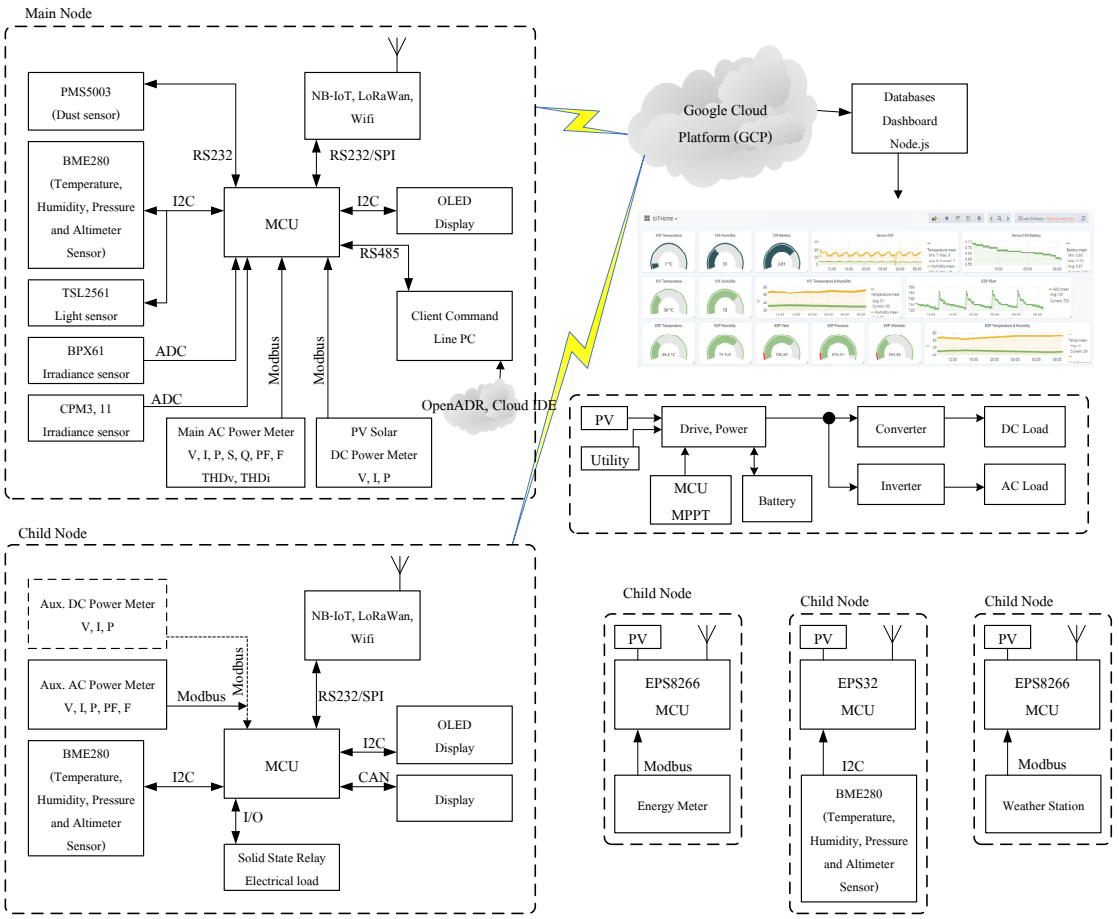


Brief STM32 Microcontroller Tutorial For Microcontroller Beginners

Documentation
Chiang Mai, 2019

Author: **Worrajak Muangjai**, Ph.D.in Electrical Engineering
Rajamangal University of Technology Lanna

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1

STM32F103C8

1.1 Basic STM32F103C8

- ไมโครคอนโทรลเลอร์ที่ใช้กับ Boot Loader arduino
- โปรแกรม Integrate Development Environment :IDE ที่ใช้
- ข้อกำหนดการใช้งาน และการตั้งค่าการใช้งานโปรแกรม
- การใช้งานฟังก์ชั่นประกอบของ IDE
- ภาษาโปรแกรม Built in ของ IDE
- การเขียนโปรแกรมภาษา C ประกอบการใช้งาน IDE เพื่อการกำหนดคุณสมบัติของไมโครคอนโทรลเลอร์ให้การใช้งานดียิ่งขึ้น
- การใช้งาน IDE กับไมโครคอนโทรเลอร์เบอร์อื่น

