Tutorial : interfacing an I2C LCD Display with STM32L4

The HD44780 is a widespread common LCD controller allowing to drive various formats of LCD displays made of 1 line with 8 characters (1X8), 2X16, 2X20, 4x20 or others.

It can be driven through a PCF8574 device which provides general-purpose remote I/O expansion by way of the I2C interface [serial clock (SCL), serial data (SDA)].

The following equipment is used

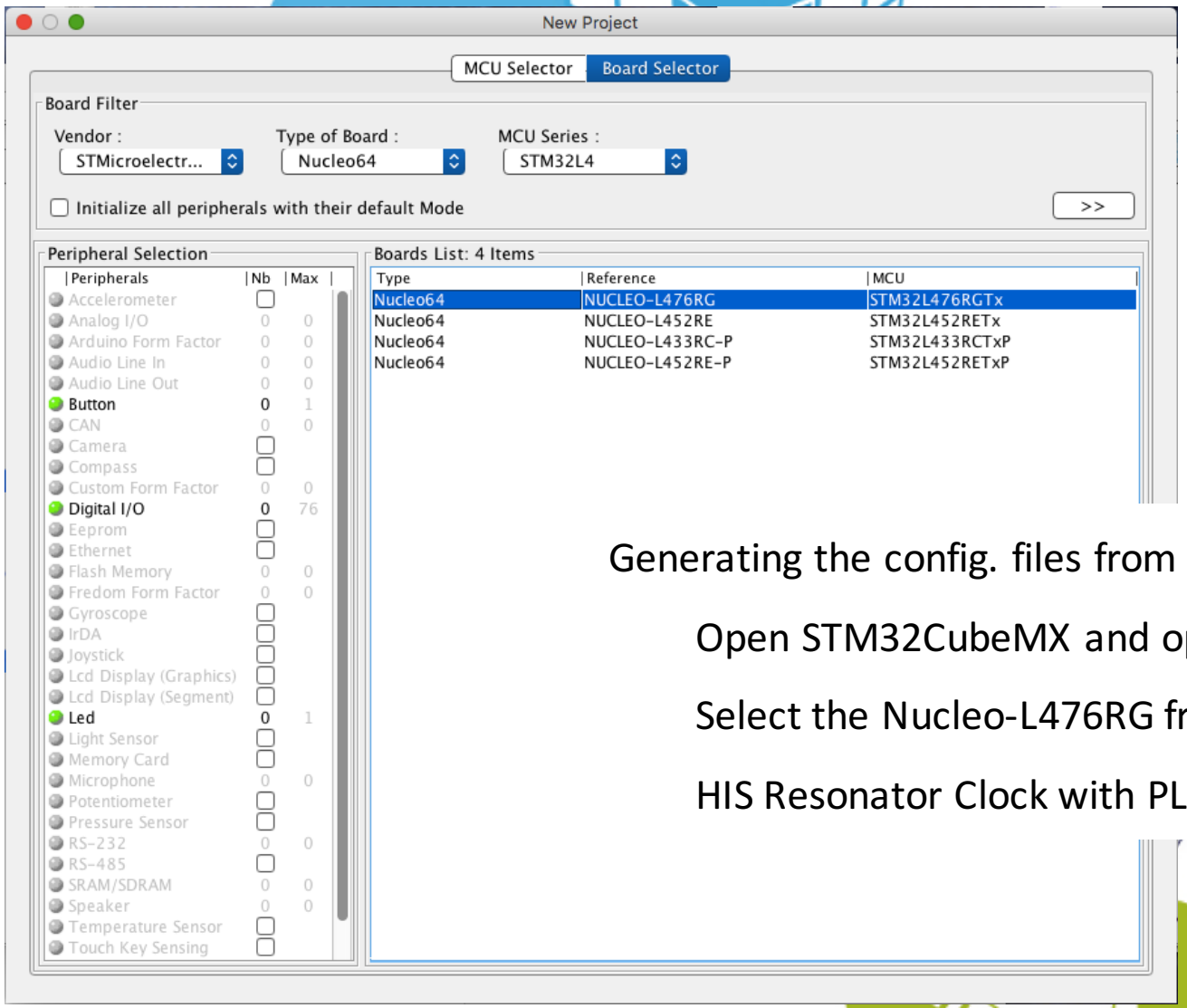
- NUCLEO-L476RG Board or equivalent and a HD44780 LCD with a PCF8574 expander
- Eclipse with the necessary packages for Nucleo boards installed
- OpenOCD or STLink USB Driver - STM32CubeMX

Jean-Christophe Toussaint Phelma Grenoble-INP France

New Project

Load Project

Help



Generating the config. files from STM32CubeMX.

Open STM32CubeMX and open a new project.

