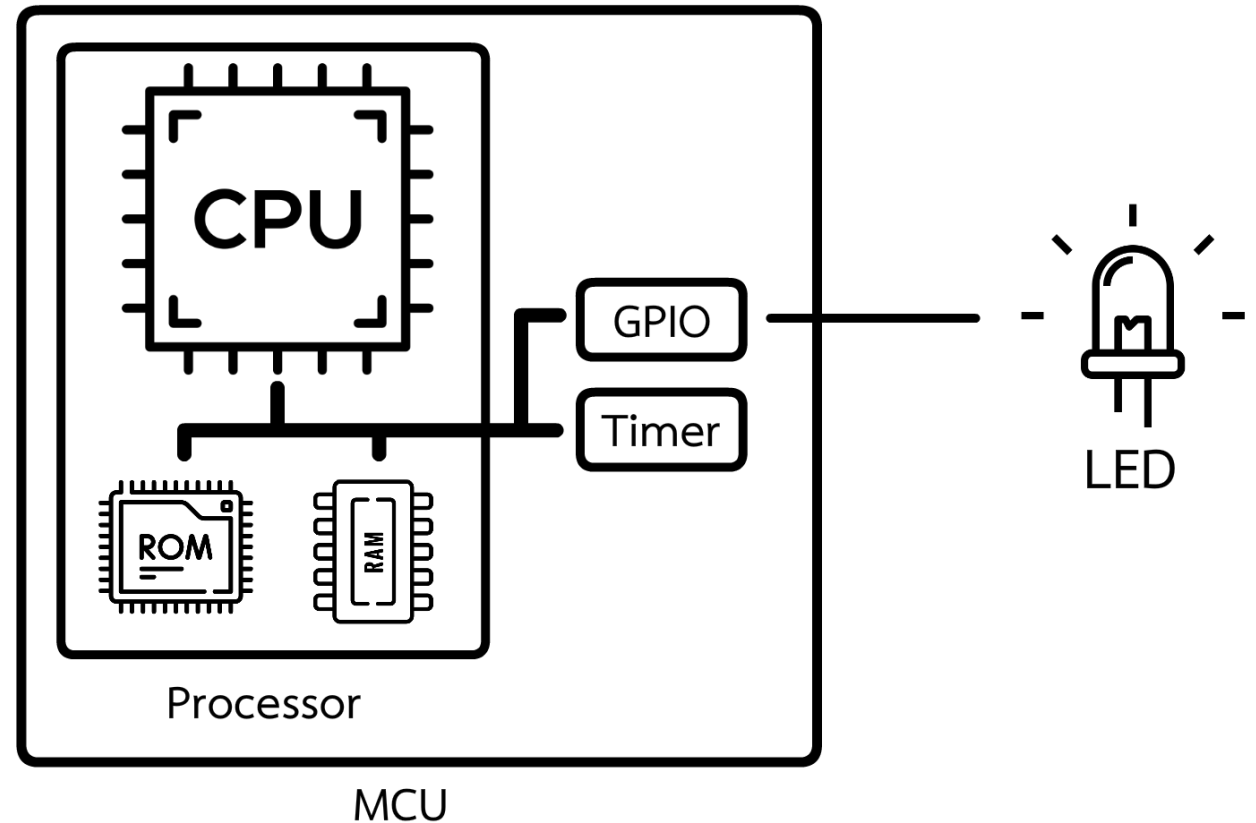
ตัวอย่าง Peripherals (Calculator)

- การประมวลผล บวกเลข ลบเลข เกิดขึ้นในที่ CPU กับ RAM วิธีการ บวกเลข ลบเลข ถูกบรรจุไว้ใน ROM
- ต้องการติดต่อกับ อุปกรณ์ภายนอก เช่น keypad LCD ผ่านทาง GPIO และ I2C



ตัวอย่าง Peripherals (blink)

- โปรแกรมไฟกระพริบทุกๆ 1 วินาที
- Timer ทำหน้าที่จับเวลา ถ้าถึง 1 วินาทีแล้ว จะไปบอกให้ CPU รู้ว่าถึงเวลาที่จะต้องเปลี่ยนสถานะของ GPIO แล้ว
- Peripherals ไม่จำเป็นต้องติดต่อกับข้างนอก MCU



Peripherals (scope)

- GPIO (General purpose Input and Output)
- Timer (PWM and interrupt)
- ADC (Analog digital Convertor)
- USART(Serial + SPI) , I2C
- EEPROM (basic nonvolatile memory)
- DMA (Direct memory access) <- so useful!

How to control peripheral in MCU

- using Register -> super hard coding (C, assembly)

0110
1001
1010

- using hardware abstraction layer (HAL)

- using Platform -> Arduino



using Register

7.4.1 GPIO port mode register (GPIOx_MODER) (x = A to F)

Address offset: 0x00

Reset value: 0xEBFF FFFF for port A

Reset value: 0xFFFF FFFF for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **MODE[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O mode.

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)

7.4.6 GPIO port output data register (GPIOx_ODR) (x = A to F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OD[15:0]**: Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR register (x = A..D, F).

From data sheet [rm0444-stm32g0x1-advanced-armbased-32bit-mcus-stmicroelectronics.pdf] page 241 and 243

Using Register

```
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    /* USER CODE END WHILE */
```

```
    /* USER CODE BEGIN 3 */
```

```
    //      0b3322222222221111111111100000000000|
```

```
    //      0b10987654321098765432109876543210
```

```
    GPIOA->MODER = 0b00000000000000000000000100000000;
```

```
    GPIOA->ODR = 0b0000000000000000000000000000010000;
```

```
    HAL_Delay(1000);
```

```
    GPIOA->ODR = 0b0000000000000000000000000000000000;
```

```
    HAL_Delay(1000);
```

```
}
```

```
/* USER CODE END 3 */
```

```
}
```

Set pin mode

Set pin state

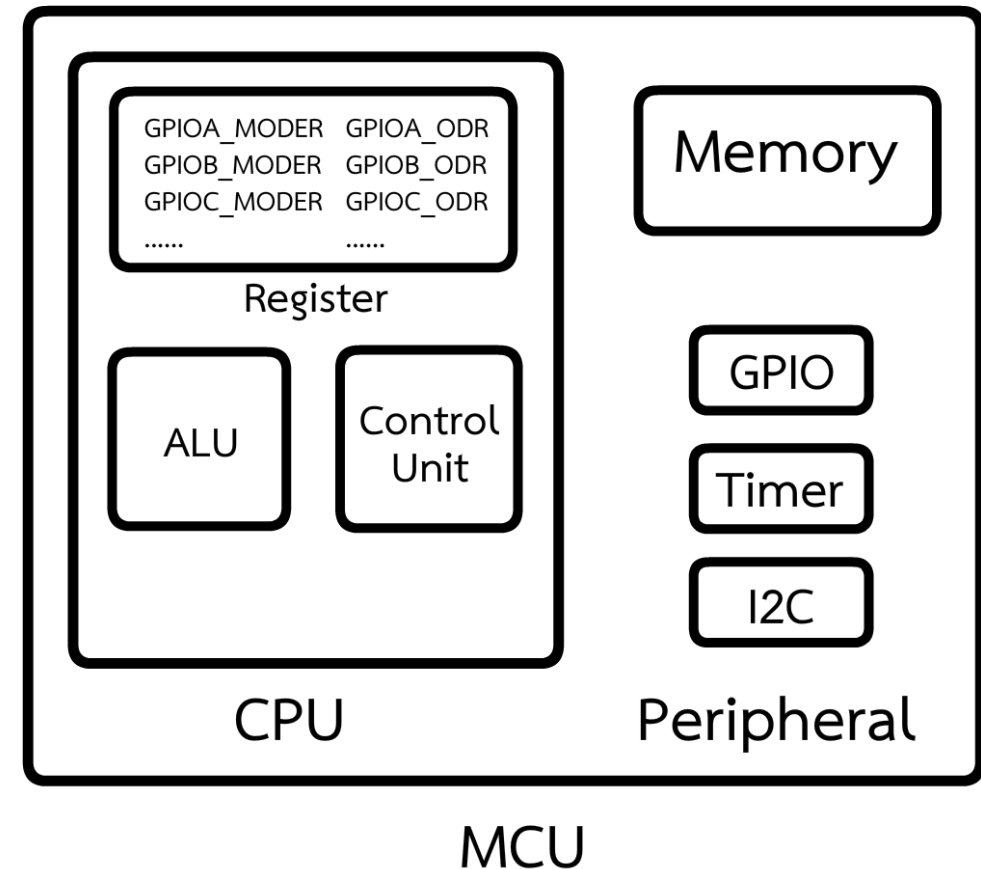
Reset pin state

What is Register (in microcomputer)

- หน่วยความจำพื้นฐานที่อยู่ภายใน CPU
- ใช้กำกับการทำงานของระบบ (ขึ้นอยู่กับรูปแบบ สถาปัตยกรรม) คำอธิบายส่วนใหญ่อยู่ใน datasheet
- เป็น Memory ที่ CPU เข้าถึงได้เร็วที่สุดในระบบ

Register

- ใน CPU จะมี set ของ register ขึ้นอยู่กับ สถาปัตยกรรม
- Register แต่ละตัวจะมีหน้าที่เฉพาะของแต่ละตัว
- Register จะกำหนดการทำงานของ CPU ทั้งภายใน (การคำนวณ) และภายนอก (ควบคุม peripheral)



Register (pros.)

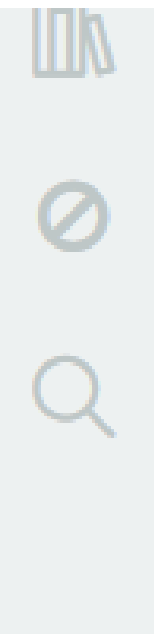
- สามารถเข้าถึงการทำงานขั้นพื้นฐานที่สุดของระบบได้
- code 1 บรรทัด \approx 1 clock (แล้วแต่ instruction)

Register (cons.)

- ยากไป! + ไม่ทันสมัย
- Hardware dependent -> หากเปลี่ยนสถาปัตยกรรมหรือแม้กระทั่งเปลี่ยน รุ่นของ MCU จะต้องศึกษาใหม่ทั้งหมด (อ่าน datasheet แค่ 1200 เอง สำหรับ MCU 1 รุ่น)
- ยากต่อการถ่ายทอด เพราะ OOP



Arduino (HW development platform)



```
~
6 void loop() {
7     // put your main code here, to run repeatedly:
8     pinMode(A5,OUTPUT);
9     digitalWrite(A5, HIGH);
10    delay(1000);
11    digitalWrite(A5, LOW);
12    delay(1000);
13 }
14
```

easy peasy!

HW development platform (pros.)

- ใช้ง่าย (function ถูกออกแบบให้ใช้งาน มี manual)
- มี Lib ให้ใช้เยอะมากๆๆๆๆๆๆๆๆๆๆๆๆๆๆๆๆ
- ใช้เวลาในการพัฒนาเร็วมากๆๆๆๆๆๆๆๆๆๆๆๆ

HW development platform (cons.)

