

# บทที่ 1

## STM32F103C8

### 1.1 Basic STM32F103C8

- ไมโครคอนโทรลเลอร์ที่ใช้กับ Boot Loader arduino
- โปรแกรม Integrate Development Environment :IDE ที่ใช้
- ข้อกำหนดการใช้งาน และการตั้งค่าการใช้งานโปรแกรม
- การใช้งานฟังก์ชันประกอบของ IDE
- ภาษาโปรแกรม Built in ของ IDE
- การเขียนโปรแกรมภาษา C ประกอบการใช้งาน IDE เพื่อกำหนดคุณสมบัติของไมโครคอนโทรลเลอร์ให้การใช้งานดียิ่งขึ้น
- การใช้งาน IDE กับไมโครคอนโทรลเลอร์เบอร์อื่น

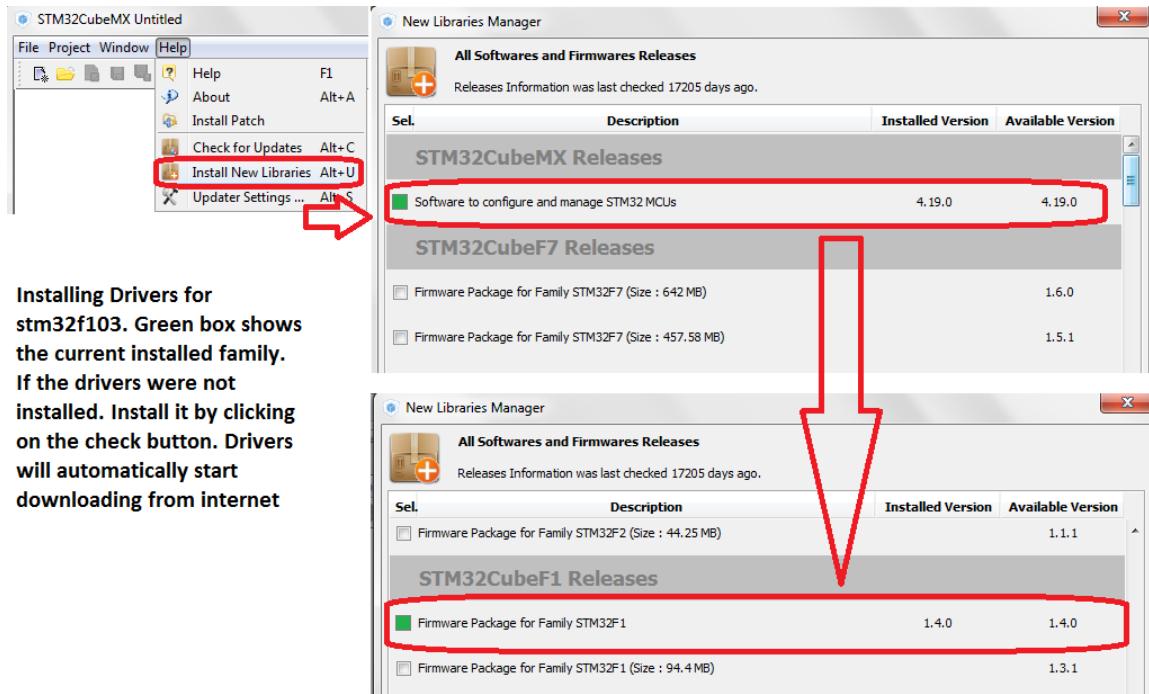
After working a lot with 8-bit microcontrollers and learning all the protocols and functions they offer, now i thought to switch to 32-bit microcontrollers. I decided to go with arm 32-bit processors, because of their popularity in the market. I choose arm cortex-m3 processor series for my new hobby/learning 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. There 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.

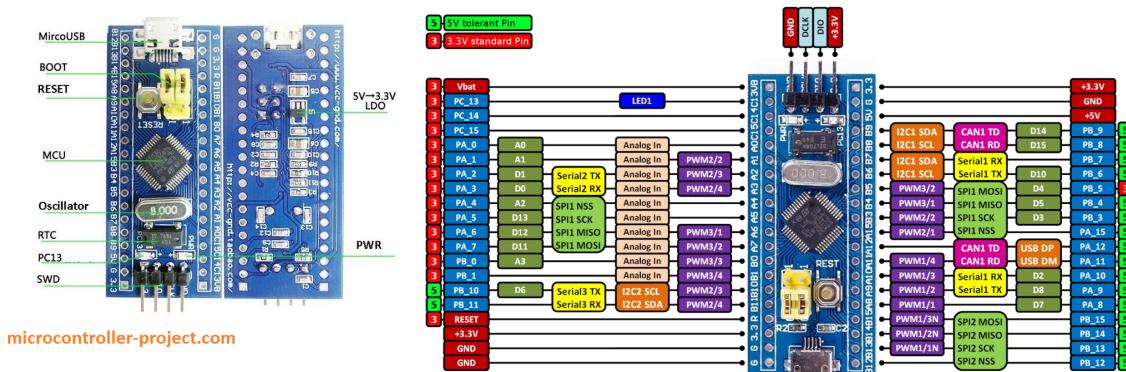
### 1.1.1 Installing StmCubeMx and Keil MDK ARM