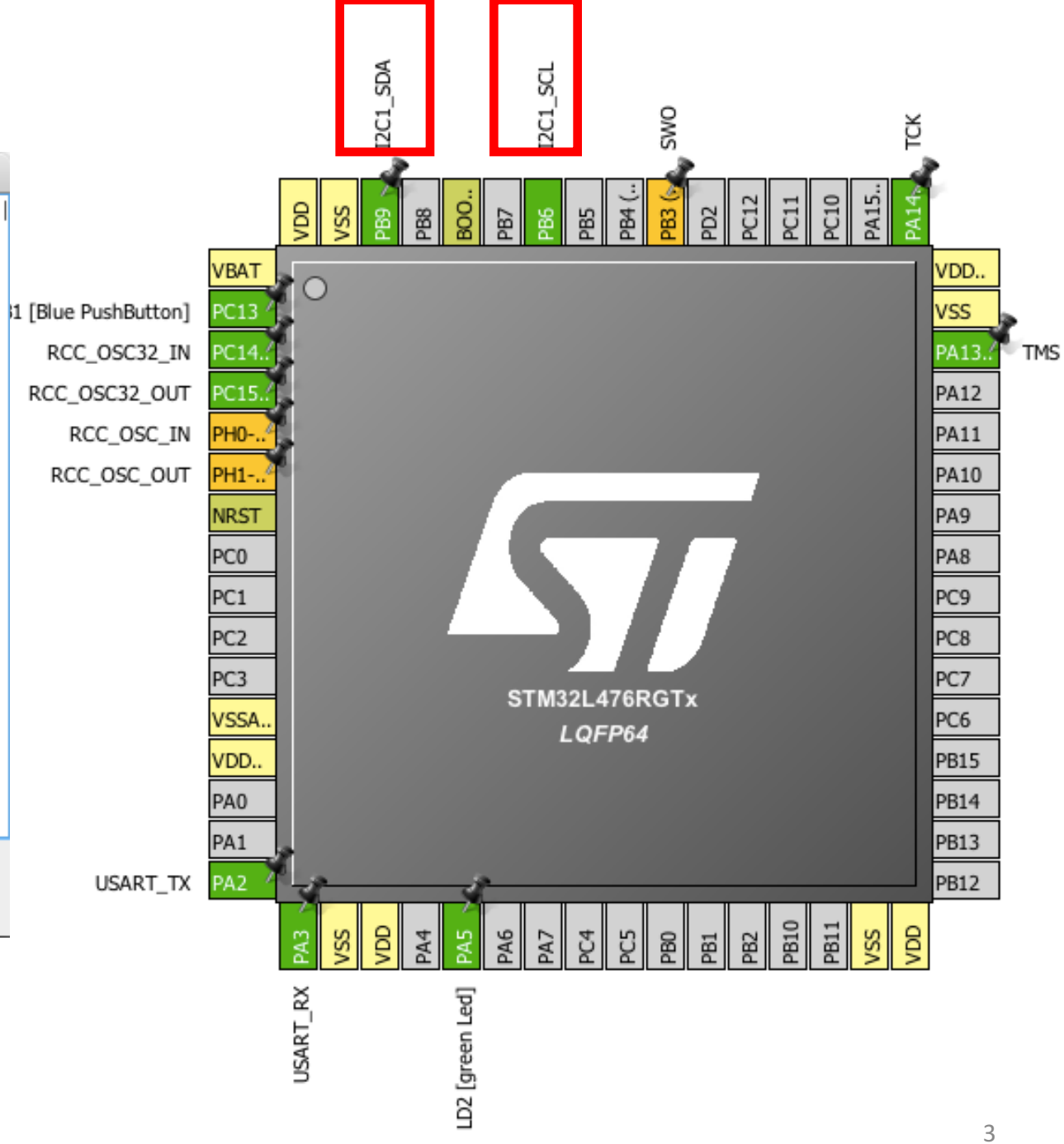
Select the Nucleo-L476RG from the Boards tab

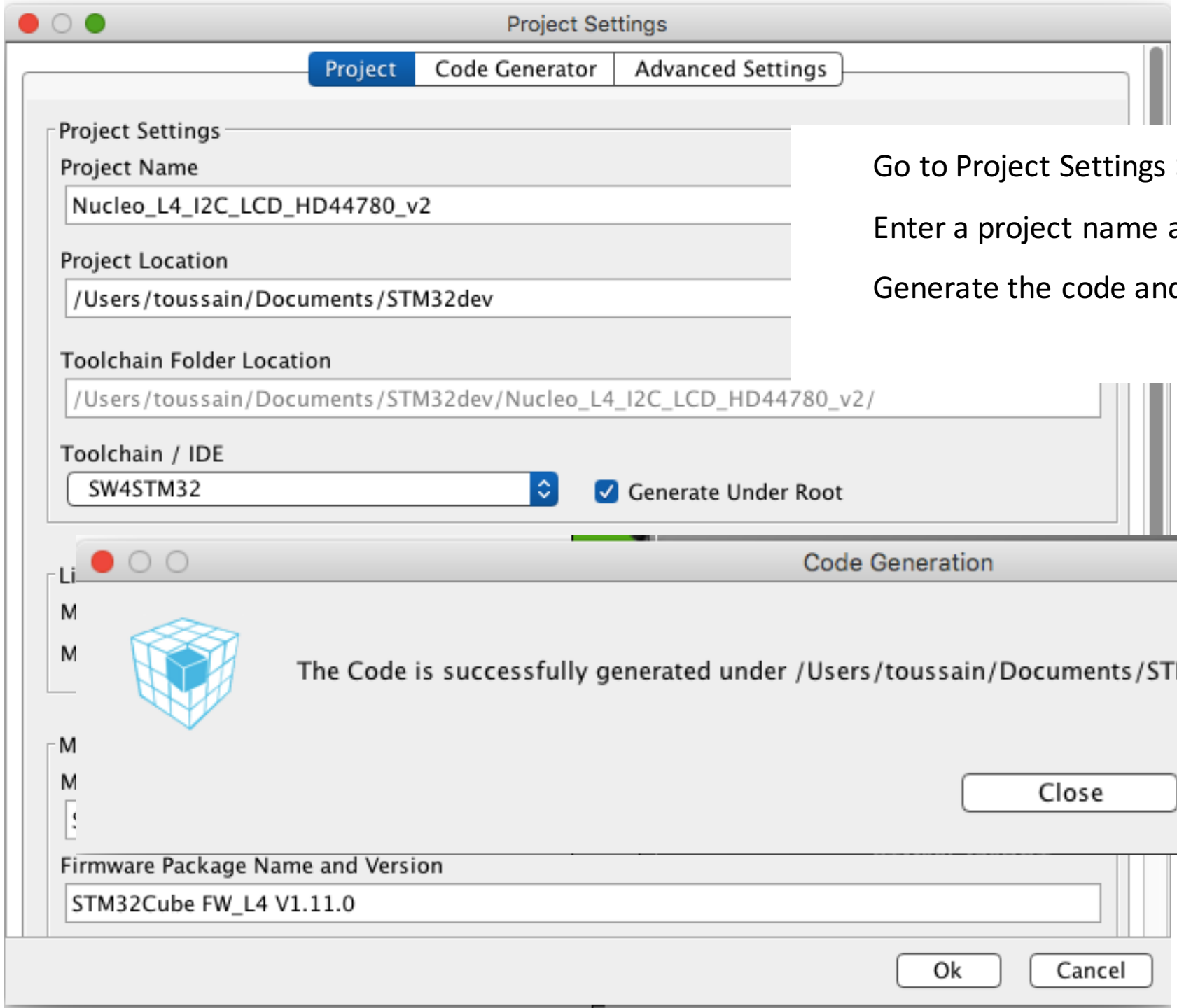
HIS Resonator Clock with PLLCLK is selected by default