- digitalWrite() จริงๆ แล้วข้างในมันคืออะไร? ใช้กี่ instruction? ใช้เวลาในการทำงานเท่าไร? ทำงานเสร็จแล้วภายในเปลี่ยนแปลงอย่างไร?
- ขึ้นอยู่กับ platform นั้นๆ
- ไม่สามารถเข้าถึง Hardware ทั้งหมดได้ (ใน platform ส่วนใหญ่)

STM Hardware abstraction layer (HAL)

- คือ APIs (application programming interfaces) ที่ทำหน้าที่ติดต่อยกหว่าง code และ hardware ระดับล่าง (พูดง่ายๆ คือ แปลง code เป็นคำสั่งไปควบคุม register) โดยอยู่ในรูปแบบของ function ที่เข้าใจได้ง่าย และไม่ขึ้นอยู่กับรุ่นของ MCU



HAL example

```
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, GPIO_PIN_SET);  
HAL_Delay(1000);  
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, GPIO_PIN_RESET);  
HAL_Delay(1000);
```

ไม่ขึ้นอยู่กับรุ่นของ MCU (ไม่ต้องไปอ่าน datasheet 1200 หน้า)

Course discription

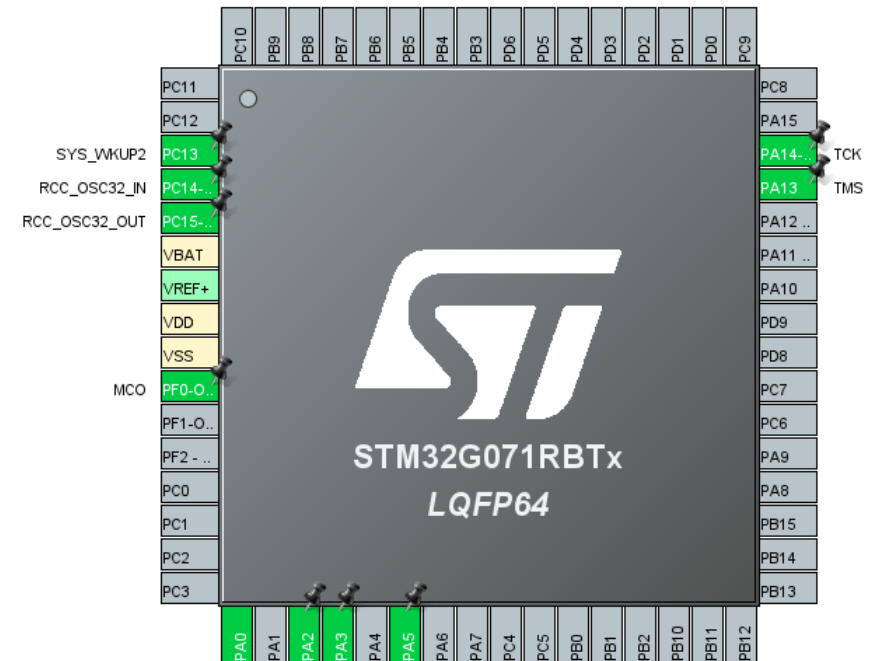
- เรียนรู้การใช้ HAL เพื่อใช้ในการควบคุม MCU
- เรียนรู้ peripheral
- เรียนรู้ flow ของ MCU

ทำไมต้อง STM32?

- ความชอบส่วนตัวล้วนนนนน
- มีรุ่นให้เลือกเยอะมาก ขึ้นอยู่กับงานที่ใช้
- ราคาถูกเมื่อเทียบกับ AVR และ Microchip (2022 so on)
- มี tool ที่ออกแบบมาดีและ ฟรี! (STM32cubeIDE)

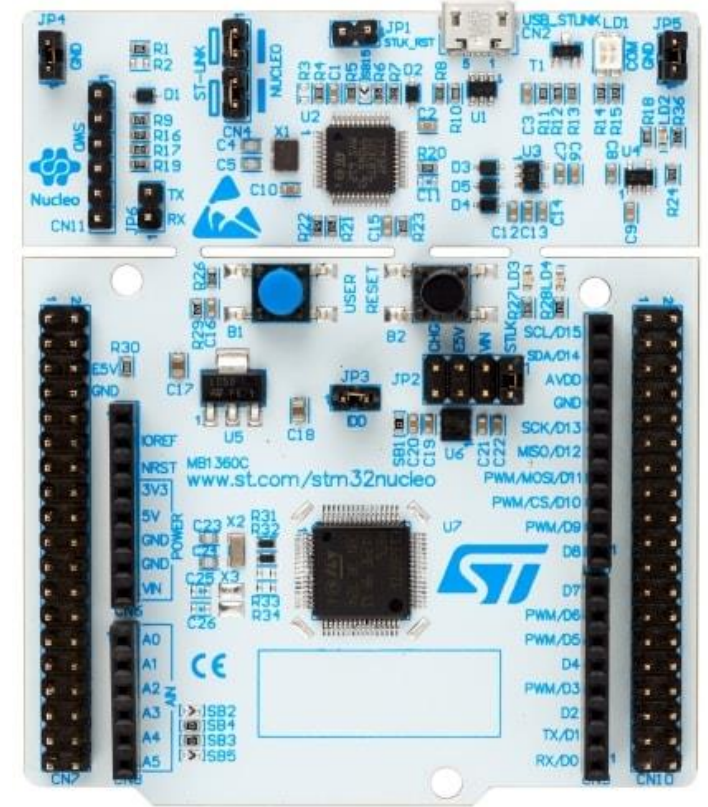
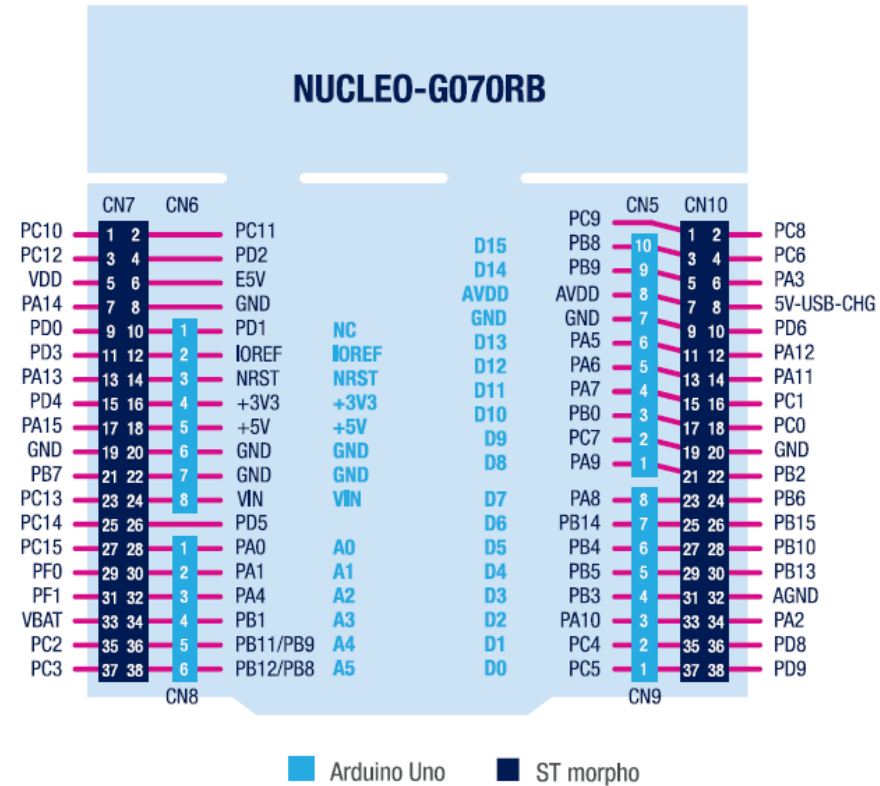
STM32

- 32bit ARM microcontroller
- From STMicroelectronics company
- Using STM32cubeIDE



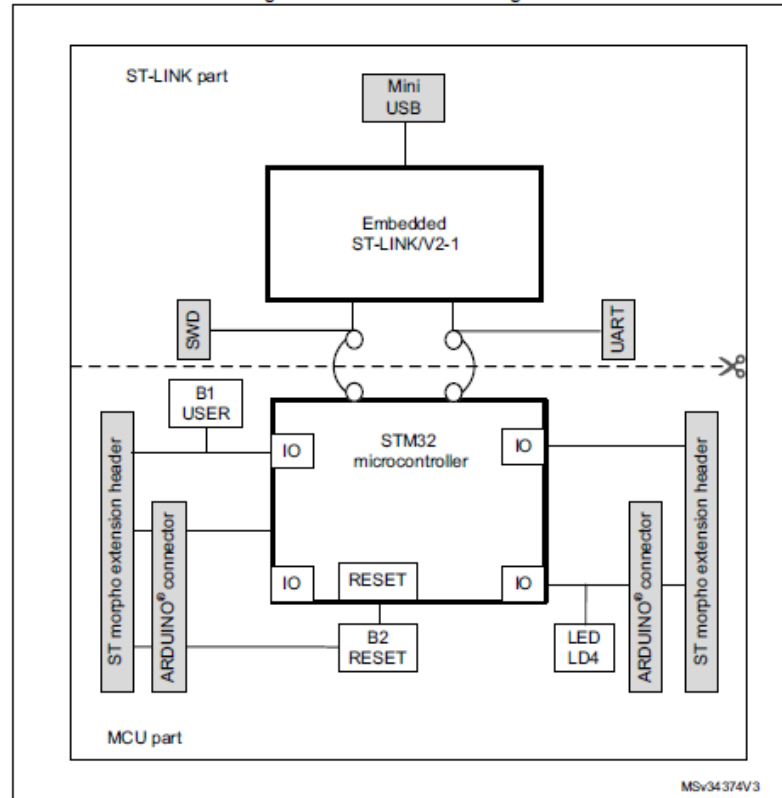
STM32 NUCLEO-G071RB

- ARM Cortex-M0+
- Single core 64 MHz
- 32 Kbytes RAM
- 14 timers
- I2C, USART
- I2S
- ADC , DAC

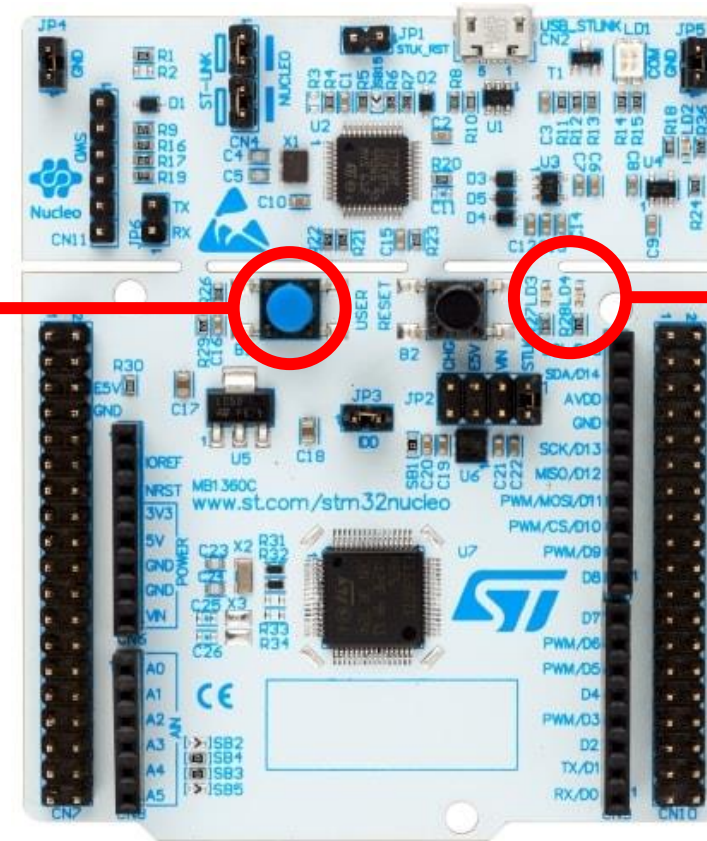


STM32 NUCLEO

Figure 2. Hardware block diagram



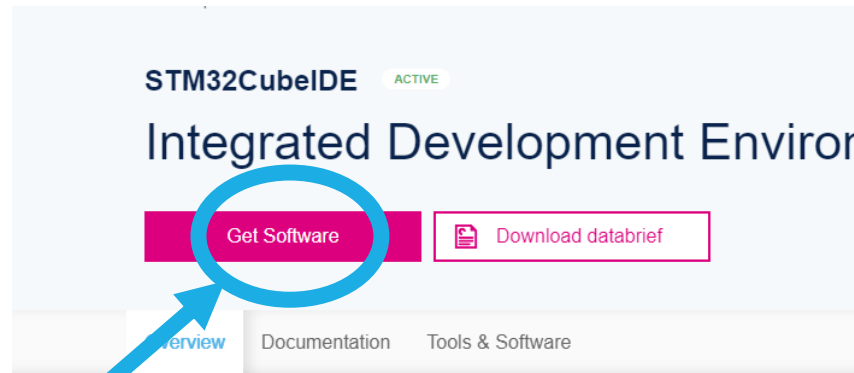
BUT [PC13]