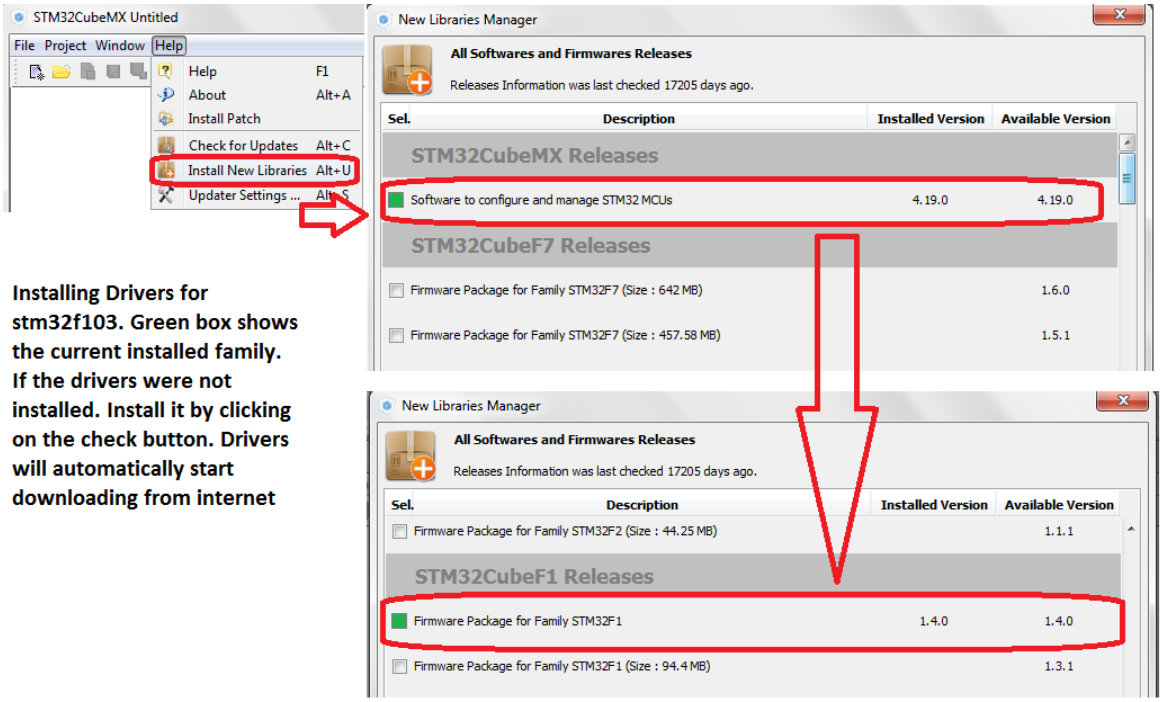
After working a lot with 8-bit microcontrollers and learning all the protocols and functions they offer, now i thought to switch to 32-bit microcontrollers. I decided to go with arm 32-bit processors, because of their popularity in the market. I choose arm cortex-m3 processor series for my new hobby/learning projects. The reason behind choosing the arm cortex-m3 series is cortex-m3 processors are especially made for connected embedded applications, and the microcontrollers built with this series are used in many mid level of embedded projects/ applications/products. After so much googling i finally took decision to move forward with stm32 microcontrollers. Stm32 is a family of 32-bit microcontrollers offered by STMicroelectronics. Stm32 microcontrollers are built around Cortex-M7, Cortex-M4F, Cortex-M3, Cortex-M0+, and Cortex-M0 processors.

After decided to go with stm32, i started to took initial tutorials on how to get started with stm32. I found much information on getting started but non of it is well organized. So i decided to make series of tutorials on getting started with stm32 microcontrollers. In this whole series i will discuss all the protocols/functions stm32 microcontrolles offered. I will present a working example of easy protocol/interface/ function with source code and circuit diagram. Their are many IDE's(Integrated development environments) that supports stm32 series and you can use any one of them to program your stm32 microcontroller. Some Ide's are Coocox, keil, mBed, Attolic, microC for Arm. I decide to go with keil and stm32CubeMx. Keil gives you in depth knowledge of the microcontroller and its interface. If you don't want to go in depths and want a piece of cake then go with MikroElectronica microC for Arm. Its very easy to work with microC ide, it has plenty of example and libraries you only need to call the functions and its all done. Stm32CubeMx is a microcontroller peripheral configurater. By using stm32cube you don't need to write configuration code for your stm32 microcontroller. Its a visual platform where you can make the microcontroller pins input, output, enable pull-ups and pull-down can define the operating frequency of microcontroller visually and lot more. After the visual configuration you can generate code for the configuration you made. I prefer to work with stmCube because it is provided by officially STMicroelectronics and its good to work with the stuff provided by the owner.

Note: Stm32CubeMx is not an Ide its a configuration manager. You make your stm32 microcontroller configuration in it and then generate code for the configuration you done to be used with any other ide. You can directly generate the keil ide project with stmCubeMx by slecting the option from stmCubeMx to translate the configuration in to keil ide project.