Pins/Signals Options...

Pin Name	Signal Name	User Label
PC13	GPIO_EXTI13	B1 [Blue PushButton]
PC14-OSC32_IN (PC14)	RCC_OSC32_IN	
PC15-OSC32_OUT (PC15)	RCC_OSC32_OUT	
PH0-OSC_IN (PH0)	RCC_OSC_IN	
PH1-OSC_OUT (PH1)	RCC_OSC_OUT	
PA2	USART2_TX	USART_TX
PA3	USART2_RX	USART_RX
PA5	GPIO_Output	LD2 [green Led]
PA13 (JTMS-SWDIO)	SYS_JTMS-SWDIO	TMS
PA14 (JTCK-SWCLK)	SYS_JTCK-SWCLK	TCK
PB3 (JTDO-TRACESWO)	SYS_JTDO-SWO	SWO
PB6	I2C1_SCL	
PB9	I2C1_SDA	

Apply OK Cancel





Go to Project Settings > Generate code

Enter a project name and select SW4STM32

Generate the code and open the project in Eclipse IDE

Example of Application :

code to be inserted in main.c

```
/* Includes ------------------------*/
#include "main.h"
#include "stm32l4xx_hal.h"

/* USER CODE BEGIN Includes */
#include <i2c_HD44780.h>
/* USER CODE END Includes */

/* Private variables ------------*/
I2C_HandleTypeDef hi2c1;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
/* Private variables ------------*/

/* USER CODE END PV */
```

```
/* USER CODE BEGIN 2 */
 lcd_init ();

 lcd_clear_display();
 lcd_home();

 lcd_locate(1, 1);
 lcd_print_string ("Phelma");

 lcd_locate(2, 1); // 2nd row
 lcd_print_string ("Engineering School");
 HAL_Delay (2000);  // wait for 2 sec

 lcd_locate(3, 1); // 2nd row
 lcd_print_string ("Grenoble");
 HAL_Delay (2000);  // wait for 2 sec

 /* USER CODE END 2 */
```

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

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
       lcd_clear_display();
       lcd_home();
      lcd_print_string ("Physics &");
      HAL_Delay (2000);  // wait for 2 sec

      lcd_locate(2, 1); // 2nd row
      lcd_print_string ("Applied Physics &");
      HAL_Delay (2000);  // wait for 2 sec

      lcd_locate(3, 1);   // 3rd row
      lcd_print_string ("Electronics &");
      HAL_Delay (2000);  // wait for 2 sec

      lcd_locate(4, 1);   // 3rd row
      lcd_print_string ("Materials");
      HAL_Delay (2000);  // wait for 2 sec

      lcd_locate(4, 1); // 4th row
      lcd_printf("value %d\n", 10);
      HAL_Delay (2000);  // wait for 2 sec

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

I2C LCD HD4470 library for STM32

```
#include <i2c_HD44780.h>
#include <stdarg.h>
#include <stdio.h>
extern I2C_HandleTypeDef hi2c1;  // change your handler here accordingly

#define SLAVE_ADDRESS_LCD (0x3F<<1) // change this to (0x27<<1) if necessary

void lcd_send_cmd (char cmd)
{
  char data_u, data_l;
        uint8_t data_t[4];
        data_u = (cmd&0xf0);
        data_l = ((cmd<<4)&0xf0);
        data_t[0] = data_u|0x0C;  //en=1, rs=0
        data_t[1] = data_u|0x08;  //en=0, rs=0
        data_t[2] = data_l|0x0C;  //en=1, rs=0
        data_t[3] = data_l|0x08;  //en=0, rs=0

        while(HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100)
             != HAL_OK){
            /* Error_Handler() function is called when Timout error occurs */
            if (HAL_I2C_GetError(&hi2c1) != HAL_I2C_ERROR_AF)
            {
              Error_Handler();
            }
          }
}
```