LD4 [PA5]

Setup CUBE

Link : <https://www.st.com/en/development-tools/stm32cubeide.html>



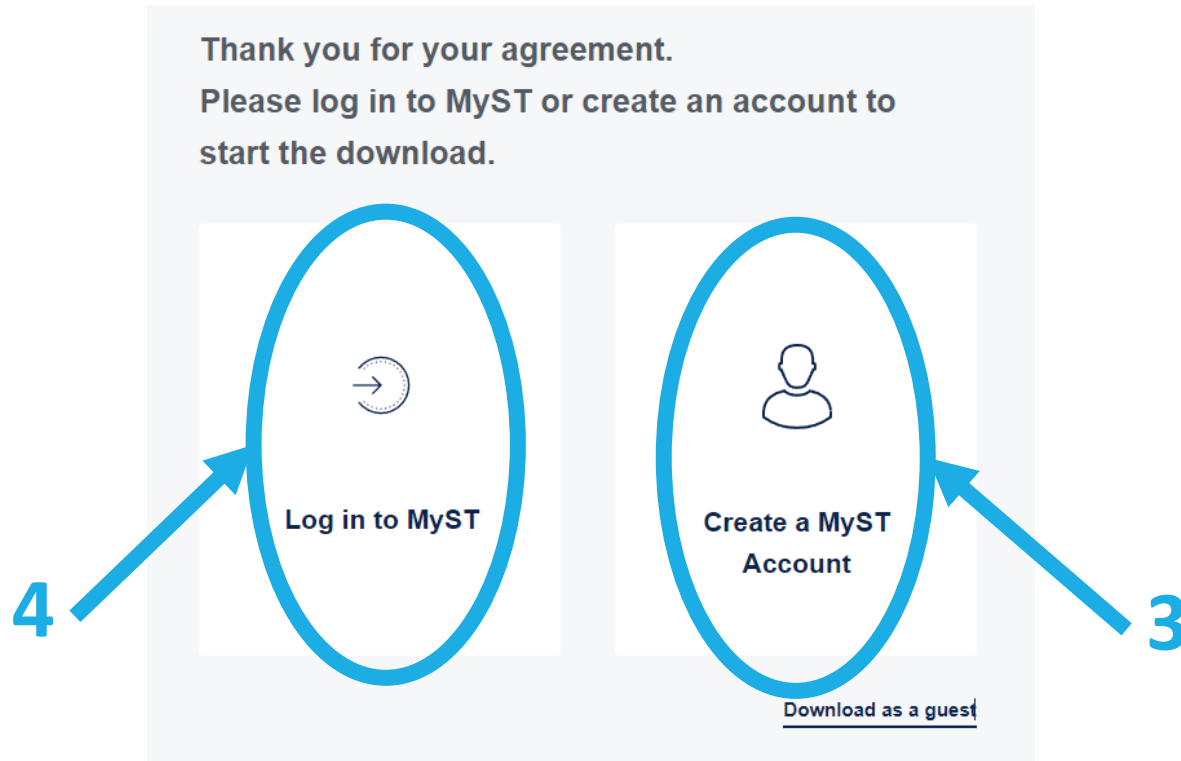
1 Product overview

*ให้ download STM32CubeIDE ไม่ใช่ STM32CubeMX

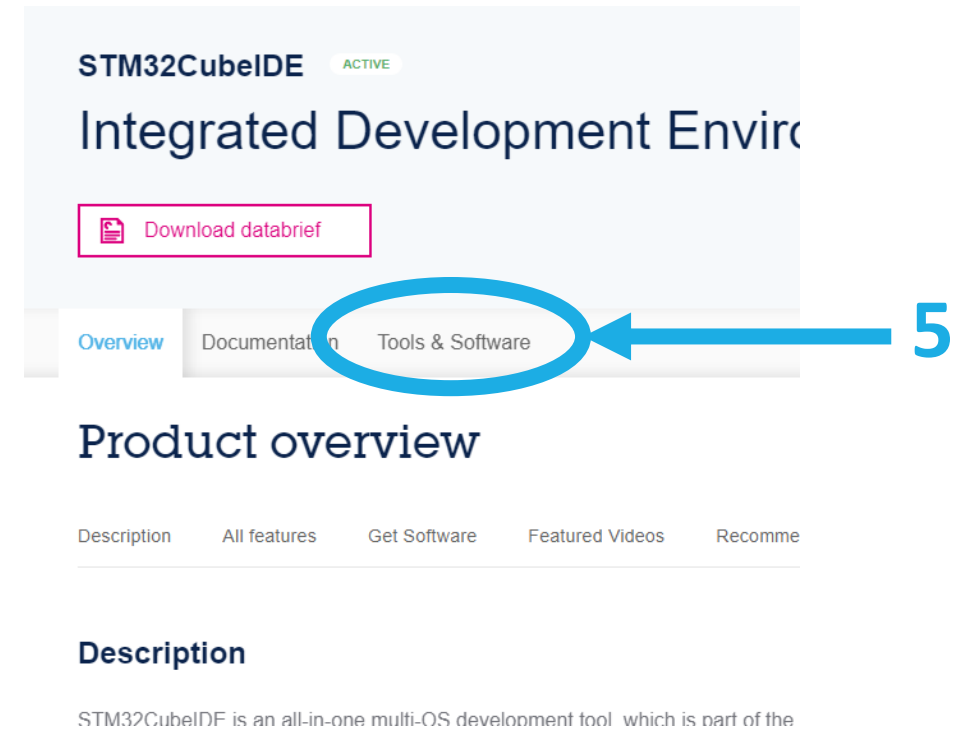
Part Number	General Description	Latest version	Download	All versions
+ STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.15.1	Get latest	Select version ▼
+ STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.15.1	Get latest	Select version ▼
+ STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	1.15.1	Get latest	Select version ▼
+ STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	1.15.1	Get latest	Select version ▼
+ STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.15.1	Get latest	Select version ▼

2 Depend on OS

Setup CUBE



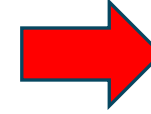
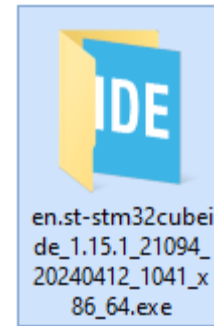
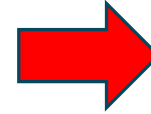
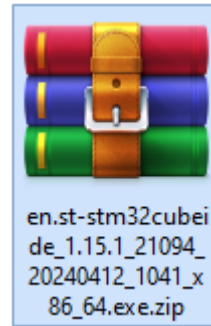
Create account and login



Click on Tools & Software

Setup CUBE

6. install program \approx 1GB

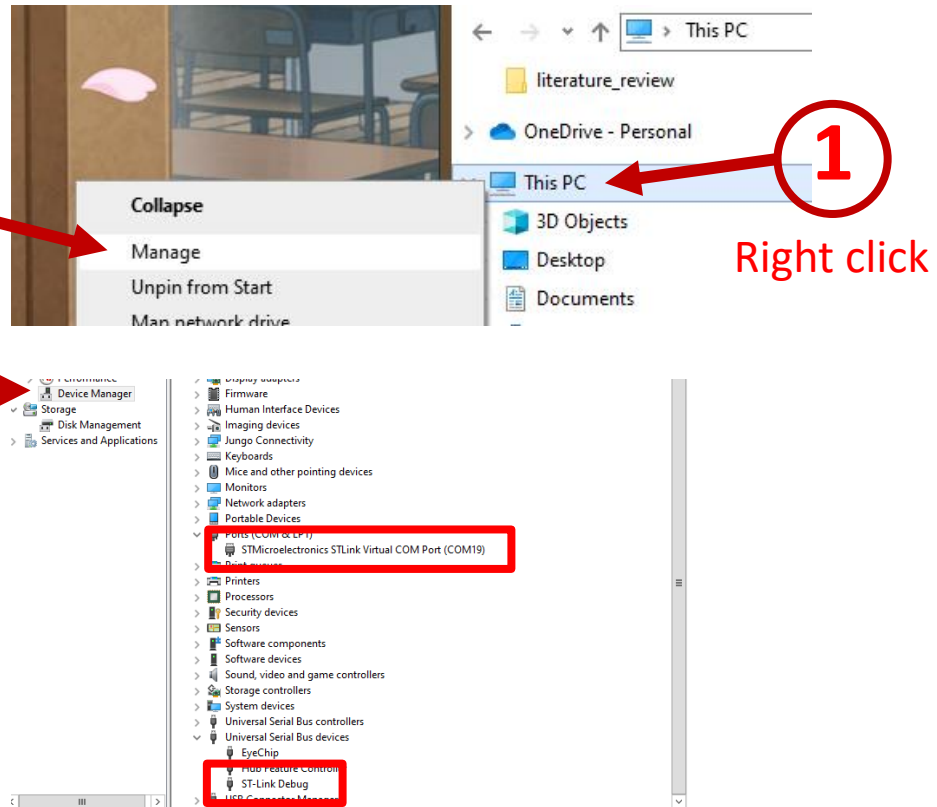


7. plugin board



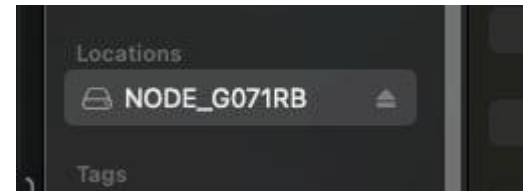
Check driver

Windows



MAC

```
pichaya@pichayas-Mac-mini:~$ ls /dev/tty.*  
/dev/tty.Bluetooth-Incoming-Port /dev/tty.usbmodem2303  
pichaya@pichayas-Mac-mini:~$
```

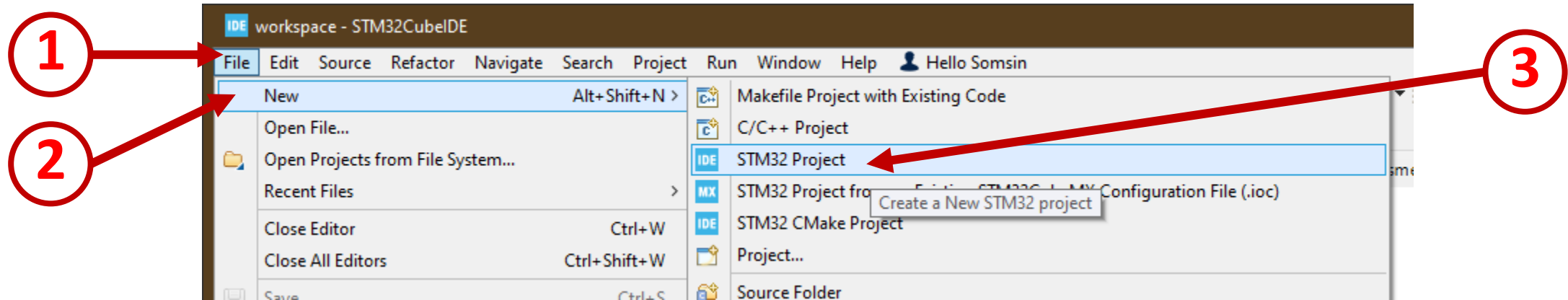


Let's start

Open STM32CubeIDE



initial CUBE



1. File
2. New
3. Select STM32 Project
4. download new firmware

Select MCU (STM32G071RBT6)

1. Click Board Selector
2. Type MCU model (STM32G071RBT6)
3. Select variant
4. Click next

MCU/MPU Selector Board Selector Example Selector Cross Selector

Commercial Part Number: NUCLEO-G071RB

PRODUCT INFO

- Type
- Supplier
- MCU / MPU Series
- Marketing Status
- Price

MEMORY

- Ext. Flash = 0 (MBit)
- Ext. EEPROM = 0 (kBytes)
- Ext. RAM = 0 (MBit)

FEATURES

Embedded Sensors

STM32G0 Series

The STM32 Nucleo-64 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the external SMPS significantly reduces power consumption in Run mode.