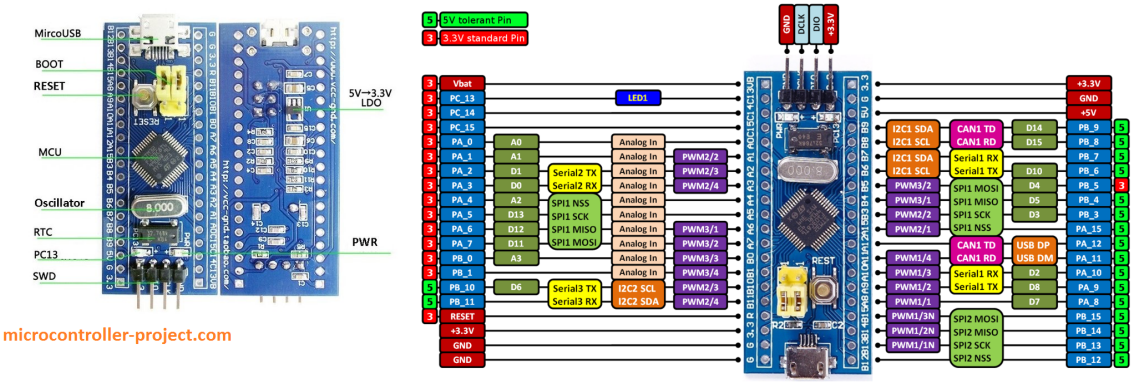
1.1.1 Installing StmCubeMx and Keil MDK ARM

Installing Keil Ide is pretty straight forward. Download the MDK ARM kit it contains the keil Ide in it you don't need the ide to install it separately. Just download the file from the link and install the ide. http://www2.keil.com/mdk5 . Arm mdk is available in many editions latest is mdk5. I am using mdk5 for my projects and this tutorial is also based on mdk5 and keil uvision–5 ide. Installing the StmCubeMx is also straight froward. Download the StmCube from the link http://www.st.com/en/development–tools/stm32cubemx.html. You might be popped up to sign in. Sign in and you are their. Once StmCubeMx is installed you now have to install the packages for each stm32 series or for series that you want to work with. StmCubeMx and the stm32 series packages are two different things. Package for each stm32 series must be installed separately. Their is also an option in StmCubeMx software that installs the series packages. Its under the Help> Install New Libraries. Since we are getting started with stm32f103 so wee need to install the package for stm32f1 series. The diagram below explains well about the packages installation.



รูปที่ 1.1

I bought a cheap stm32f103c8x module from aliexpress. Its cost me about $4.5 with free shipping to Pakistan. Shipping took almost about 1.5 months 45 days. The board is cool and offer almost all the features necessary for getting started with stm32 microcontrollers. Pin out of the board is given below.



รูปที่ 1.2

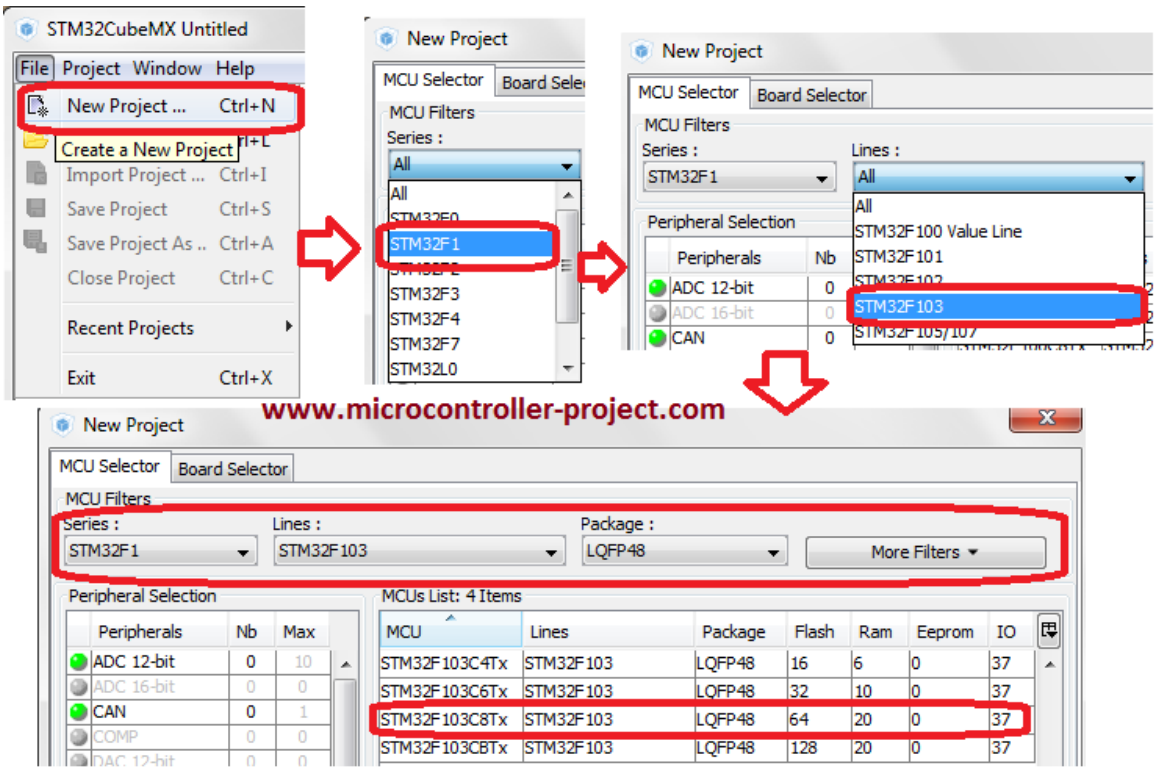
1.1.2 Blink led with stm32f103 keil and stmcubemx