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

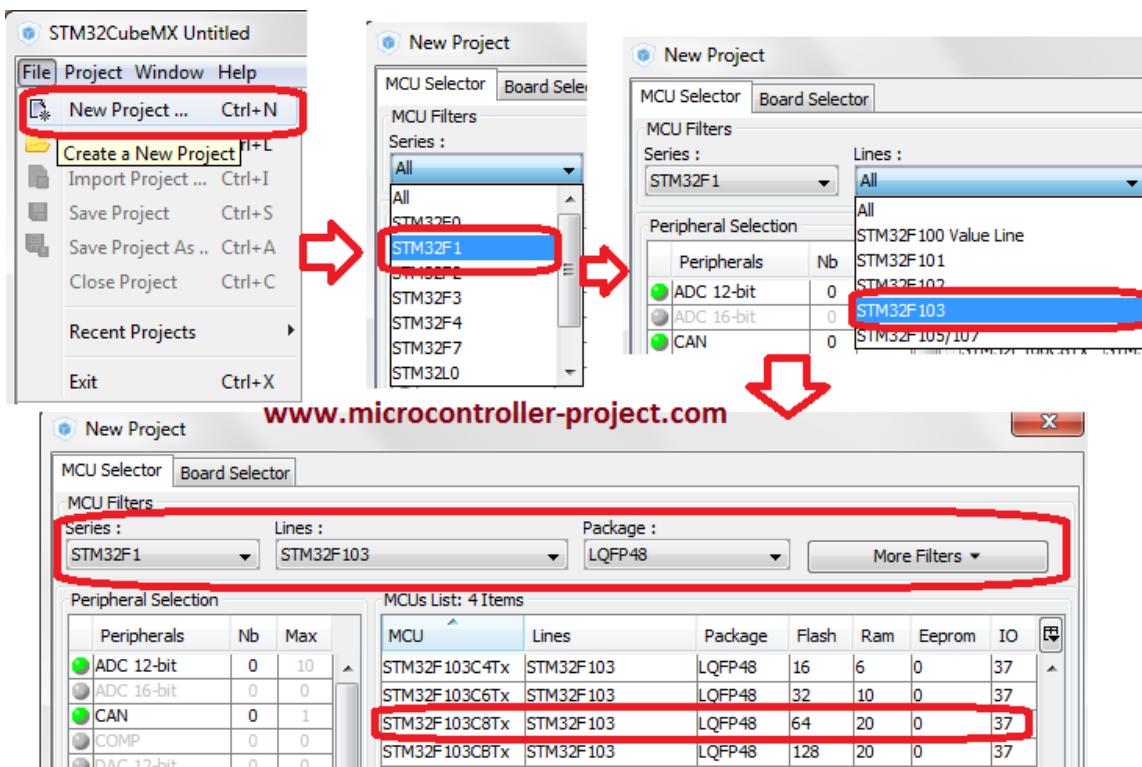
### 1.1.2 Blink led with stm32f103 keil and stmcubemx

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

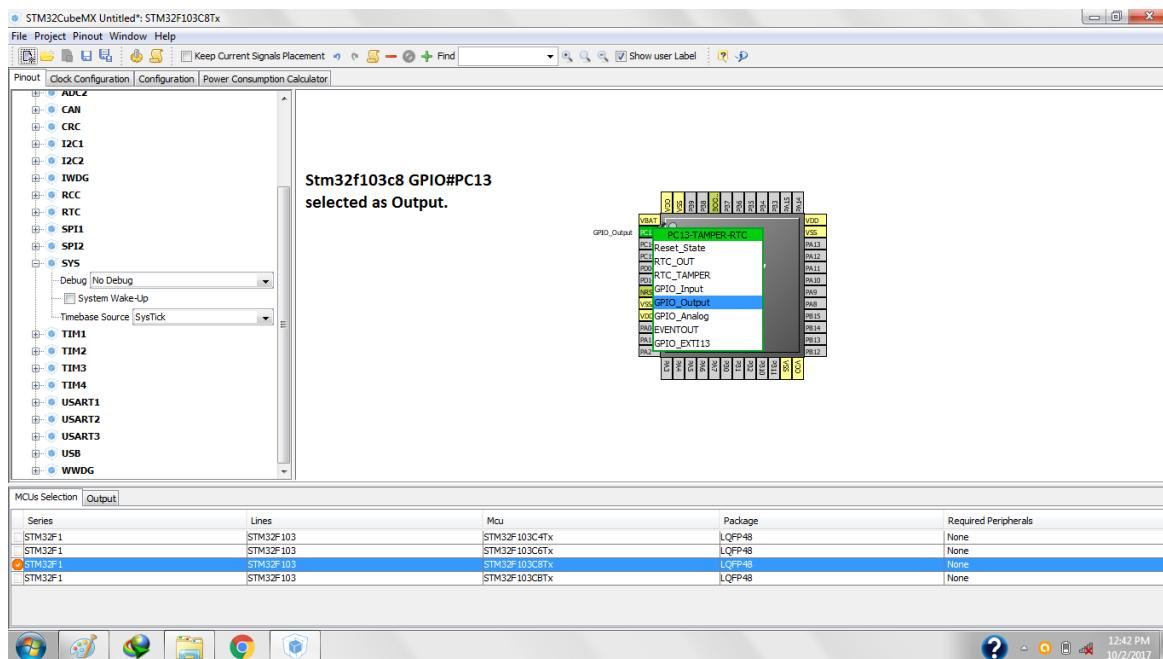
After Selecting the microcontroller stm32f103c8 a window will appear containing the mcu diagram. Click on the pin#13 and set the gpio as output.

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 the changes.

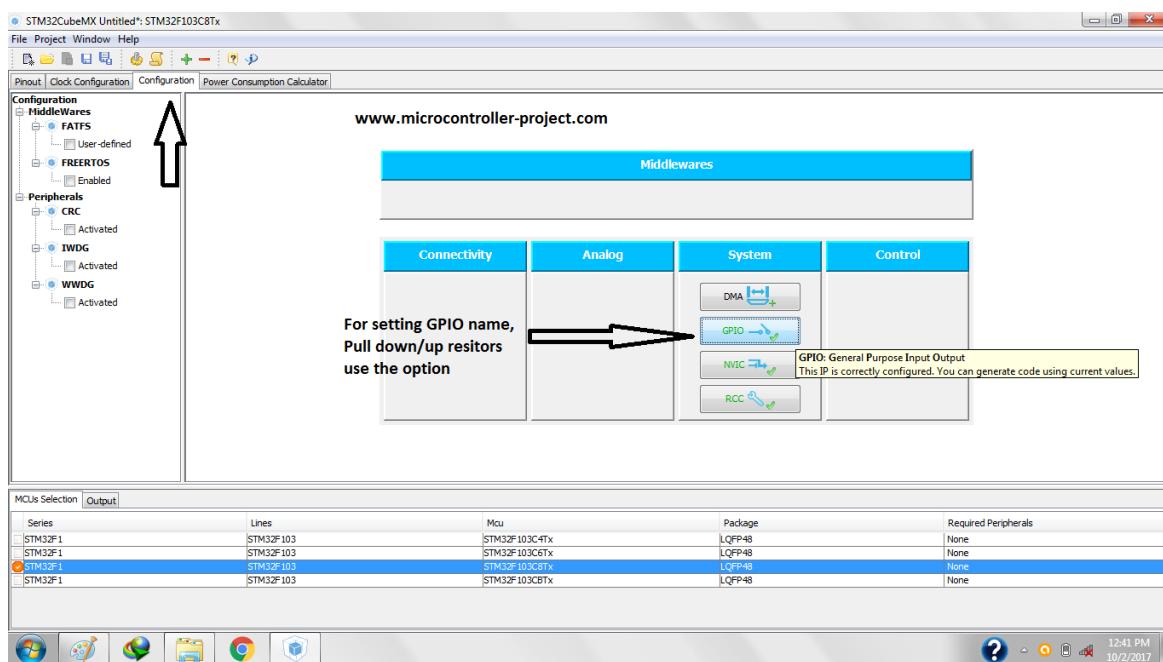
The name we gave to pin now appears on the pin. We can use the gpio in code with this name. Now its time to generate the code. Click on the gear icon for generating code.