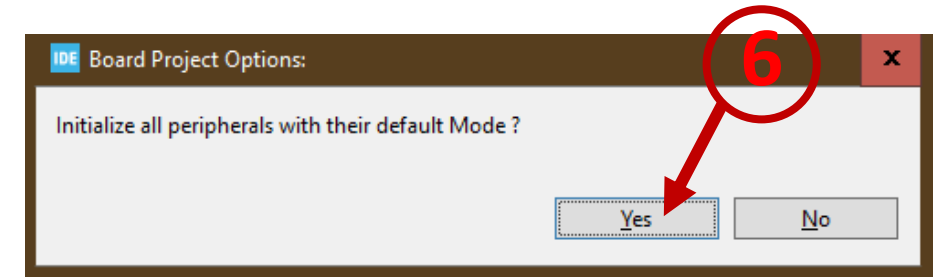
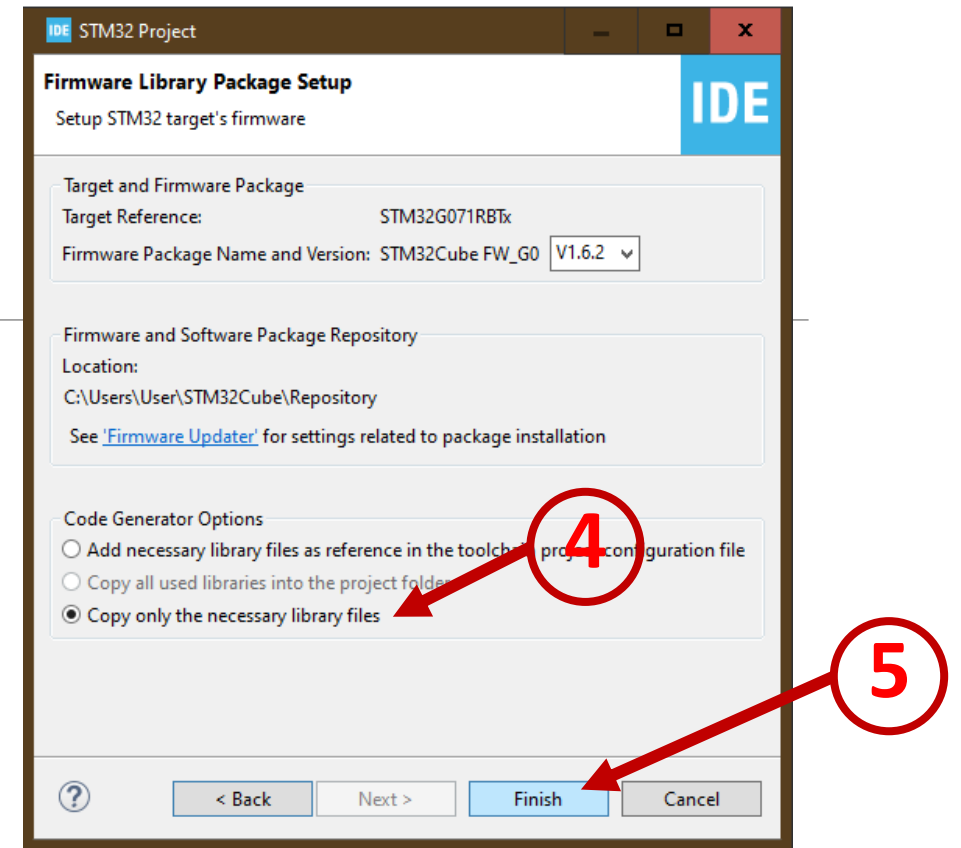
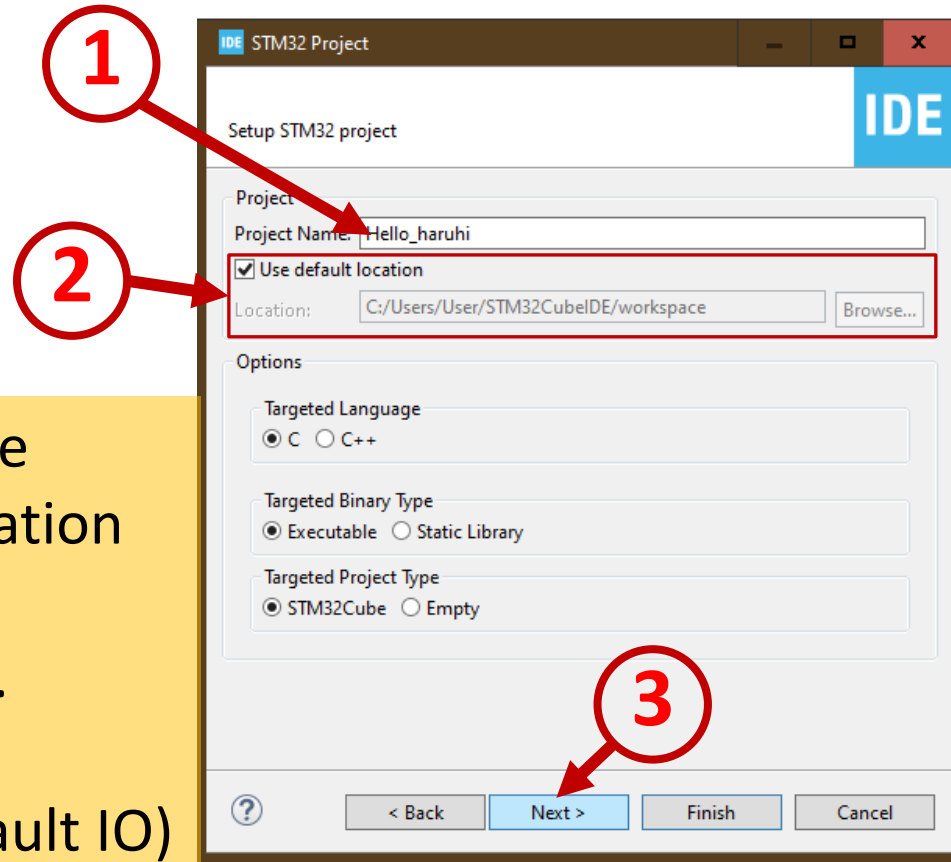
The ARDUINO® Uno V3 connectivity support and the ST morpho headers allow the easy expansion of the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields. The STM32 Nucleo-64 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.

Boards List: 1 item

	Overview	Commercial ...	Type	Marketing Sta...	Unit Price (US\$)	Mounted Device
☆		NUCLEO-G07...	Nucleo-64	Active	10.32	STM32G071RBT6

< Back Next > Finish Cancel

Select MCU



1. Type project name
2. Select project location
3. Click next
4. Select copy only...
5. Click finish
6. Click yes (init default IO)
7. wait

Pinout & Configuration

Clock Configuration

Project Manager

Tools

Software Packs

Pinout

Q [v]



Categories A->Z

System Core >

Analog >

Timers >

Connectivity v

I2C1
I2C2
IRTIM
LPUART1
SPI1
SPI2
UCPD1
UCPD2
USART1
✓ USART2
USART3
USART4

Multimedia >

Computing >

Middleware and Software Packs >

Utilities >

Hardware selector

USART2 Mode and Configuration

Mode

Mode Asynchronous v

Hardware Flow Control (RS232) Disable v

☐ Hardware Flow Control (RS485)

Slave Select(NSS) Management Disable v

Configuration

Reset Configuration

✓ DMA Settings

✓ GPIO Settings

✓ Parameter Settings

✓ User Constants

✓ NVIC Settings

Configure the below parameters :

Q Search (Ctrl+F)

Basic Parameters

Baud Rate 115200 Bits/s

Word Length 8 Bits (including Parity)

Parity None

Stop Bits 1

Advanced Parameters

Data Direction Receive and Transmit

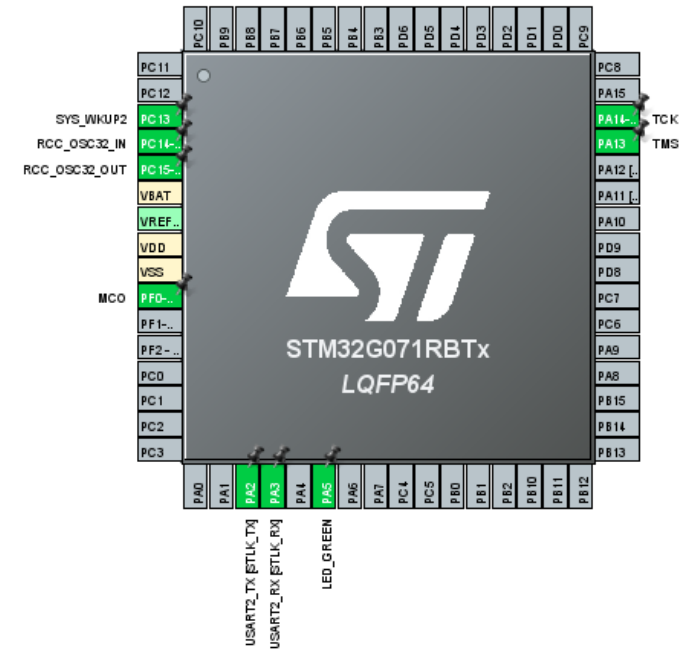
Over Sampling 16 Samples

Hardware properties

Pinout view

System view

Pin config



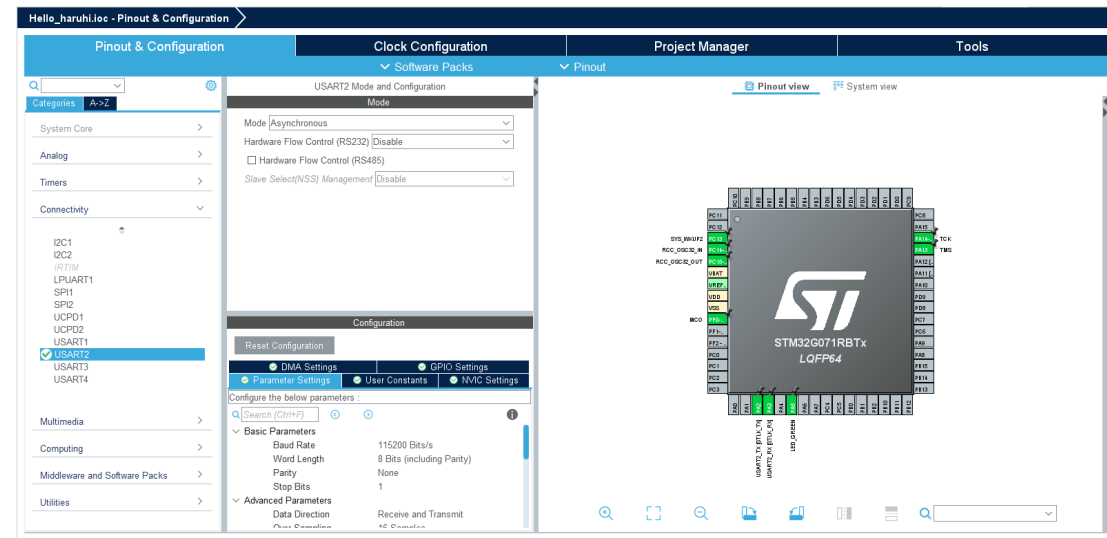
Q [v]

detail

Hardware selector : เลือก hardware ที่จะ config เช่น ADC , I2C, Serial

Hardware properties : config hardware ที่เลือก

Pin config : เลือก pin IO ที่จะใช้



Example configuration

1. Pinout & config
2. Click Connectivity
3. Click USART2
4. Select Asynchronous Mode
5. Set Baud Rate to 38400

The screenshot displays the STM32CubeMX Pinout & Configuration window. The interface is divided into several sections:

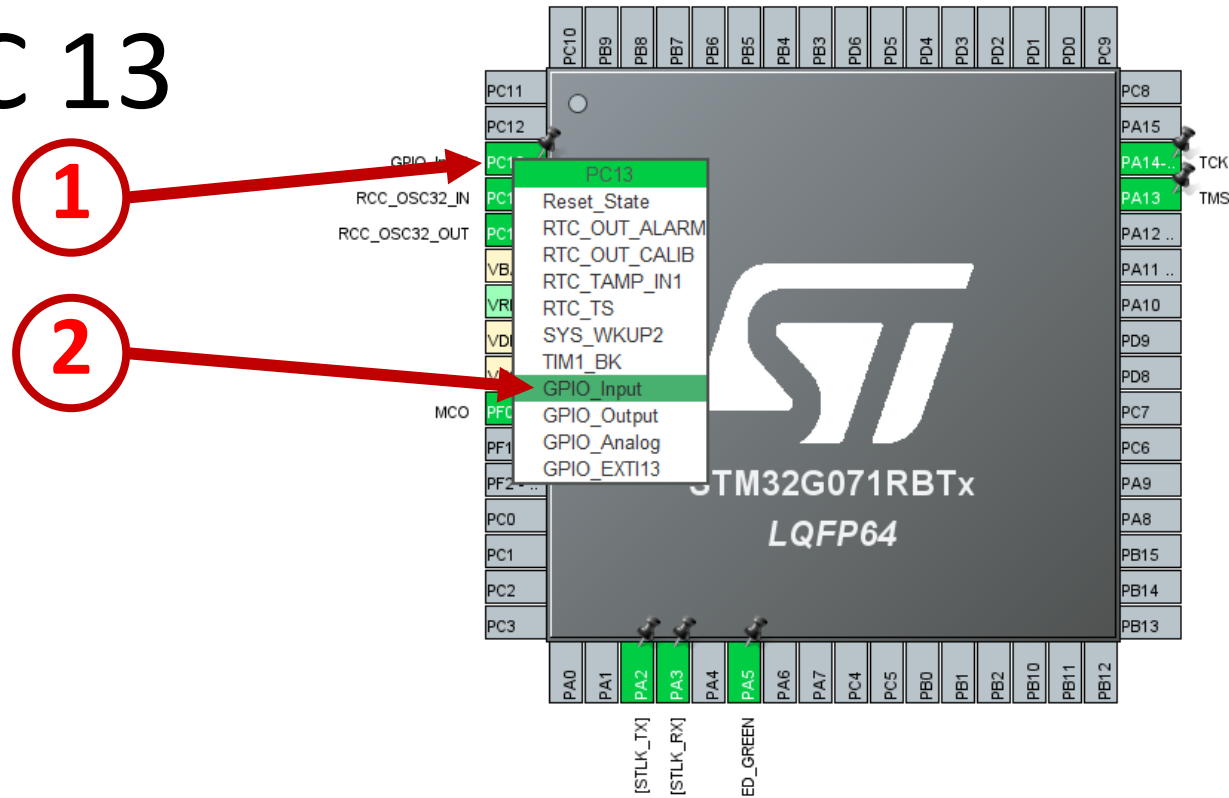
- Pinout & Configuration** (Top Tab): This section is active, showing the configuration for the selected peripheral.
- Categories** (Left Panel): A list of system components. **Connectivity** is selected, and **USART2** is highlighted in the list.
- USART2 Mode and Configuration** (Main Panel): This panel shows the configuration for USART2. The **Mode** is set to **Asynchronous**. The **Hardware Flow Control (RS232)** is set to **Disable**. The **Slave Select(NSS) Management** is set to **Disable**.
- Configuration** (Bottom Panel): This panel shows the configuration for the selected peripheral. The **Parameter Settings** tab is active, showing the **Baud Rate** set to **38400 Bits/s**.
- Pinout view** (Right Panel): A diagram of the STM32G071RBTx LQFP64 package showing the pinout. The pins are labeled with their names and numbers.

Red circles with numbers 1 through 5 are overlaid on the image, indicating the steps in the configuration process:

1. Pinout & config
2. Click Connectivity
3. Click USART2
4. Select Asynchronous Mode
5. Set Baud Rate to 38400

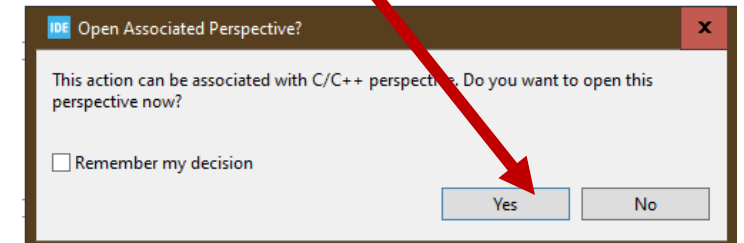
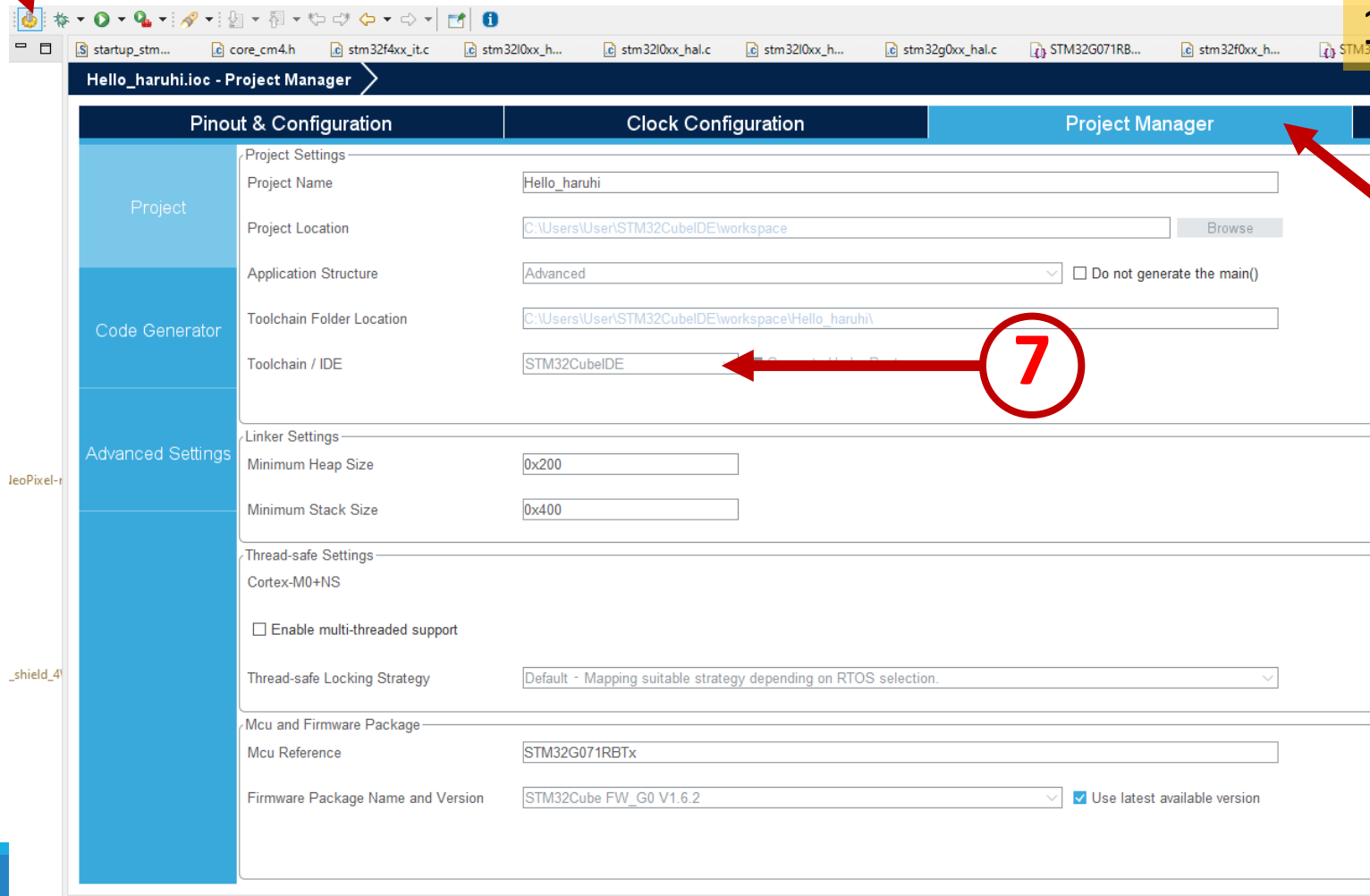
Set input button

PC 13



8 Example configuration

6. Select Project Manager
7. Select STM32CubeIDE
8. Click Generate Code
9. Click Yes (change perspective)
10. Click Yes (Generate code)



Coding in Cube System

```
19 / * Includes -----
20 #include "main.h"
21
22 /* Private includes -----
23 /* USER CODE BEGIN Includes */
24 |
25 /* USER CODE END Includes */
26
27 /* Private typedef -----
28 /* USER CODE BEGIN PTD */
29
30 /* USER CODE END PTD */
31
32 /* Private define -----
33 /* USER CODE BEGIN PD */
34
35 /* USER CODE END PD */
36
37 /* Private macro -----
38 /* USER CODE BEGIN PM */
39
40 /* USER CODE END PM */
41
42 /* Private variables -----
43 TIM_HandleTypeDef htim2;
44
45 UART_HandleTypeDef huart2;
46
47 /* USER CODE BEGIN PV */
48
49 /* USER CODE END PV */
50
```

- เมื่อ Generate code ขึ้นมาแล้ว Cube จะสร้าง section สำหรับให้ผู้ใช้เขียนโปรแกรม ซึ่ง User จะต้องเขียน code ให้อยู่ในระหว่าง BEGIN กับ END เท่านั้น

- หากเขียน code นอก section ดังกล่าว เมื่อ generate code อีกรอบ code ดังกล่าว จะถูกลบ

Code section

- USER CODE Header -> comment and license
- USER CODE BEGIN Includes -> include library
- USER CODE BEGIN PTD -> Typedef, struct
- USER CODE BEGIN PD -> Define
- USER CODE BEGIN PM -> Macro , Inline function
- USER CODE BEGIN PV -> Variables
- USER CODE BEGIN PFP -> Function prototypes, Function declaration
- etc.

```
20 #include "main.h"
21
22 /* Private includes -----
23 /* USER CODE BEGIN Includes */
24
25 /* USER CODE END Includes */
26
27 /* Private typedef -----
28 /* USER CODE BEGIN PTD */
29
30 /* USER CODE END PTD */
31
32 /* Private define -----
33 /* USER CODE BEGIN PD */
34
35 /* USER CODE END PD */
36
37 /* Private macro -----
38 /* USER CODE BEGIN PM */
39
40 /* USER CODE END PM */
41
42 /* Private variables -----
43 TIM_HandleTypeDef htim2;
44
45 UART_HandleTypeDef huart2;
46
47 /* USER CODE BEGIN PV */
48
49 /* USER CODE END PV */
50
```

example

```
36 /* USER CODE BEGIN 2 */
37 HAL_TIM_Base_Start_IT(&htim2);
38
39 /* USER CODE END 2 */
40
41 /* Infinite loop */
42 /* USER CODE BEGIN WHILE */
43 while (1)
44 {
45     /* USER CODE END WHILE */
46
47     /* USER CODE BEGIN 3 */
48     //                0b33222222222211111111110000000000
49     //                0b10987654321098765432109876543210
50     GPIOA->MODER = 0b0000000000000000000000001000000000;
51     GPIOA->ODR   = 0b00000000000000000000000000000000;
52     HAL_Delay(1000);
53     GPIOA->ODR   = 0b00000000000000000000000000000000;
54     HAL_Delay(1000);
55
56     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, GPIO_PIN_SET);
57     HAL_Delay(1000);
58     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, GPIO_PIN_RESET);
59     HAL_Delay(1000);
60 }
61 /* USER CODE END 3 */
62 }
63 }
```