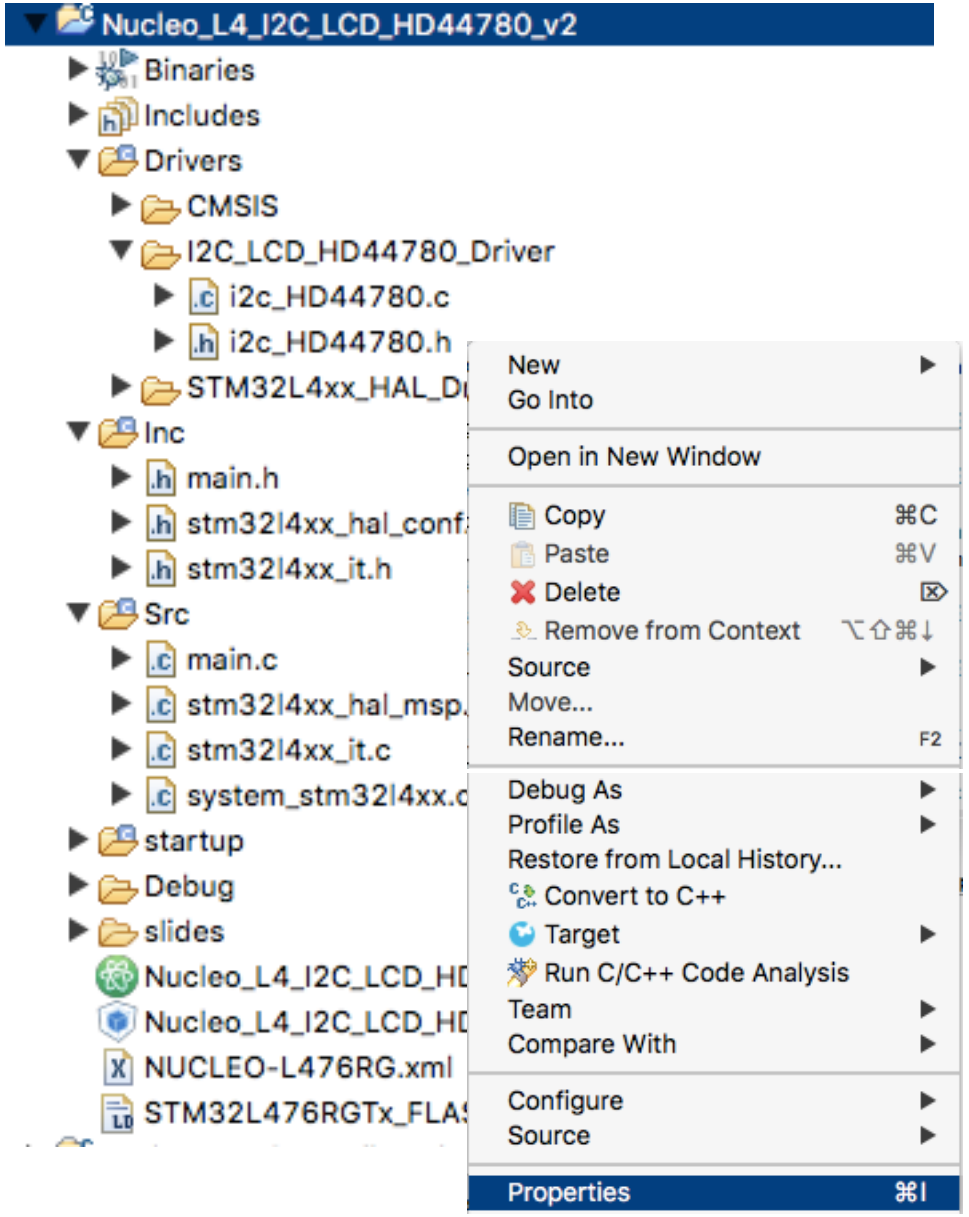
```
/**
  * @brief  send data to HD44780 LCD module via I2C
  * @param  data
  * @param  None
  * @retval None
  */
void lcd_send_data (char data)
{
        char data_u, data_l;
        uint8_t data_t[4];
        data_u = (data&0xf0);
        data_l = ((data<<4)&0xf0);
        data_t[0] = data_u|0x0D;  //en=1, rs=0
        data_t[1] = data_u|0x09;  //en=0, rs=0
        data_t[2] = data_l|0x0D;  //en=1, rs=0
        data_t[3] = data_l|0x09;  //en=0, rs=0

        while(HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100)
              != HAL_OK)
          {
            /* Error_Handler() function is called when Timout error occurs */
            if (HAL_I2C_GetError(&hi2c1) != HAL_I2C_ERROR_AF)
            {
              Error_Handler();
            }
          }
}
```