A window appears after clicking on the gear icon. Name the project and select 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.

## eBook Microcontroller



รูปที่ 1.4



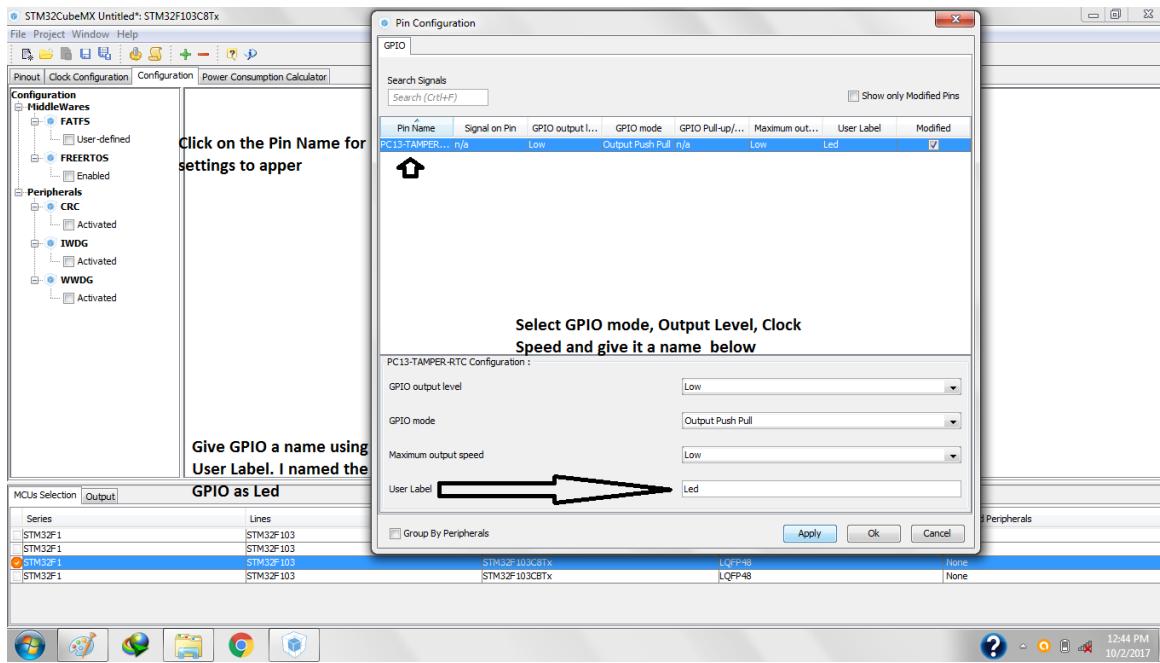
รูปที่ 1.5

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

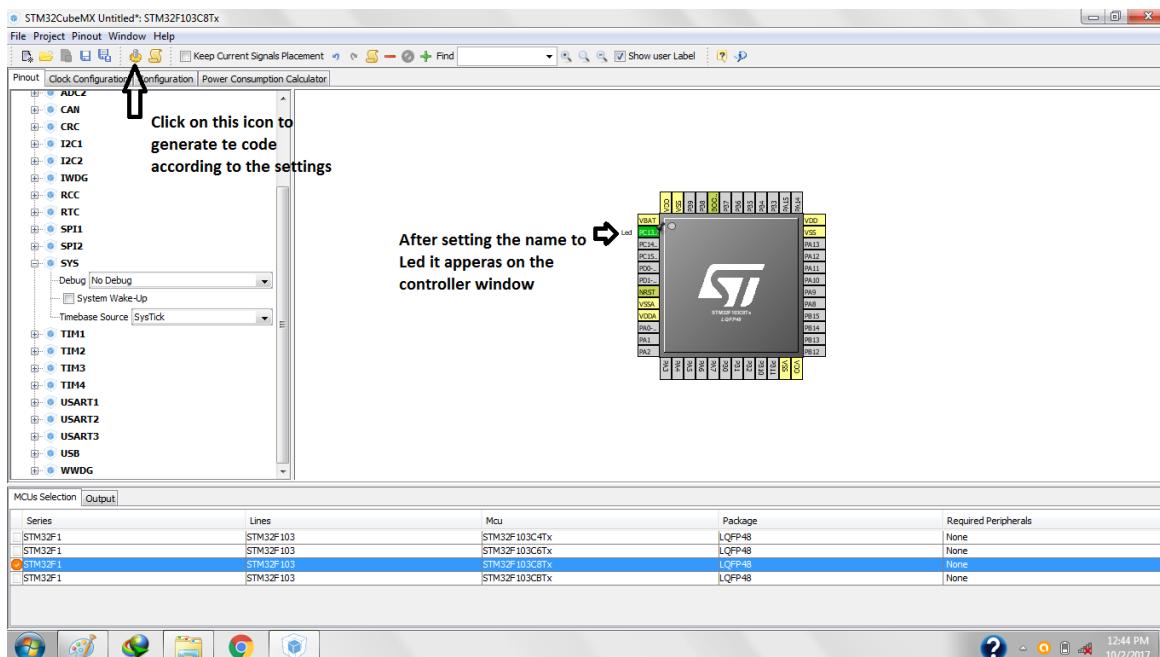
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */
```

## eBook Microcontroller



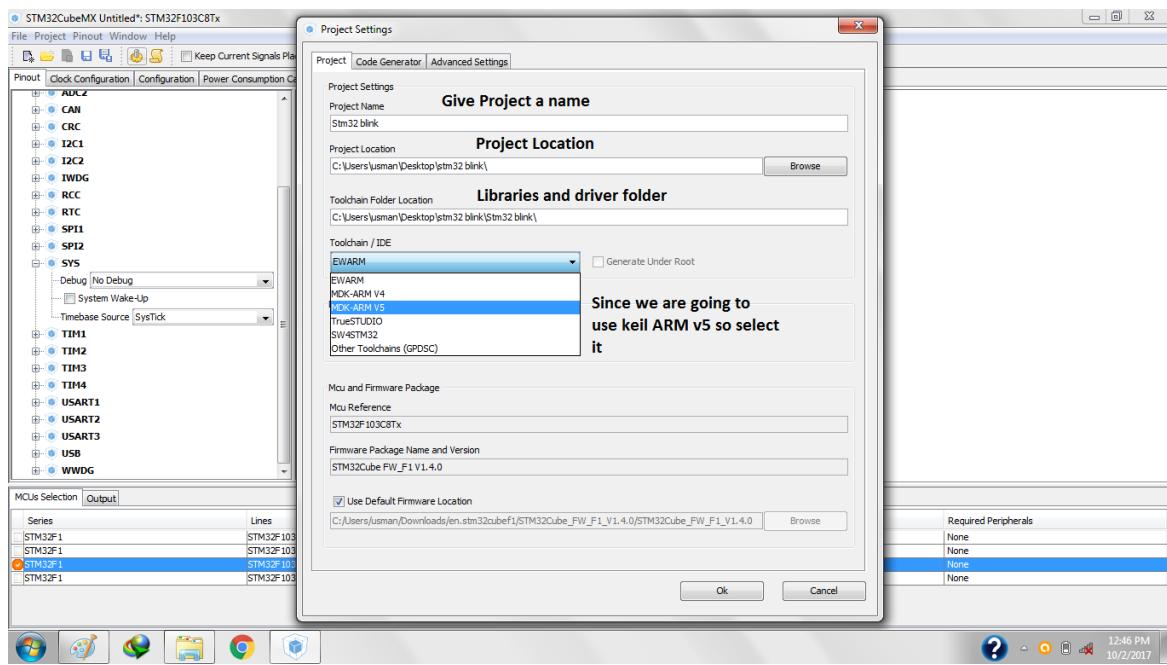
รูปที่ 1.6



รูปที่ 1.7

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

## eBook Microcontroller



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.HSISState = 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.AHCLKDivider = 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 */

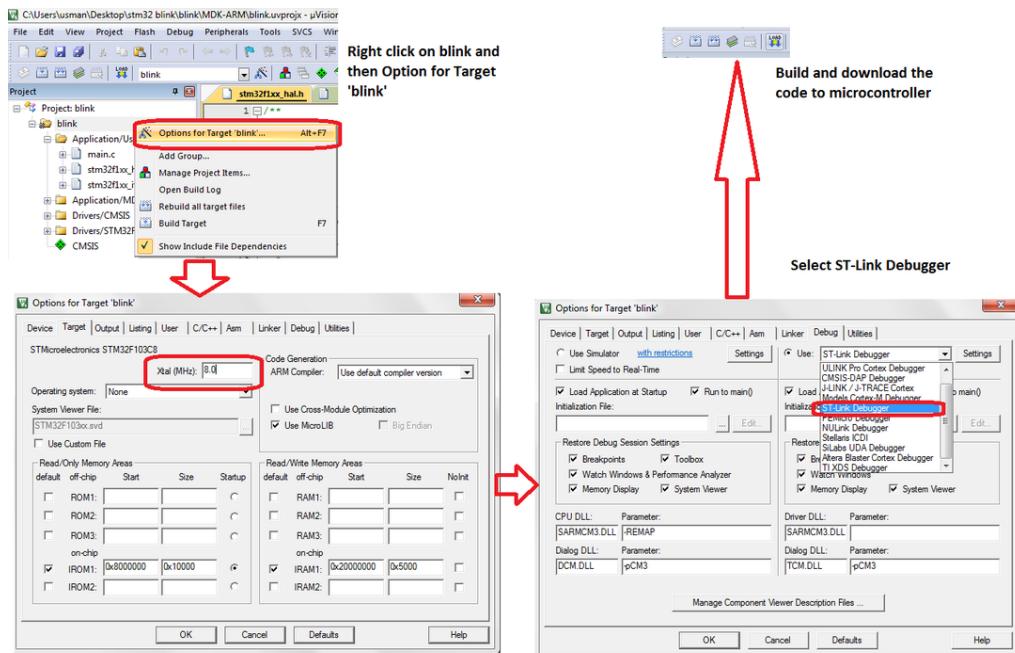
/**
 * @brief This function is executed in case of error occurrence.
 * @param file: The file name as string.
 * @param line: The line in file as a number.
 * @retval None
 */
void _Error_Handler(char *file, int line)
{
/* USER CODE BEGIN Error_Handler_Debug */
/* User can add his own implementation 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 Upload code to stm32 microcontroller

Since the development board has no on board 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.



กูเกิล 1.9

```

C:\2017-data\STM32\LED_Blink\MDK-ARM\LED_Blink.uvproj - aVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project LED_Blink
  Application/MDK-ARM
  Application/User
    stm32f1xx_hal_msp.c
    stm32f1xx_it.c
    main.c
  Drivers\CMSIS
  Drivers/STM32F1xx_HAL_Drv
  CMSIS
main.c
92  /* Initialize all configured peripherals */
93  MX_GPIO_Init();
94  /* USER CODE BEGIN 2 */
95
96  /* USER CODE END 2 */
97
98  /* Infinite loop */
99  /* USER CODE BEGIN WHILE */
100
101  while (1)
102
103
104  /* USER CODE END WHILE */
105  HAL_GPIO_TogglePin(LED_GPIO_Port,LED_Pin);
106  HAL_Delay(500);
107  /* USER CODE BEGIN 3 */
108
109  }
110  /* USER CODE END 3 */
111
112 }
113
114 /**
115 * @brief System Clock Configuration

```

Build Output