Right

```
97 HAL_TIM_Base_Start_IT(&htim2);
98
99 /* USER CODE END 2 */
100
101 /* Infinite loop */
102 /* USER CODE BEGIN WHILE */
103 char[20] str = "Haruhi";
104 while (1)
105 {
106     for(int i=0;i<5;i++){
107         sprintf(str,"haruhi = %s", "haruhi");
108         HAL_UART_Transmit_IT(&huart2, str, 18);
109     }
110     /* USER CODE END WHILE */
111
112     /* USER CODE BEGIN 3 */
113 }
114 /* USER CODE END 3 */
115
116 }
117 /**
118 * @brief System Clock Configuration
119 */
```

Wrong

Basic HAL Function

- HAL_Delay(Delay)
- หยุดการทำงานของ MCU เป็นเวลาเท่ากับ [Delay] หน่วยเป็น ms

เช่น HAL_Delay(2000); คือการหน่วงเวลา 2 วินาที

Basic HAL Function

- HAL_GPIO_WritePin(GPIOx, GPIO_Pin, PinState)
- สั่งเปลี่ยนสถานะของ PIN
- GPIOx คือ port เช่น GPIOA หรือ GPIOB
- GPIO_Pin คือ pin ที่ต้องการสั่งการ,
- PinState คือ สถานะที่ต้องการเปลี่ยน 1 คือ HIGH state 0 คือ LOW state
- เช่น HAL_GPIO_WritePin(GPIOC, 12, 1); คือการสั่งให้ PC12 มีสถานะ 3.3V

Basic HAL Function

- HAL_GPIO_ReadPin(GPIOx, GPIO_Pin)
- อ่านสถานะของ PIN
- GPIOx คือ port เช่น GPIOA หรือ GPIOB
- GPIO_Pin คือ pin ที่ต้องการอ่าน
- ค่าที่ได้ออกมาคือ 0 (low state) หรือ 1 (high state)

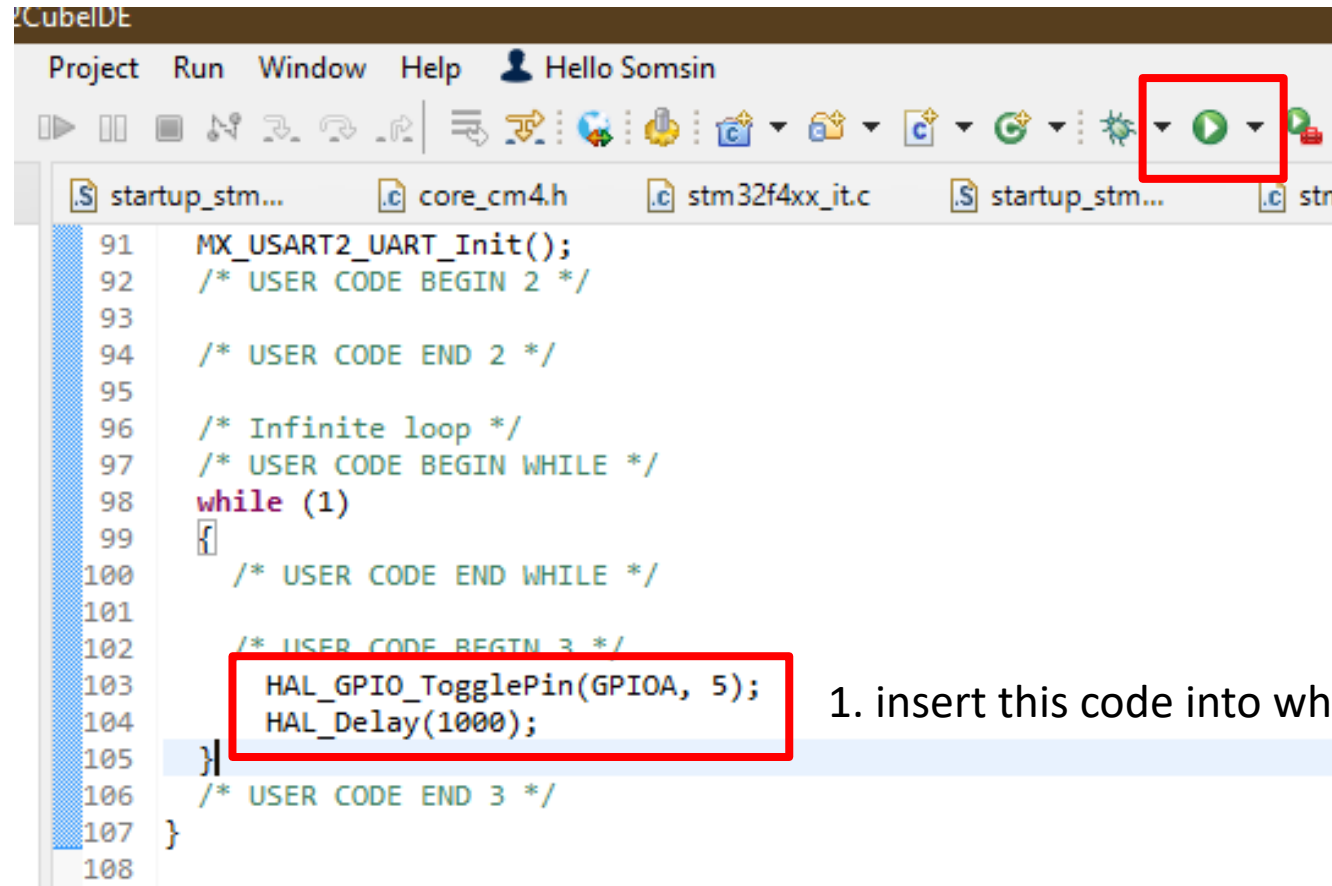
เช่น HAL_GPIO_ReadPin(GPIOC, 13); คือการอ่านค่า PIN C13

Basic HAL Function

- HAL_GPIO_TogglePin(GPIOx, GPIO_Pin)
- สลับสถานะของ PIN เป็นตรงกันข้าม (High เป็น Low , Low เป็น High)
- GPIOx คือ port เช่น GPIOA หรือ GPIOB
- GPIO_Pin คือ pin ที่ต้องการเปลี่ยน

เช่น HAL_GPIO_TogglePin(GPIOA, 5); คือการสลับค่าของ PIN A5

Try this



The screenshot shows the STM32CubeIDE interface. The top menu bar includes Project, Run, Window, and Help. The toolbar contains various icons, with the Run button (a green play icon) highlighted by a red box. The main editor window displays a C code file with the following content:

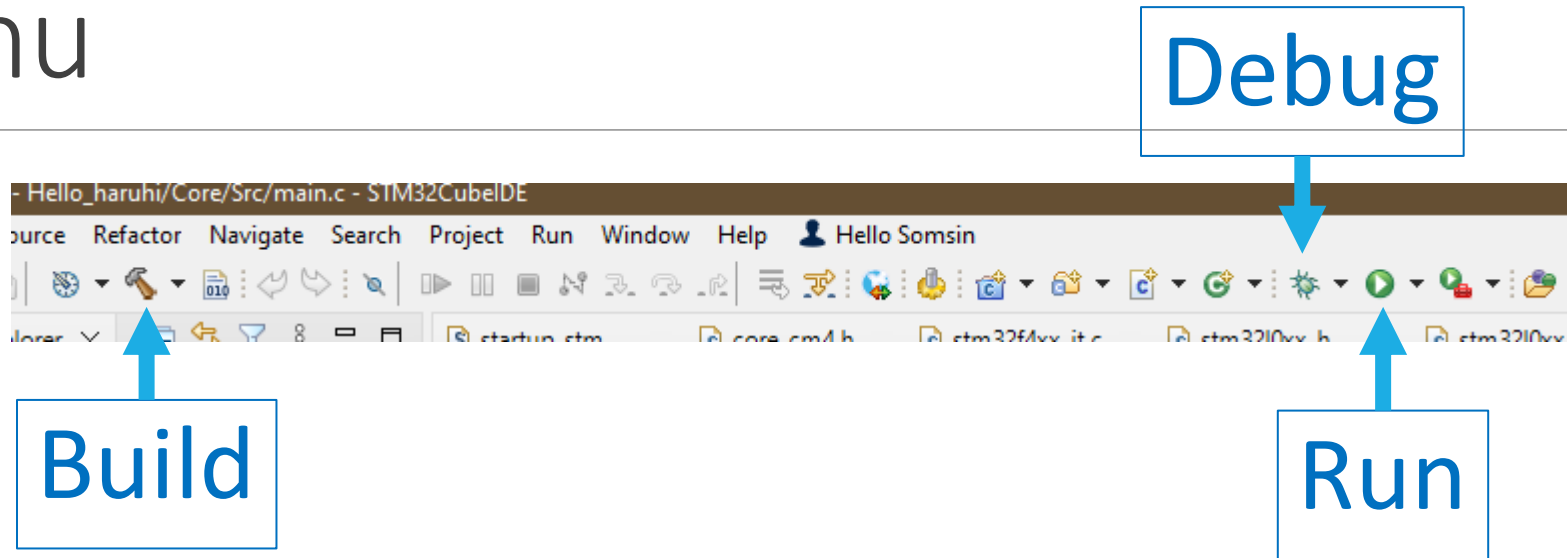
```
91  MX_USART2_UART_Init();
92  /* USER CODE BEGIN 2 */
93
94  /* USER CODE END 2 */
95
96  /* Infinite loop */
97  /* USER CODE BEGIN WHILE */
98  while (1)
99  {
100     /* USER CODE END WHILE */
101
102     /* USER CODE BEGIN 3 */
103     HAL_GPIO_TogglePin(GPIOA, 5);
104     HAL_Delay(1000);
105 }
106 /* USER CODE END 3 */
107 }
108
```

The code between lines 103 and 104 is highlighted by a red box, indicating it should be inserted into the while loop.