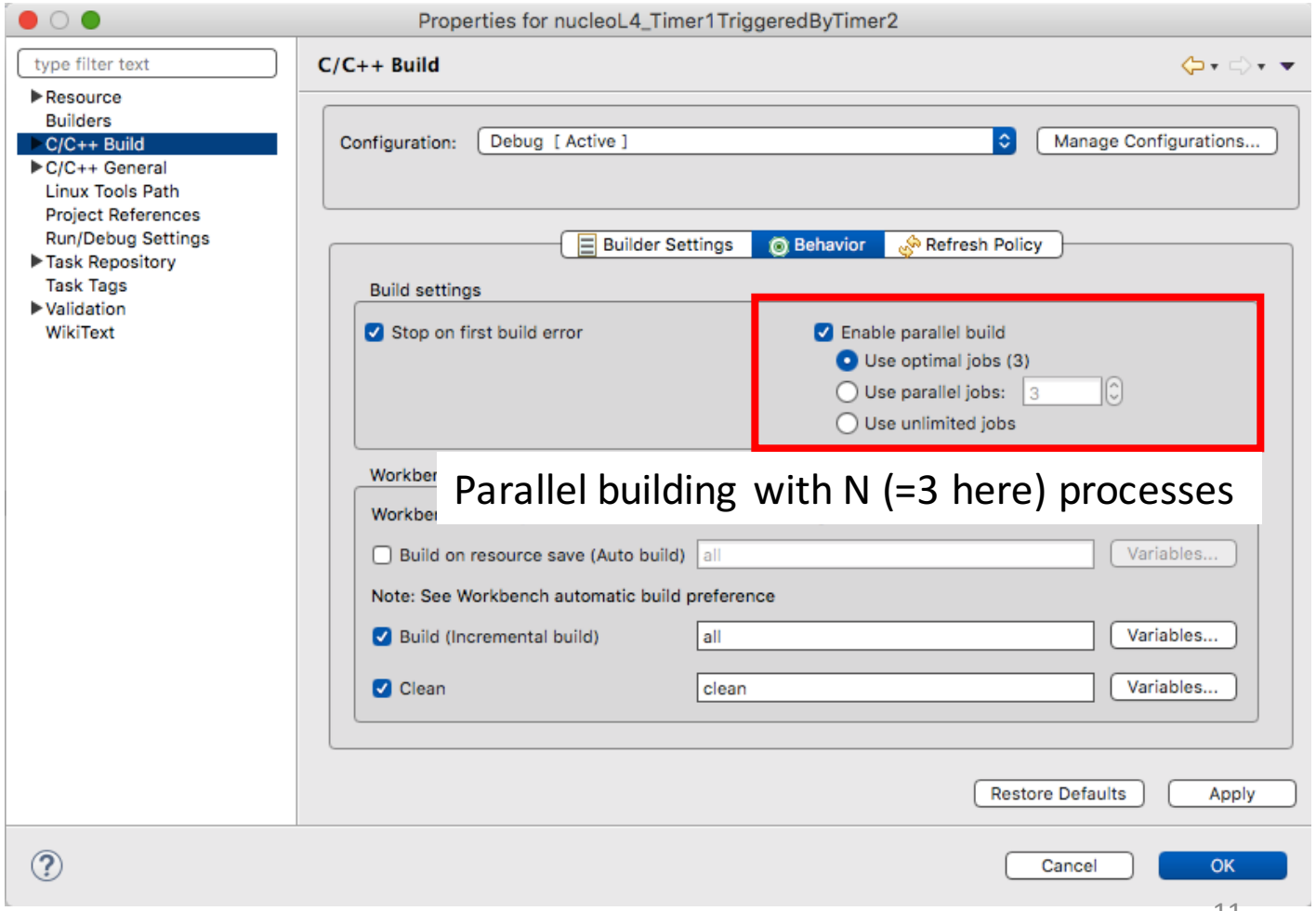
```
/**
  * @brief  Initializes HD44780 LCD module in 4-bit mode
  * @param  None
  * @param  None
  * @retval None
  */
void lcd_init (void)
{
        //Initialization of HD44780-based LCD (4-bit HW)
       lcd_send_cmd (0x33);
       lcd_send_cmd (0x32);
       lcd_send_cmd (0x28);   //Function Set 4-bit mode
       lcd_send_cmd (0x0c);   //Display On/Off Control
       lcd_send_cmd (0x06);   //Entry mode set
       lcd_send_cmd (0x02);   //Clear Display
       //Minimum delay to wait before driving LCD module
       HAL_Delay(200);
}
```

```
/**
  * @brief  Print Character on LCD module
  * @param  Ascii value of character
  * @param  None
  * @retval None
  */
void lcd_printchar(unsigned char ascode)
{
	lcd_send_data(ascode);
}

/**
  * @brief  Display of a characters string
  * @param  Text to be displayed
  * @param  None
  * @retval None
  */
void lcd_print_string (char *str)
{
	while (*str) lcd_send_data (*str++);
}
```

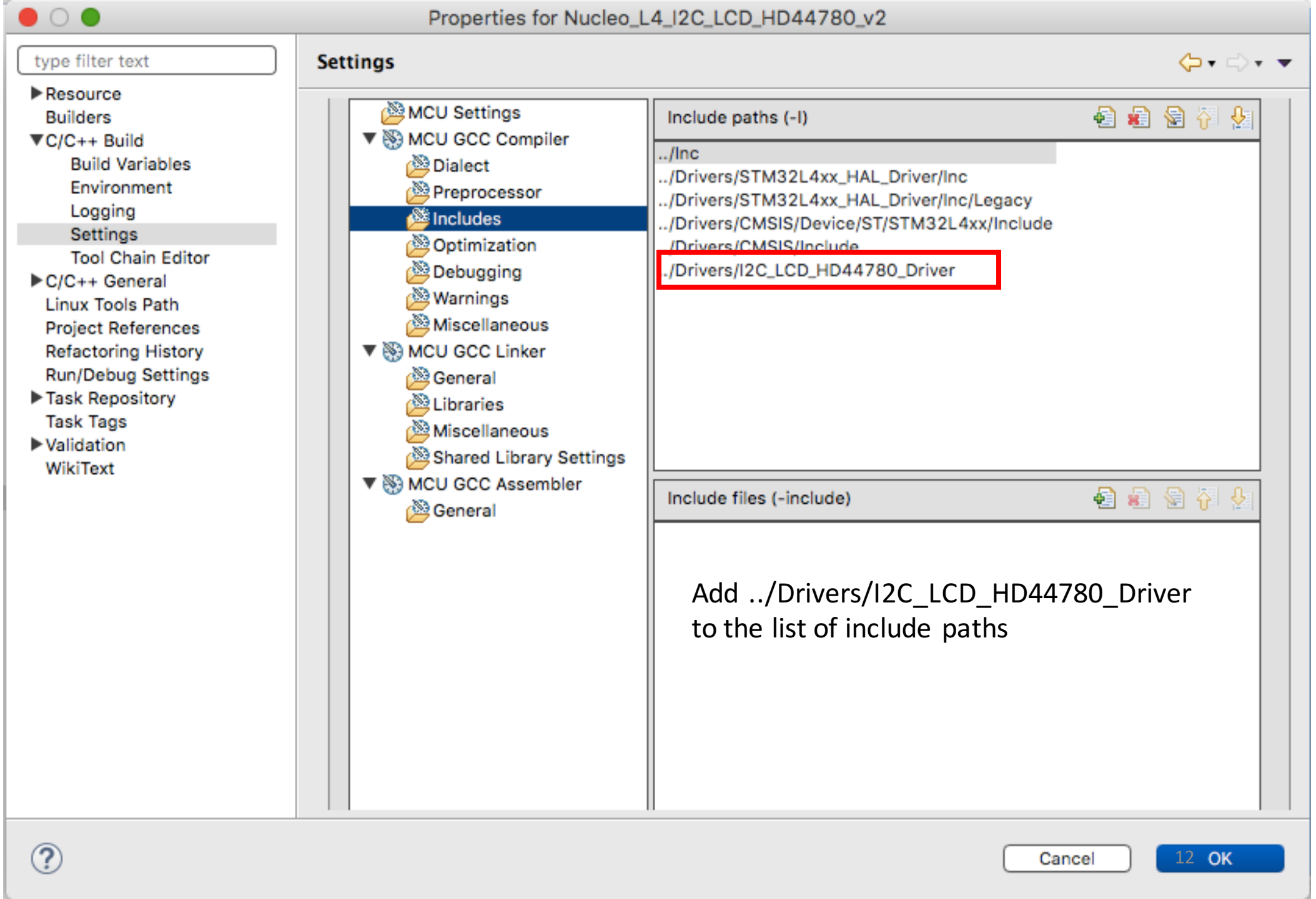
```
/**
  * @brief  lcd printf function
  * @param  string with standard defined formats
  * @param
  * @retval None
  */
void lcd_printf(const char *fmt, ...)
{
  uint32_t i;
  uint32_t text_size, letter;
  static char text_buffer[32];
  va_list args;

  va_start(args, fmt);
  text_size = vsprintf(text_buffer, fmt, args);

  // Process the string
  for (i = 0; i < text_size; i++){
    letter = text_buffer[i];

    if (letter == 10)
      break;
    else{
      if ((letter > 0x1F) && (letter < 0x80))
        lcd_printchar(letter);
    }
  }
}
```