```

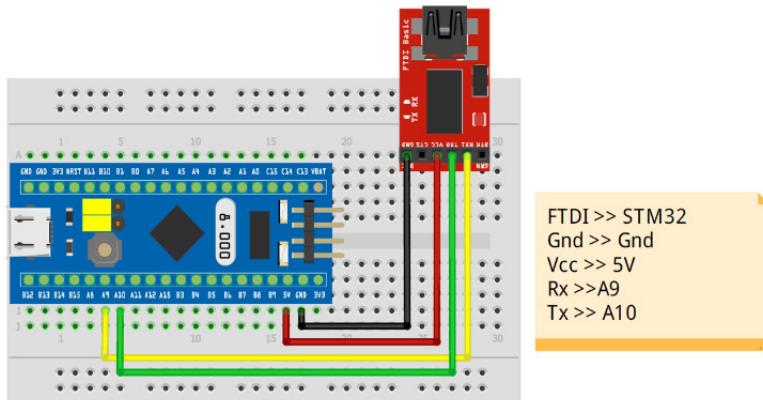
*** Using Compiler 'V5.06 update 5 (build 528)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'LED_Blink'
compiling main.c...
linking...
Program Size: Code=2600 RO-data=284 RW-data=16 ZI-data=1024
"LED_Blink\LED_Blink.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:02

```

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

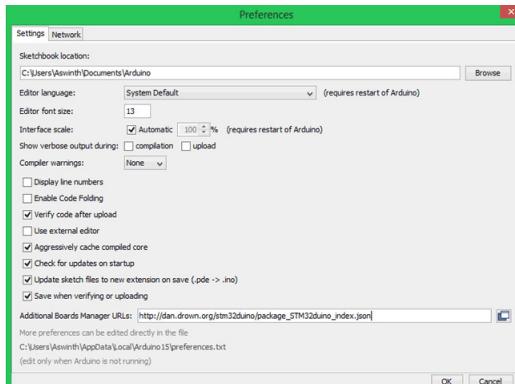
## eBook Microcontroller

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



รูปที่ 1.12: www.arduino.cc

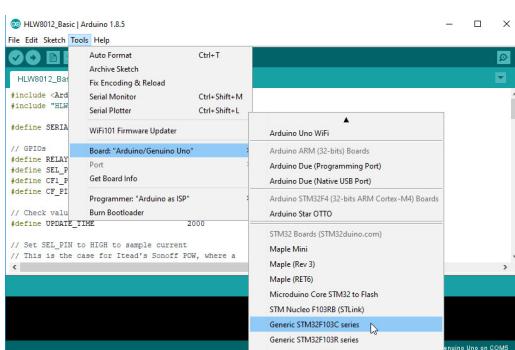
Step 2:- After Installing the Arduino IDE open and download the required packages for the STM32 board. This can be done by selecting File -> Preferences.



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](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

## 1.3 ไป้งการทดลอง INPUT/OUTPUT

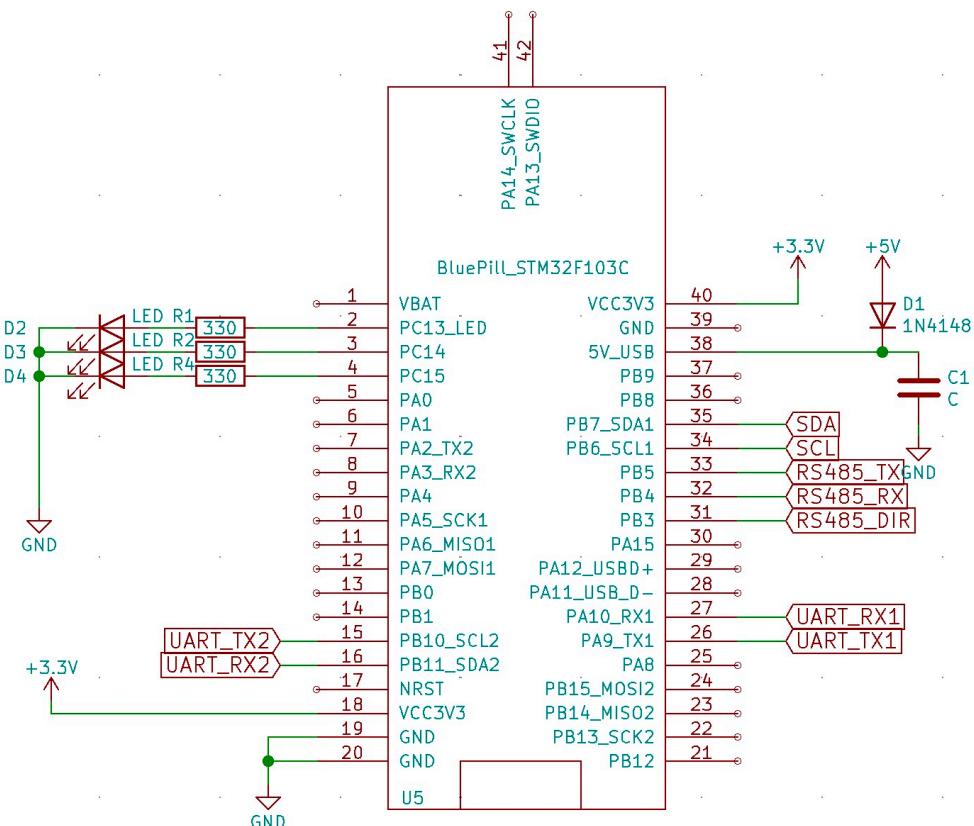
### 1.3.1 วัตถุประสงค์

1. Introduction
2. Hardwares
3. I/O structure
4. Create a LED Blink program
5. Conclusion
6. Reference
7. FAQ

### 1.3.2 อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ดทดลอง
2. STM32F103C8T6 บอร์ด
3. คอมพิวเตอร์ และอุปกรณ์ต่อพ่วง

### 1.3.3 วงจรการทดลอง Schematic



#### 1.3.4 โปรแกรมการทดลอง

---