2. Click RUN

1. insert this code into while section

Menu

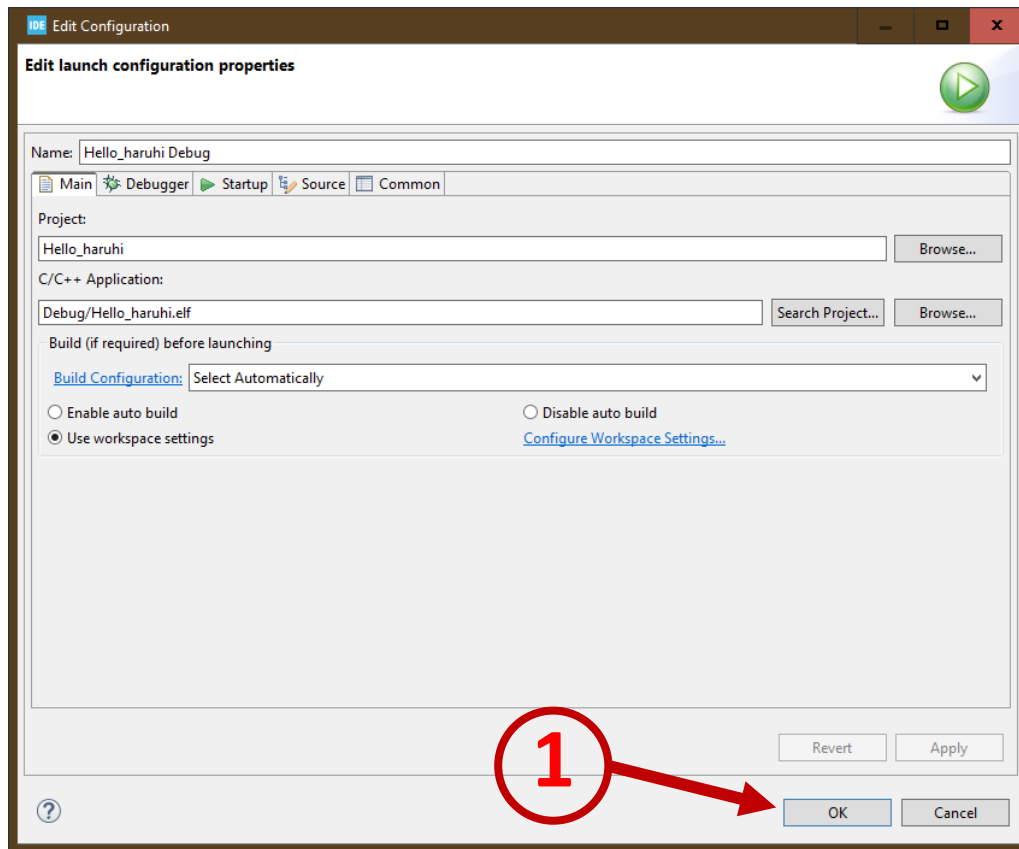


Build : Compile Code and report result (Error)

Run : compile code to binary file and upload to board

Debug : Run with Debug

Upload Code



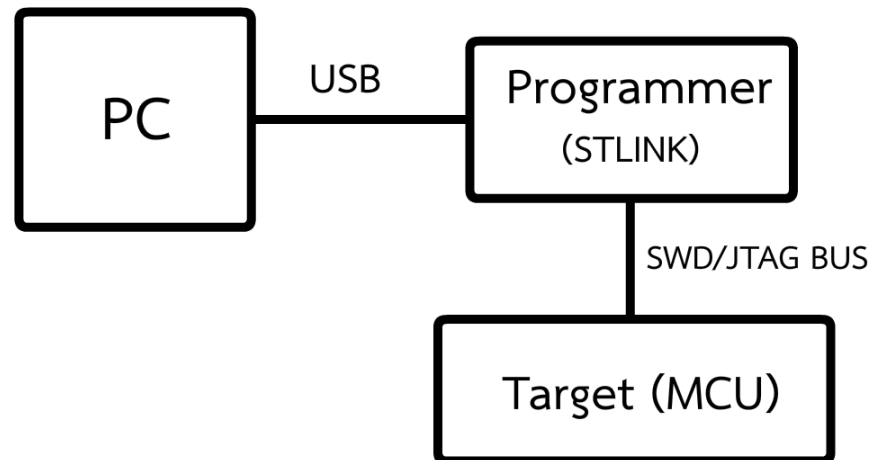
```
terminated: Hello_haruhi Debug [SIMS2 C/C++ A  
Verifying ...  
  
Download verified successfully  
  
Shutting down...  
Exit.
```

Programmer

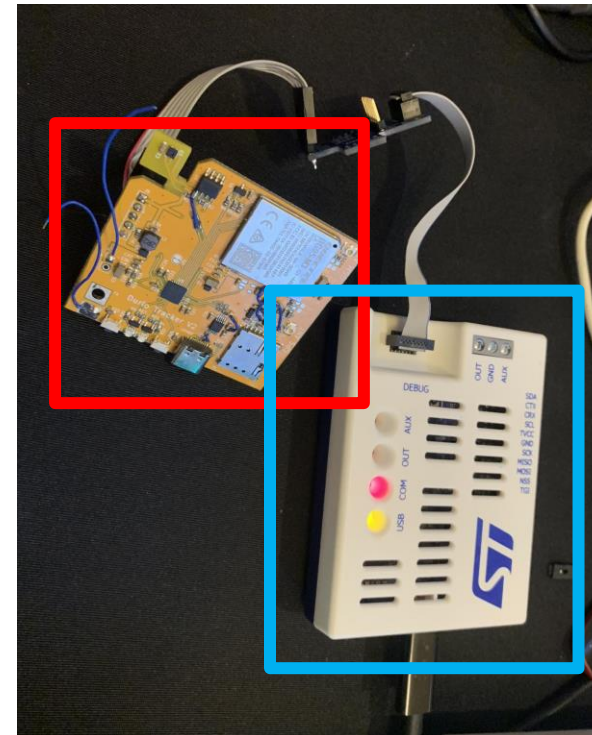
- A hardware device accompanied with software which is used to transfer the machine language code to the microcontroller/EEPROM from the PC
- อุปกรณ์ที่สามารถนำ code (compiled) ไปไว้บน MCU ได้



STLINK (programmer)



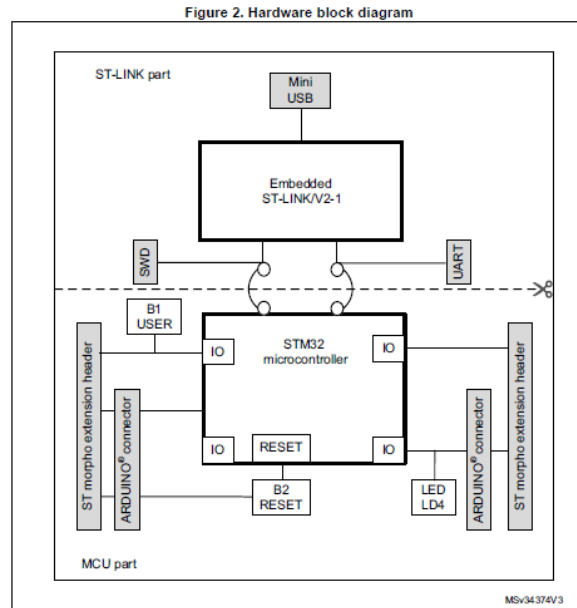
Target



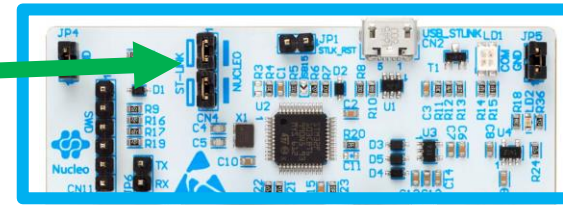
Programmer

STM32NUCLEO STLINK

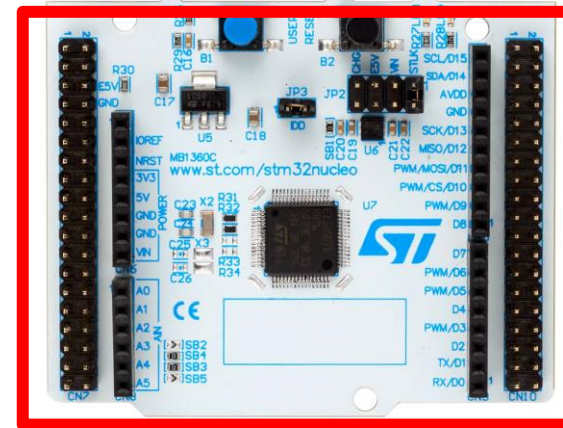
- บนบอร์ด STM32NUCLEO จะมี STLINK ฟังก์ชันมาอยู่แล้ว (ขายมาพร้อมกัน - -)
- เชื่อมต่อ CN4 เพื่อใช้งาน หรือถอดออกเพื่อ นำ STLINK ไปใช้ที่อื่น / ใช้ programmer ตัวอื่นเพื่อ program target



CN4



Programmer



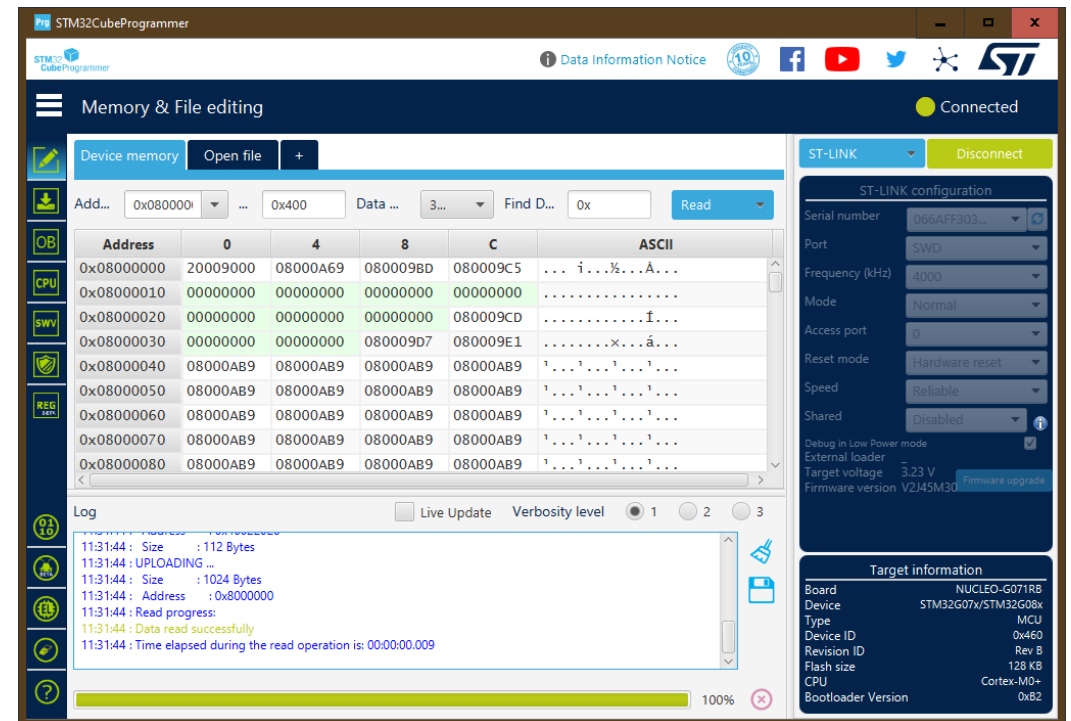
Target

STM32CubeProgrammer

<https://www.st.com/en/development-tools/stm32cubeprog.html>

- ใช้เพื่อดูอ่านสถานะของ STLINK และ TARGET
- สามารถ read/write/erase ข้อมูลบน target ได้

* ใครไม่สามารถ upload code ได้ให้ download โปรแกรมมาเพื่อเช็คดู แต่ใครที่ upload ได้ปกติสามารถข้ามไปได้เลย



Quiz

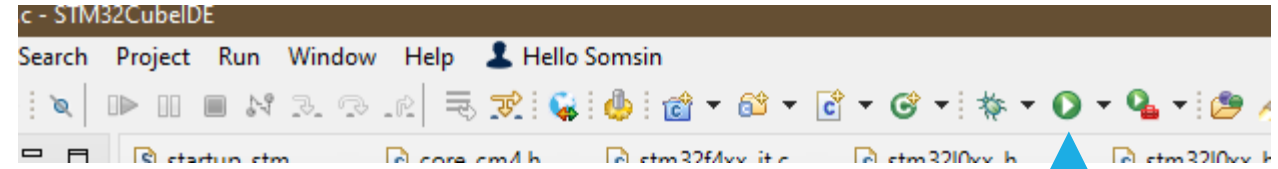
IO toggle

Serial UART

HAL_StatusTypeDef HAL_UART_Transmit(*huart, *pData, Size, Timeout)