Eclipse Configuration

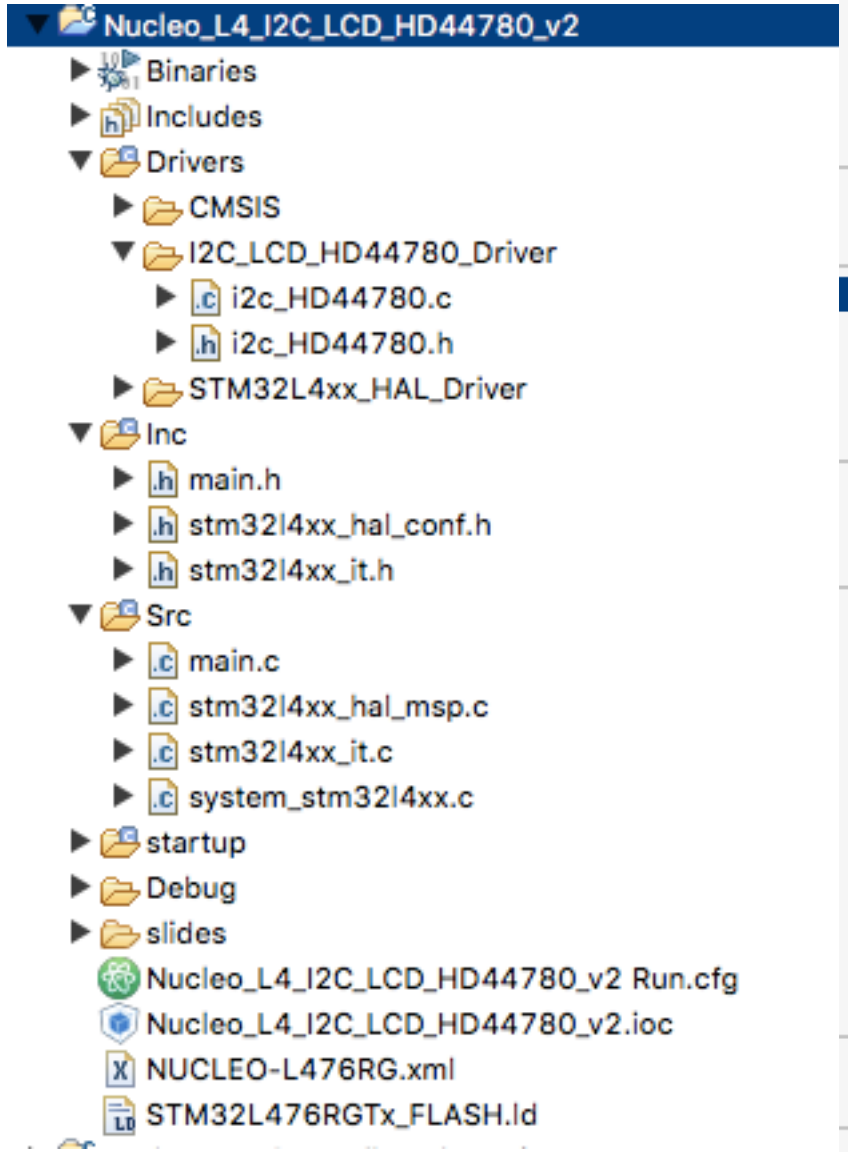


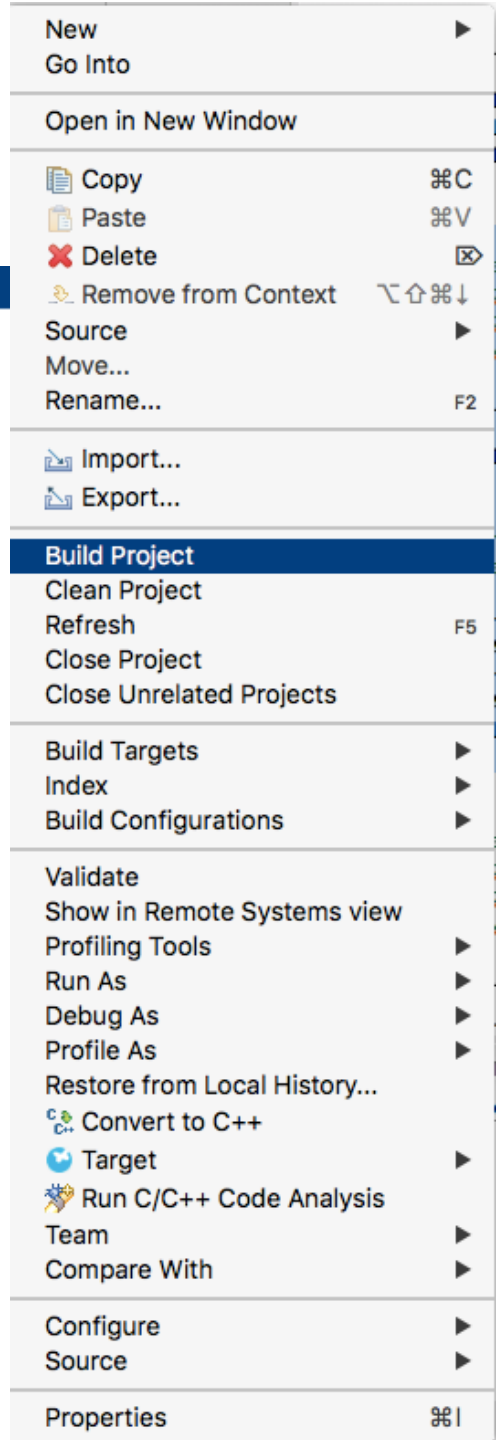


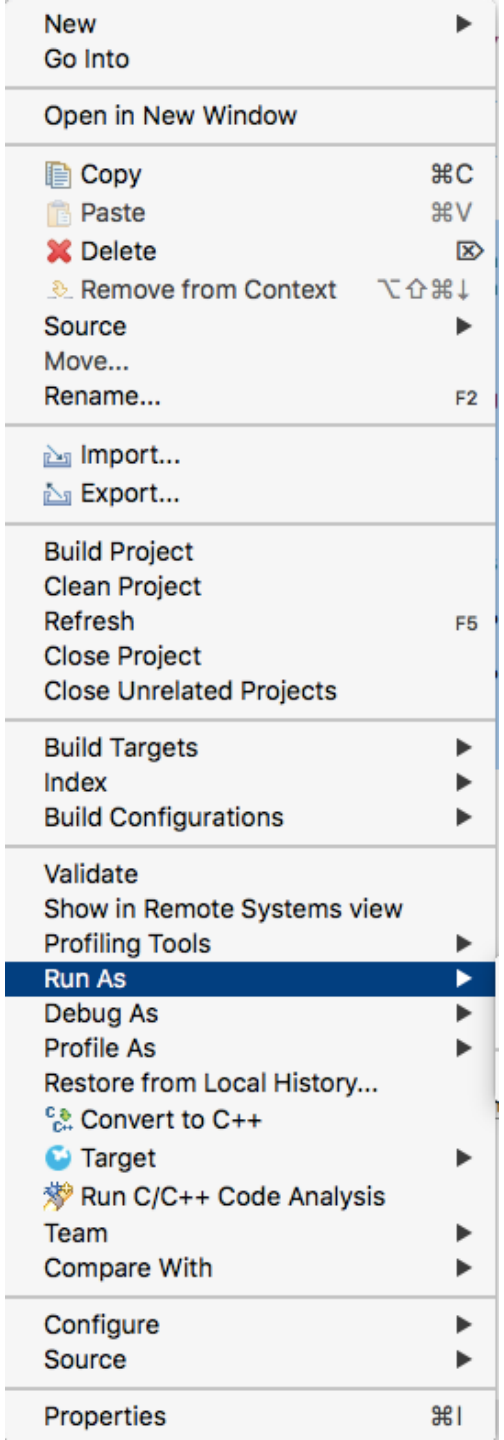
Include Paths

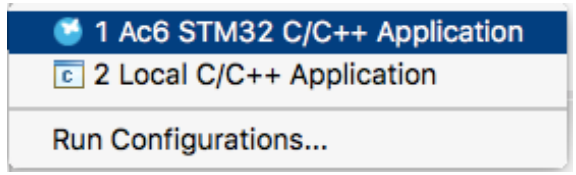


Compile and then Run as a STM32 runtime

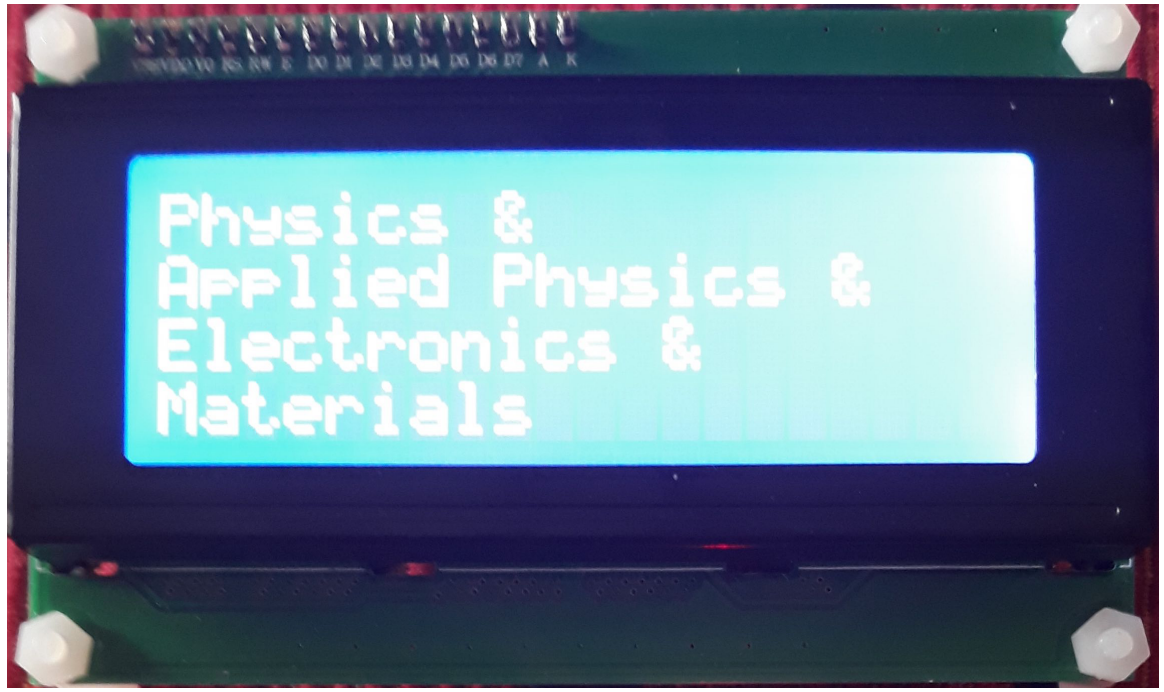