```
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(PC13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
digitalWrite(PC13, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(PC13, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait PC13
}
```

---

1.3.5 ข้อมูลการทดลอง Experimental Data

---

---

---

---

---

---

---

---

---

---

---

---

---

1.3.6 ผลการทดลองและข้อเสนอแนะ Results and Conclusions

---

---

---

---

---

---

---

---

---

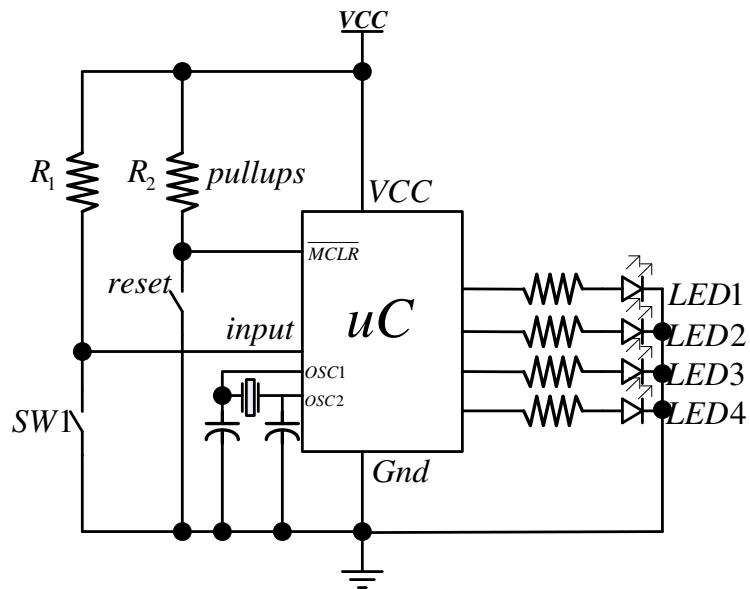
---

---

---

---

### 1.3.7 โจทย์การทำงานเพิ่มเติม



รูปที่ 1.16: PICXpress I/O 4 LED Flash Program

---

---

---

---

---

---

---

---

---

---

## 1.4 ใบงานการทดลองการสื่อสาร UART

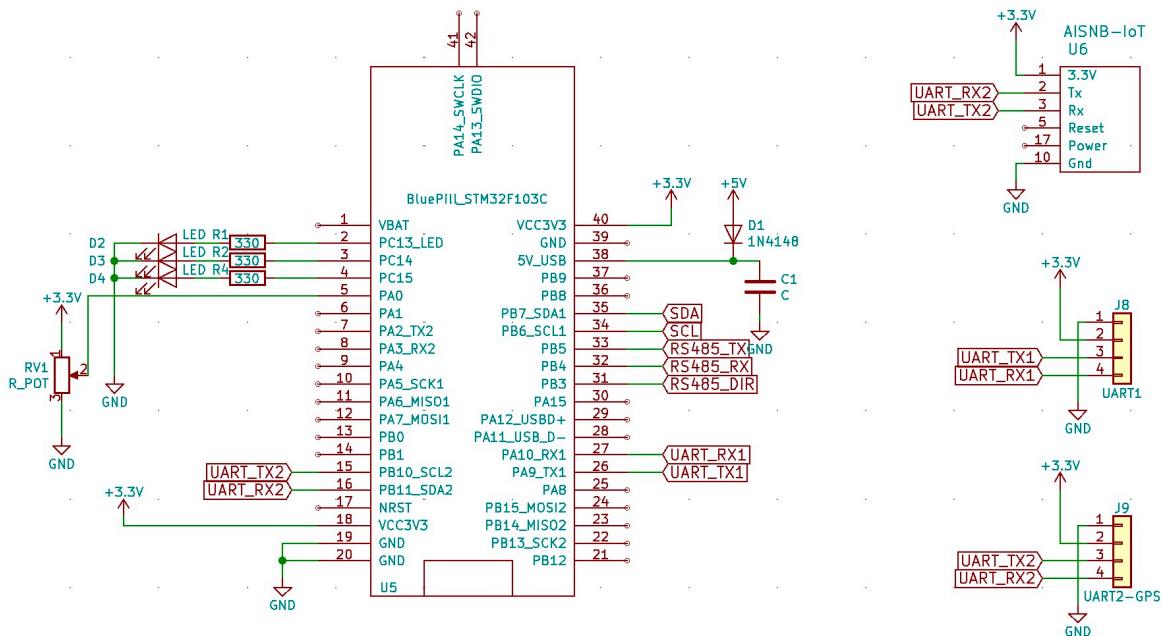
### 1.4.1 วัตถุประสงค์

1. Introduction
2. Hardwares
3. I/O structure
4. Create a UART program
5. Conclusion
6. Reference
7. FAQ

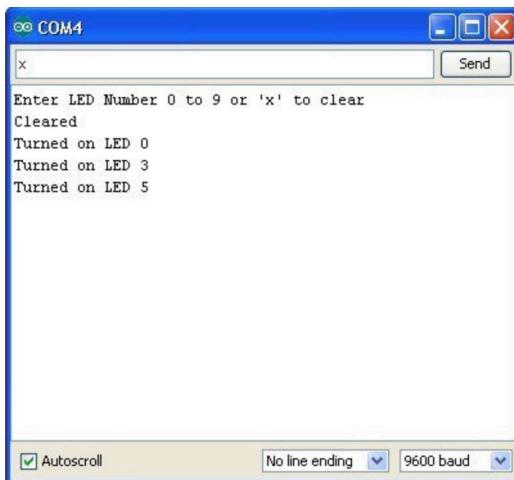
### 1.4.2 อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ดทดลอง
2. STM32F103C8T6 บอร์ด
3. คอมพิวเตอร์ และอุปกรณ์ต่อพ่วง

### 1.4.3 วงจรการทดลอง Schematic



รูปที่ 1.17: วงจรการต่อใช้งาน STM32F103 กับ ADC ส่งข้อมูลผ่าน UART



รูปที่ 1.18: วงศารการต่อใช้งาน STM32F103 ดูข้อมูลผ่าน UART serial monitor

#### 1.4.4 โปรแกรมการทดลอง

---

```
void setup() {
    Serial.begin(9600);
    Serial1.begin(9600);
}

void loop() {
    if (Serial.available()) { // If anything comes in Serial (USB),
        Serial1.write(Serial.read()); // read it and send it out Serial1 (pins 0 & 1)
    }