- huart คือ channel ของ serial เช่น huart1, huart2
- pData คือ ข้อมูลที่ต้องการส่งออก ในรูปของ array of char
- Size คือ จำนวนข้อมูลที่ต้องการส่งออก (มีหน่วยเป็น byte)
- Timeout คือ เวลาจำกัดในการรอการ clear buffer
- เช่น HAL_UART_Transmit(&huart2, "Haruhi\r\n", 8, 1000);

Serial UART



Run

```
96  /* Infinite loop */
97  /* USER CODE BEGIN WHILE */
98  while (1)
99  {
100     /* USER CODE END WHILE */
101
102     /* USER CODE BEGIN 3 */
103     char str[50];
104     char str_cnt;
105     char pin_state = 0;
106
107     pin_state = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
108
109     str_cnt = sprintf(str, "haruhi = %d\r\n", pin_state);
110     HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, pin_state);
111     HAL_UART_Transmit(&huart2, str, str_cnt, 1000);
112     HAL_Delay(1000);
113 }
114 /* USER CODE END 3 */
115 }
```

Serial UART

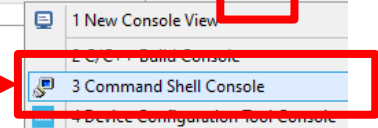
```
98 while (1)
99 {
100     /* USER CODE END WHILE */
101
102     /* USER CODE BEGIN 3 */
103     char str[50];
104     char str_cnt;
105     str_cnt = sprintf(str, "haruhi = %d\r\n", HAL_GPIO_ReadPin(GPIOC, 13));
106     HAL_UART_Transmit(&huart2, str, str_cnt, 1000);
107     HAL_Delay(1000);
108 }
109 /* USER CODE END 3 */
110 }
111
112 /**
113  * @brief System Clock Configuration
114  * @retval None
115  */
116 void SystemClock_Config(void)
117 {
118     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
119     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
120     RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
121
122     /** Initializes the RCC Oscillators according to the specified parameters
123      * in the RCC_OscInitTypeDef structure.
124      */
125     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
126     RCC_OscInitStruct.HSISState = RCC_HSI_ON;
```

1. New console

```
Console X
CDT Build Console [somsin_MCU_class_template_CUBE]
01:58:16 **** Build of configuration Debug for project somsin_MCU_class_template_CUBE ****
make -j8 all
arm-none-eabi-size  somsin_MCU_class_template_CUBE.elf
text  data  bss  dec  hex filename
16256  12  1796  18064  4690 somsin_MCU_class_template_CUBE.elf
Finished building: default.size.stdout

01:58:17 Build Finished. 0 errors, 0 warnings. (took 273ms)
```

2. Command shell console

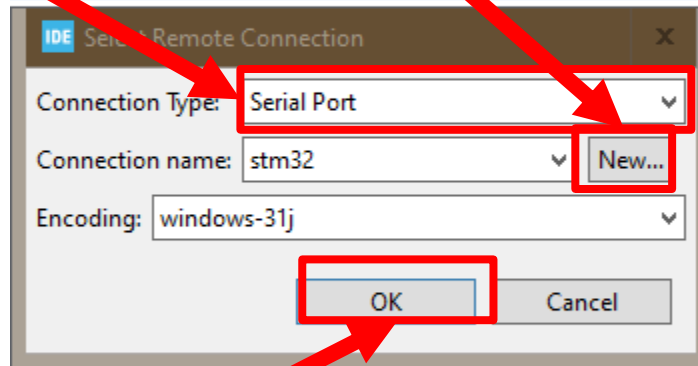


Serial UART

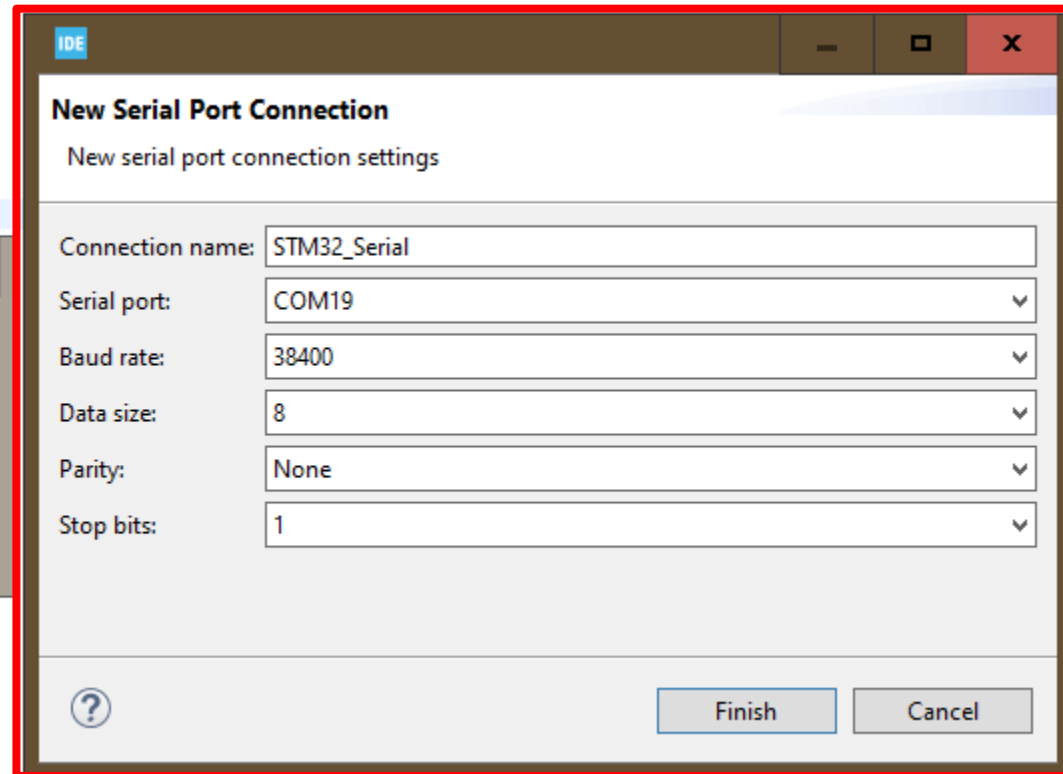
5. Set parameter and click finish

3. Select Serial Port

4. Click New...



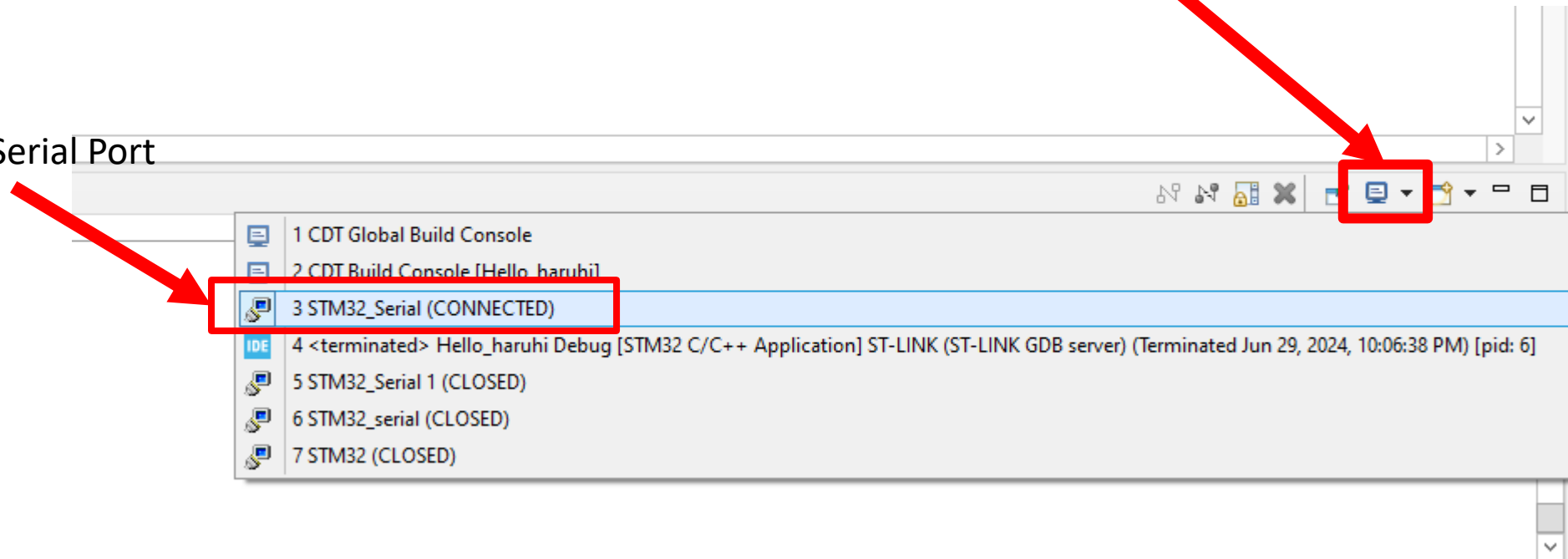
6. Click OK



Reopen console

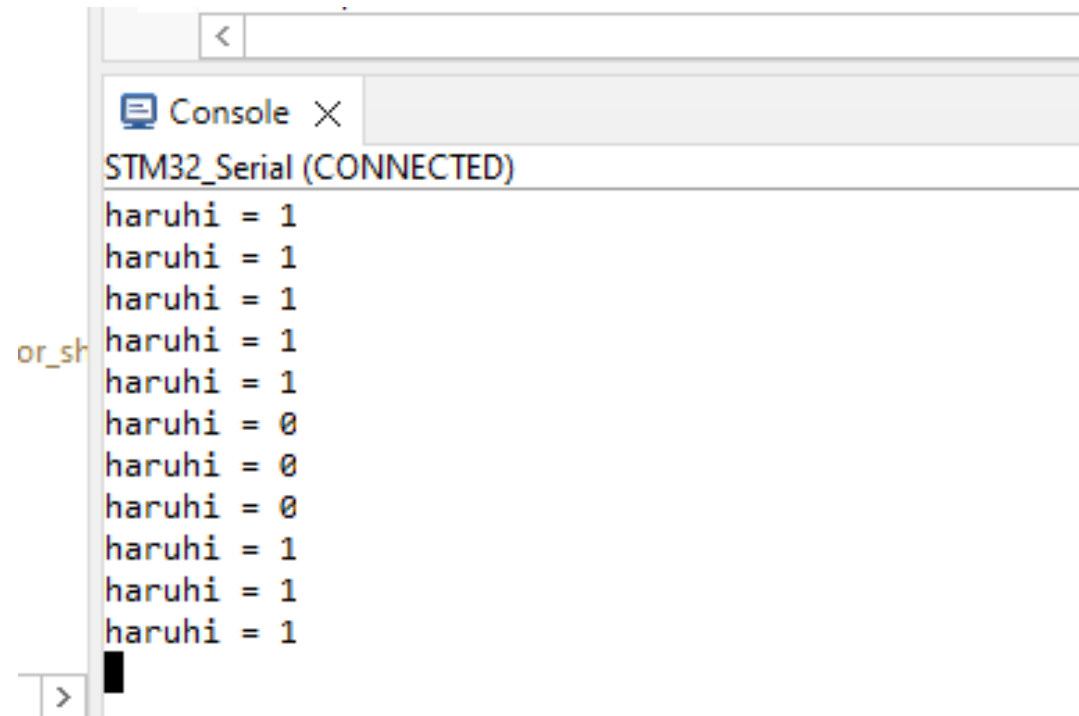
1. Display console

2. Select Serial Port



Result

Push blue button



```
or_sh haruhi = 1
haruhi = 1
haruhi = 1
haruhi = 1
haruhi = 1
haruhi = 0
haruhi = 0
haruhi = 0
haruhi = 1
haruhi = 1
haruhi = 1
```

LAB

1. setup STM32CebelIDE
2. initial STM32 CUBE board (STM32 NUCLEO-G071RB)
3. Sent out Student Number and name via Serial port