The upper module has an led connected to port-c pin#13. In getting started we are going to blink it. Stm32 microcontroller pins offers multiple features on a single pin. Selecting one and disabling others should be handled carefully. Stm32 microcontrollers I/O pins can be used in five modes Input mode Analog mode Output mode Alternate function mode External interrupt/event lines For our purpose we are going to use port-c pin#13 as output. Almost all the stm32 pins have internal pull up and pull down resistors. Since we are not using the gpio in input mode, so we are not using pull up and down resistors. Stm32 pins can work at different frequencies, we are going to operate the pin at low frequency. Stm32 pins can also be initialized as low or high after booting. I made pin#13 low.

Note: I am going to use the internal 8Mhz RC oscillator of the microcontroller. The upper board has an external 8Mhz crystal but i am not going to use it. In later tutorials we will use it.

1.1.3 Creating new project in StmCubeMx

Creating new project is straight forward. Go to File> New Project. A window will appear select your microcontroller series, series lines and package. After selecting the package click on the microcontroller that you are using for your project. In our case its STM32f103C8Tx.



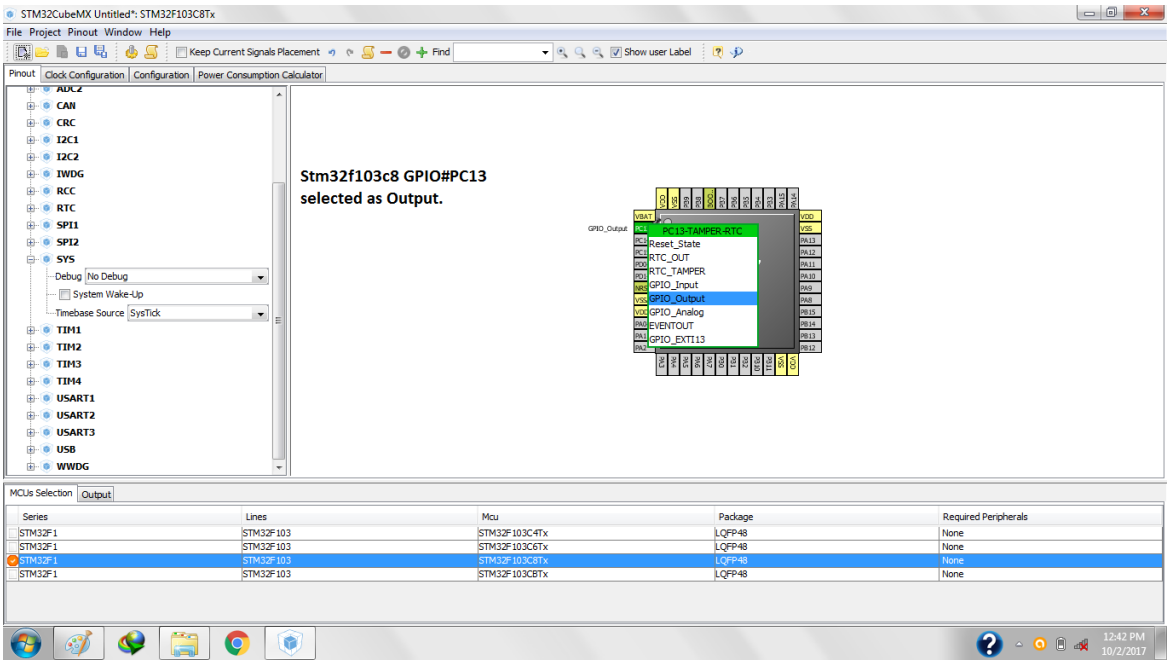
รูปที่ 1.3

After Selecting the microcontroller stm32f103c8 a window will appear containing the mcu diagram. Click on the pin#13 and set the gpio as output.