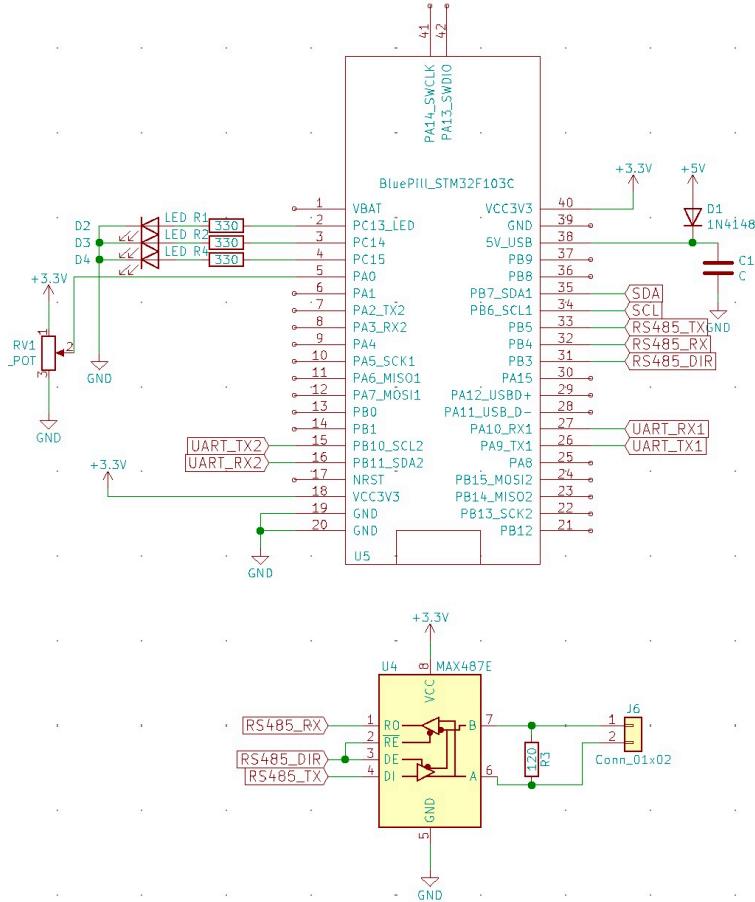
    if (Serial1.available()) { // If anything comes in Serial1 (pins 0 & 1)
        Serial.write(Serial1.read()); // read it and send it out Serial (USB)
    }
}
```

---



รูปที่ 1.19: การเขียนโปรแกรมบน arduino IDE

## 1.5 การทดลองการสื่อสาร UART ชนิด RS485



รูปที่ 1.20: การทดลองการสื่อสารโดยใช้ modbus

### 1.5.1 ข้อมูลการทดลอง Experimental Data

### 1.5.2 ผลการทดลองและข้อเสนอแนะ Results and Conclusions

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

### 1.5.3 โจทย์การทำงานเพิ่มเติม

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## 1.6 ใบงานการทดลอง ADC

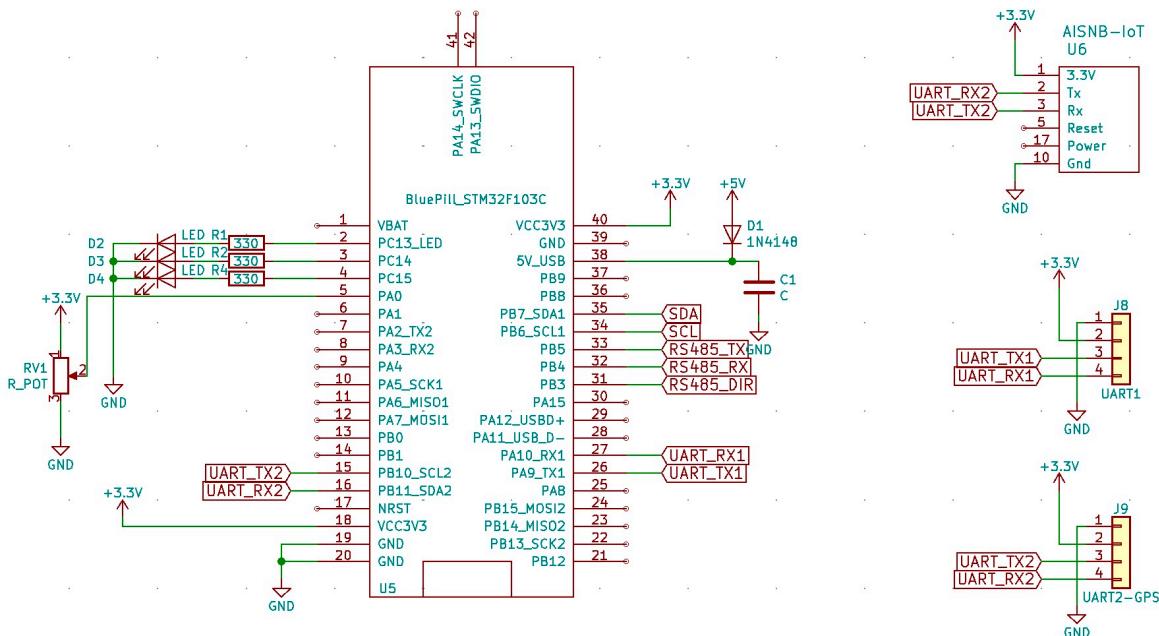
### 1.6.1 วัตถุประสงค์

1. Introduction
2. Hardwares
3. I/O structure
4. Create a POT ADC program
5. Conclusion
6. Reference
7. FAQ

### 1.6.2 อุปกรณ์ที่ใช้ในการทดลอง

1. บอร์ดทดลอง
2. STM32F103C8T6 บอร์ด
3. คอมพิวเตอร์ และอุปกรณ์ต่อพ่วง

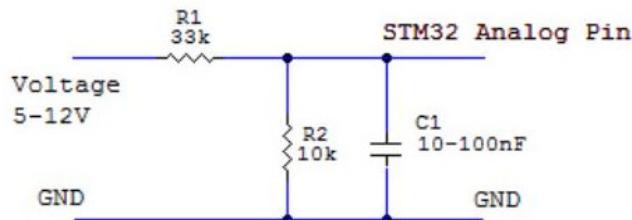
### 1.6.3 วงจรการทดลอง Schematic



รูปที่ 1.21: วงจรการต่อใช้งาน STM32F103 กับ ADC ส่งข้อมูลผ่าน UART

#### 1.6.4 การกำหนดค่าเพื่อใช้งาน ADC

การใช้งาน ADC ของ STM32F103 ในกรณีที่แรงดันเกินกว่าข้อกำหนดจะเป็นต้องมีวงจรแบ่งแรงดันเพื่อให้แรงดัน Input มีความหมายสมดังรูป 1.22



รูปที่ 1.22: วงจรแบ่งแรงดัน

#### 1.6.5 โปรแกรมการทดลอง

```

uint32_t adcin;           // data from ADC
float voltage, r_divider, r1, r2;

void setup() {
    delay(3000);          // wait on USB ready
    Serial.begin(115200);
    r1 = 33000.0;          // R1 in ohm, 33k = 33000.0
    r2 = 10000.0;          // R2 in ohm, 10k = 10000.0
    r_divider = r2 / (r1 + r2); // calculate the voltage divider ratio R2/(R1+R2)
}
void loop() {
    adcin = analogRead(PA0); // read ADC from PA0 or other PAx pin
    voltage = 3.3 / 4095 * adcin; // voltage at the PA0, ref voltage is 3.3V and 12bit ADC is 4095
    steps
    voltage = voltage / r_divider; // calculate the voltage at the divider's input
    Serial.println(voltage, 3); // Print out the voltage in Volts, 3 decimal places
    delay(1000); // wait 1 sec
}
    
```

Program ที่ 2 การอ่านค่าแรงดันไฟเลี้ยงตัว STM32F103

```

void setup_vdd_sensor() {
    adc_reg_map *regs = ADC1->regs;
    regs->CR2 |= ADC_CR2_TSVREFE; // enable VREFINT and temp sensor
    regs->SMPR1 = (ADC_SMPR1_SMP17 /* | ADC_SMPR1_SMP16 */); // sample rate for VREFINT ADC channel
}

void setup(){
    Serial.begin(9600);
}

void loop() {
    adc_enable(ADC1);
    vbat_int = 120 * 4096 / adc_read(ADC1, 17);
    adc_disable(ADC1);
    Serial.print("ADC=");Serial.println(ADC1);
    delay(5000);
}
    