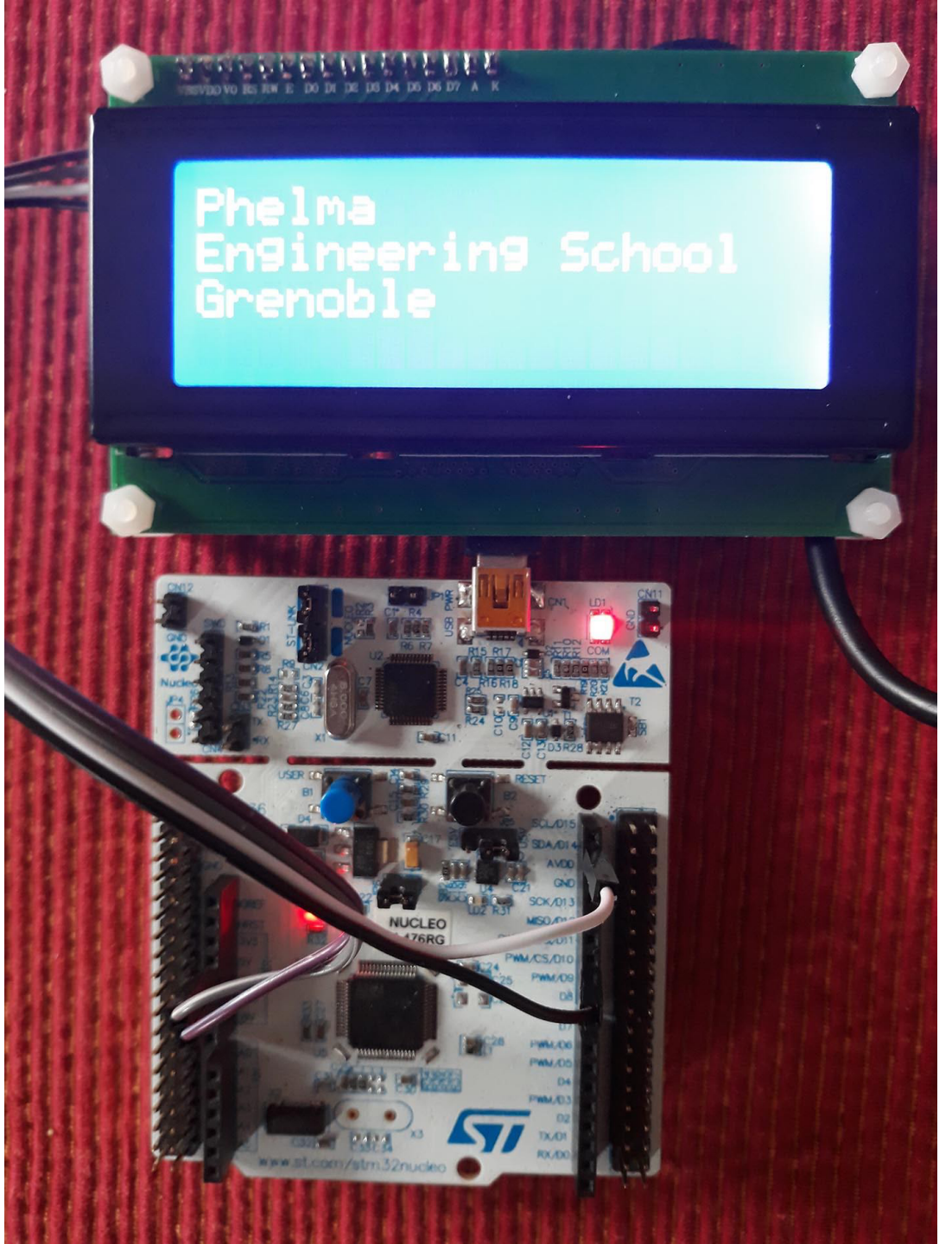






In Practice


Physics &
Applied Physics &
Electronics &
Materials


Phelma
Engineering School
Grenoble
NUCLEO

References

1. Carmine Noveillo, “Mastering STM32”, https://leanpub.com/mastering-stm32

2. Marc Laury, “A la découverte des cartes Nucleo”, Eyrolles, EAN13 : 9782212673692

3. Reference Manual for STM32L4, DM00083560.pdf

4. https://community.st.com/docs/DOC-1949-tutorial-interfacing-an-hd44780-lcd-display-with-stm32l4

5. https://community.st.com/docs/DOC-1413-tutorial-interfacing-a-stm32l053-discovery-with-an-i2c-sensor