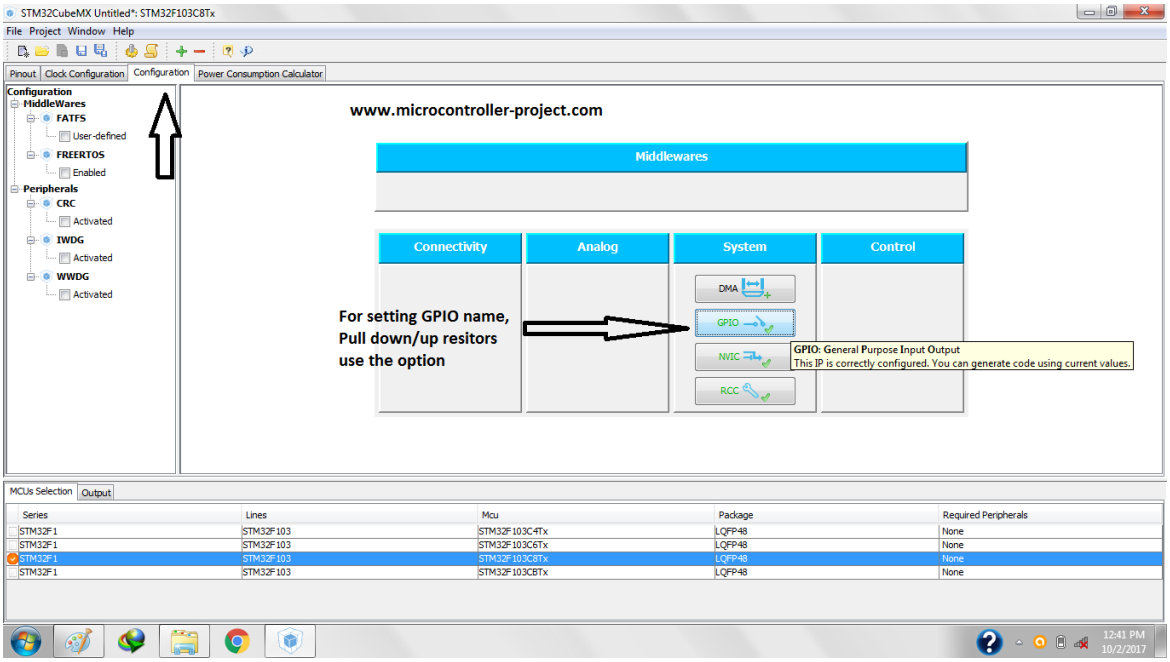
Click on the pin names for its settings to appear. In setting window set the gpio mode, output level, clock speed and give gpio a name. Press apply to apply to apply the changes.

The name we gave to pin now appears on the pin. We can use the gpio in code with this name. Now its time to generate the code. Click on the gear icon for generating code.

A window appears after clicking on the gear icon.Name the project and selecte the location for the project files. Since we are using MDK-ARM V5 so in ToolChain/IDE select mdk-arm. Click OK to generate the code.

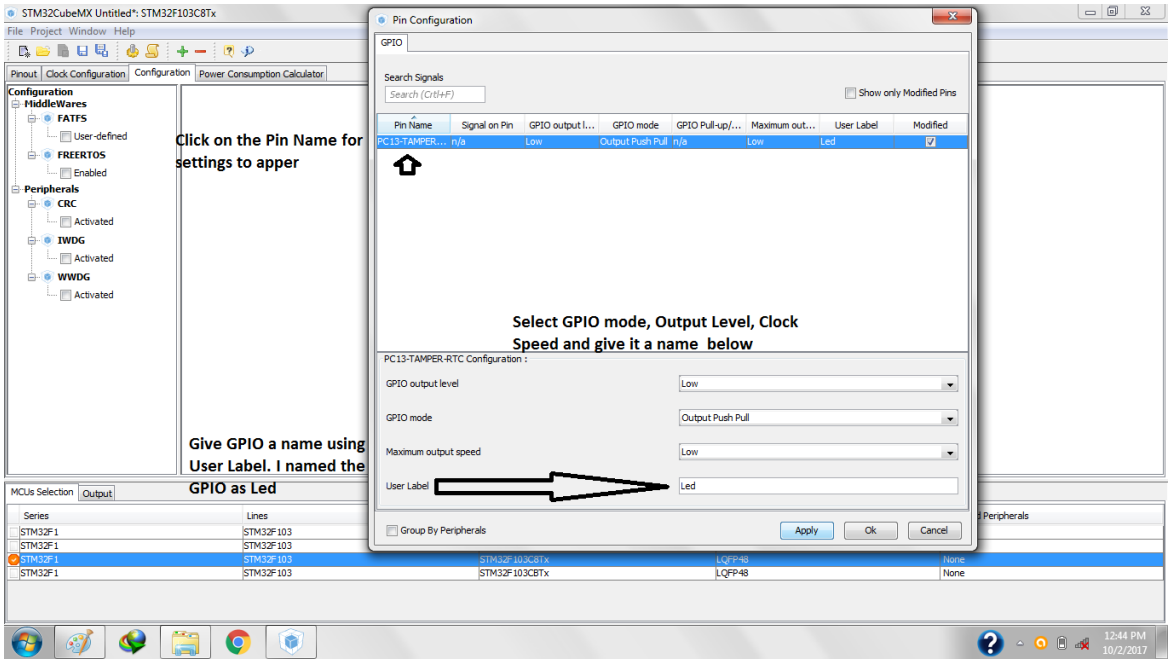


รูปที่ 1.4

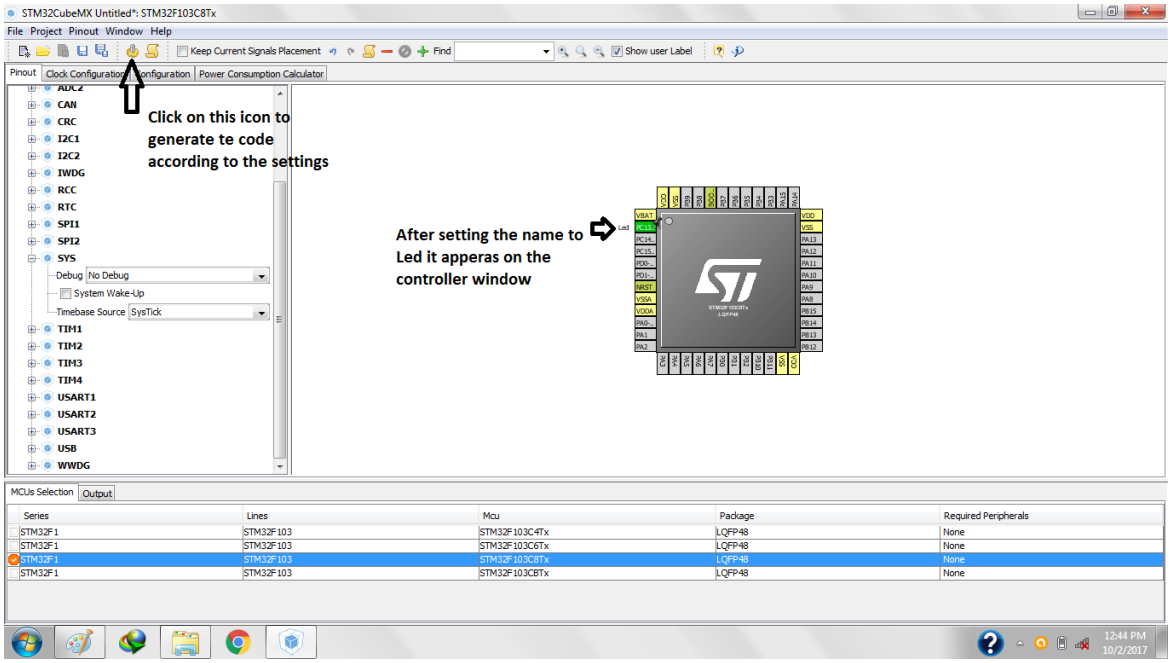


รูปที่ 1.5

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f1xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */
```

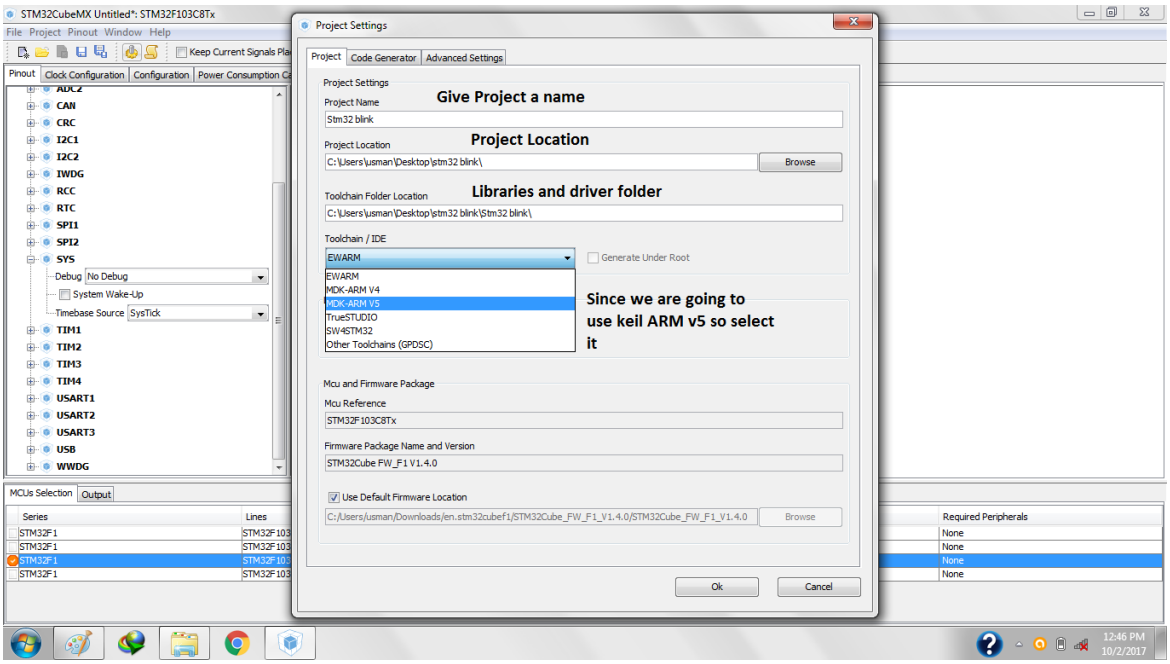


รูปที่ 1.6



รูปที่ 1.7

```
/* Private variables ---------------------------------------------------------*/

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

/* USER CODE END PV */
```



รูปที่ 1.8

```
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
* @brief The application entry point.
*
* @retval None
*/
int main(void)
{
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* MCU Configuration--------------------------------------------------------*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
```

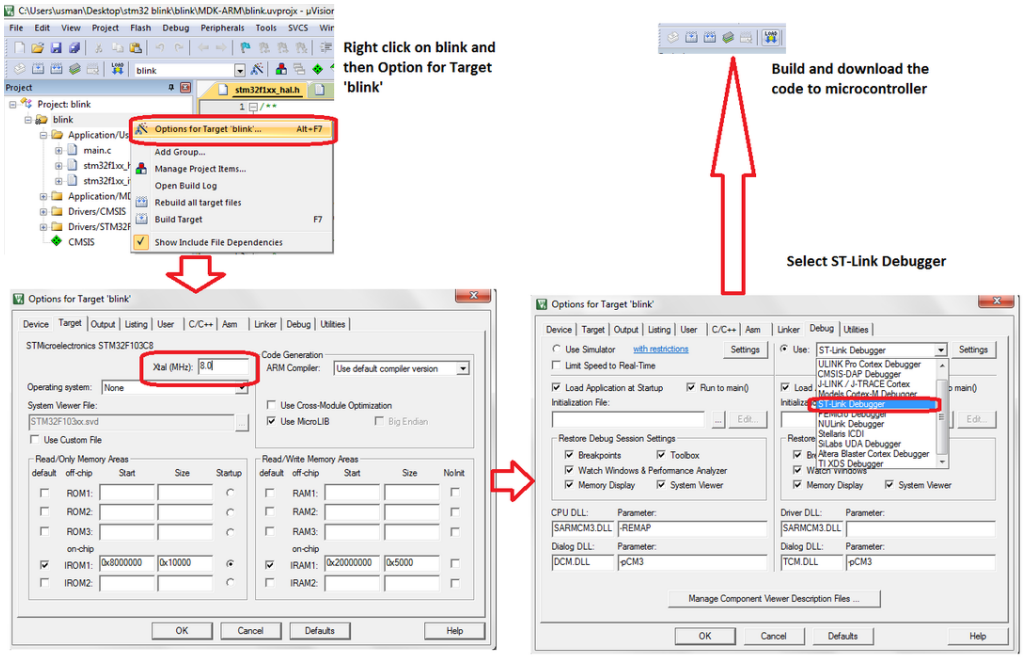
```
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

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

/* USER CODE END WHILE */
HAL_GPIO_TogglePin(LED_GPIO_Port,LED_Pin);
HAL_Delay(50);
/* USER CODE BEGIN 3 */

}
/* USER CODE END 3 */

}

/**
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{

RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = 16;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
```

```
}

/**Configure the Systick interrupt time
*/
HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

/**Configure the Systick
*/
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

/* SysTick_IRQn interrupt configuration */
HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/** Configure pins as
* Analog
* Input
* Output
* EVENT_OUT
* EXTI
*/
static void MX_GPIO_Init(void)
{

GPIO_InitTypeDef GPIO_InitStruct;

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOC_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin : LED_Pin */
GPIO_InitStruct.Pin = LED_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
* @brief This function is executed in case of error occurrence.
* @param file: The file name as string.
* @param line: The line in file as a number.
* @retval None
*/
void _Error_Handler(char *file, int line)
{
/* USER CODE BEGIN Error_Handler_Debug */
/* User can add his own implementation to report the HAL error return state */
while(1)
{
}
/* USER CODE END Error_Handler_Debug */
}
```

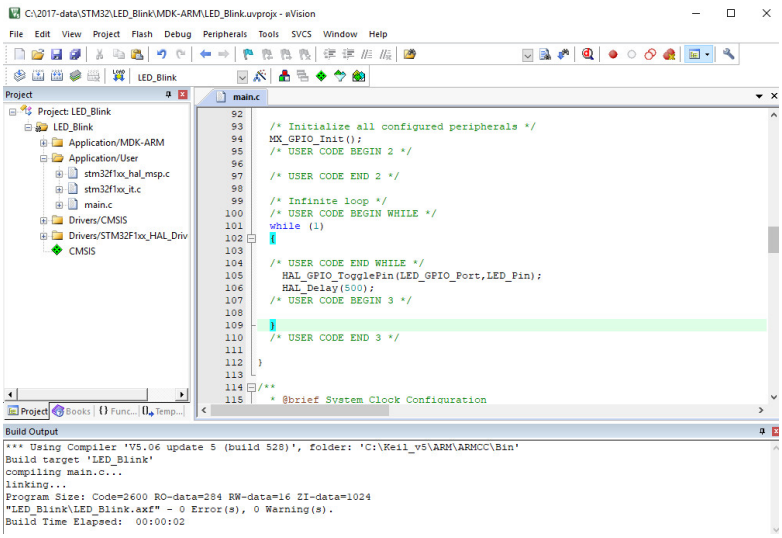
```
#ifdef USE_FULL_ASSERT
/**
* @brief Reports the name of the source file and the source line number
*        where the assert_param error has occurred.
* @param file: pointer to the source file name
* @param line: assert_param error line source number
* @retval None
*/
void assert_failed(uint8_t* file, uint32_t line)
{
/* USER CODE BEGIN 6 */
/* User can add his own implementation to report the file name and line number,
tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

1.1.4 Upolad code to stm32 microcontroller

Since the development board has no on build programmer so i used an external St Link Programmer to upload the code on the board. For uploading the code you have to make some settings in keil. Follow the below steps to upload the code.



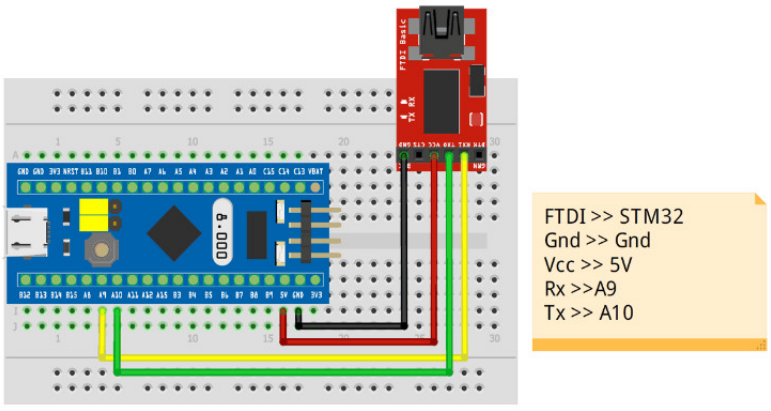
รูปที่ 1.9



รูปที่ 1.10

1.2 การใช้งาน arduino IDE เพื่อการโปรแกรม STM32F103C8

การต่อวงจรจาก USB to Serial เพื่อโปรแกรม STM32F103 ต่อดังรูป 1.11

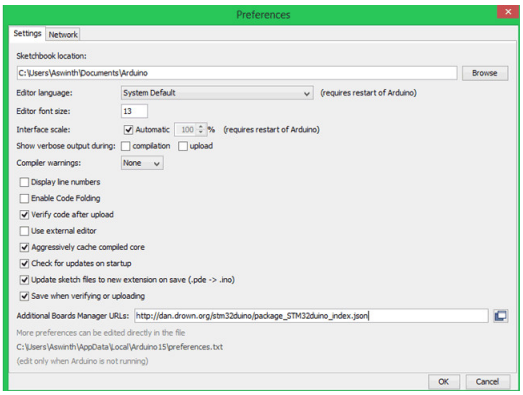


รูปที่ 1.11: การโปรแกรม STM32F103 board bluepill

Step 1:– If you have not yet installed the Arduino IDE, download and install it from this link [https://www.arduino.cc/en/Main/Software]. Make sure you select your correct operating system.

รูปที่ 1.12: www.arduino.cc

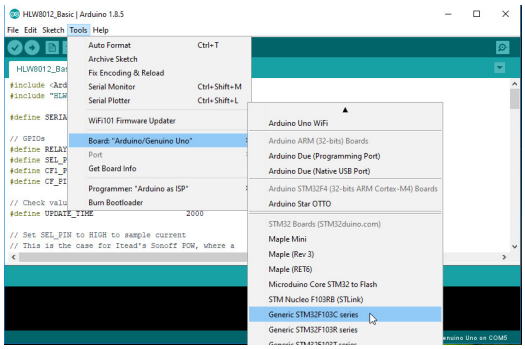
Step 2:– After Installing the Arduino IDE open and download the required packages for the STM32 board. Thi can be done by selecting File –> Preferences.



รูปที่ 1.13: การเพิ่ม STM32F103 เข้าไปใน Preferences

Step 3:– Clicking on Preferences will open the below shown dialog box. In the additional Boards Manager URL text box paste the below link http://dan.drown.org/stm32duino/package_STM32duino_index.json and press OK.

Step 4:– Now go to Tool –> Boards –> Board Manager. This will open the Boards manager dialog box, search for “STM32F1” and install the package that appears.



รูปที่ 1.14: การเลือกใช้งาน MCU STM32F103

Step 5:– After the package, installation is completed. Go to Tools and scroll down to find the Generic STM32F103C series as shown below. Then make sure the variant is 64kFlah type, CPU speed is 72MHz and change the upload method to Serial.

Step 6:– Now, connect your FTDI board to the computer and check to which COM port the FTDI board is connected to using device manager. Then, select the same port number in Tools–>Port

Step 7:– After all the changes are made, check the bottom right corner of the Arduino IDE and you should notice the following setting being set. My FTDI board is connected to COM7 but yours might differ