```

1.6.6 ข้อมูลการทดลอง Experimental Data

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

1.6.7 ผลการทดลองและข้อเสนอแนะ Results and Conclusions

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

### 1.6.8 โจทย์การทำงานเพิ่มเติม

---

---

---

---

---

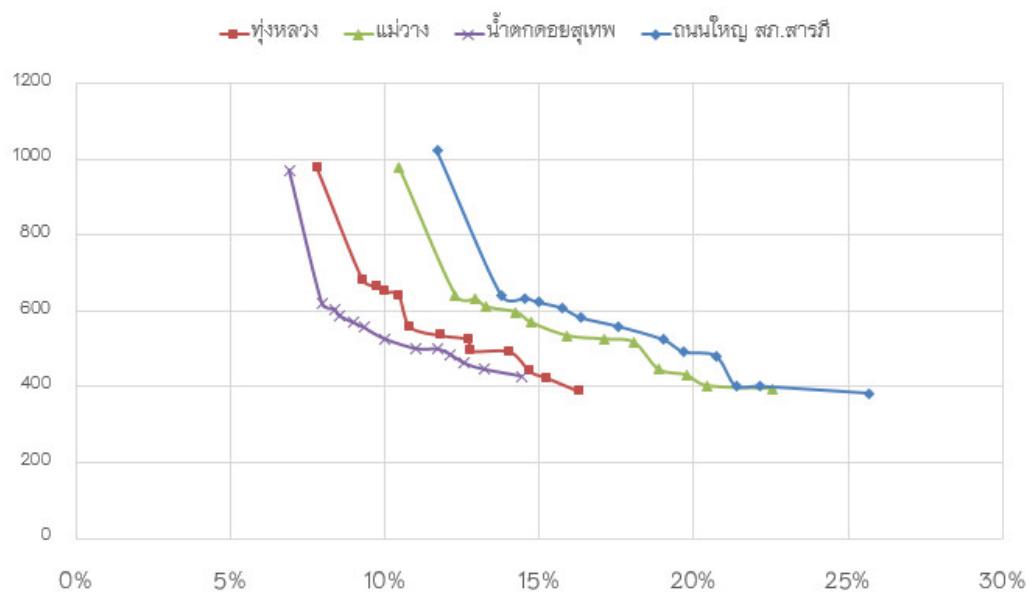
---

---

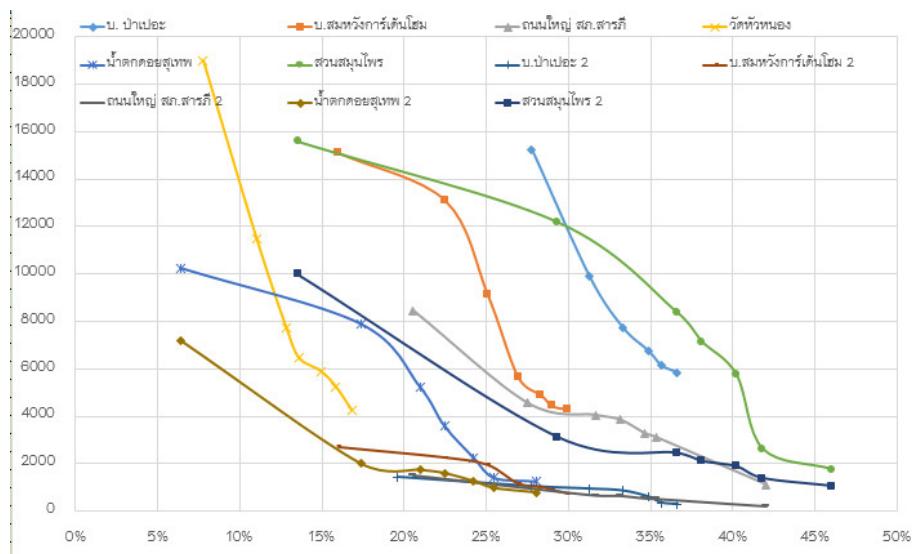
---

---

---



รูปที่ 1.23: Soilmoisture in field test Capacitive sensor [Worrajak:08/2018]



รูปที่ 1.24: Soil moisture in field test Resistive sensor [Worrajak:08/2018]

รูปที่ 1.25: PICXpress I/O Connection

## 1.7 ໃບງານກារທດລອງ PWM

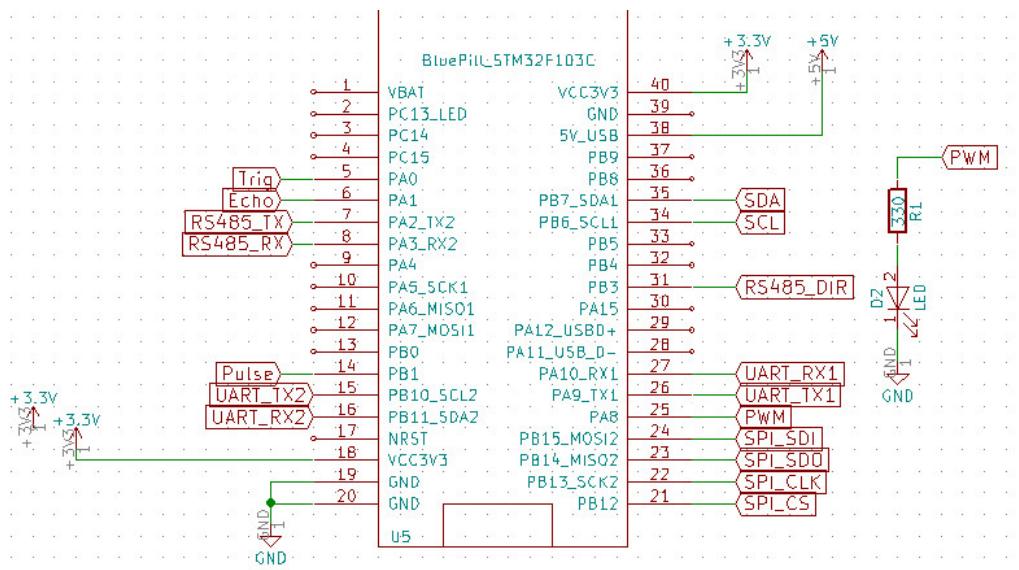
### 1.7.1 ວັດຖະສົງ

1. Introduction
2. Hardwares
3. I/O structure
4. Create a PWM with LED
5. Conclusion
6. Reference
7. FAQ

### 1.7.2 ອຸປກຣນີ້ໃຊ້ໃນການທດລອງ

1. ບ່ອຮົດທດລອງ
2. STM32F103C8T6 ບ່ອຮົດ
3. ຄອມພິວເຕອີ່ ແລະ ອຸປກຣນີຕ່ອື່ພ່ວງ

### 1.7.3 ວັດຈະການທດລອງ Schematic

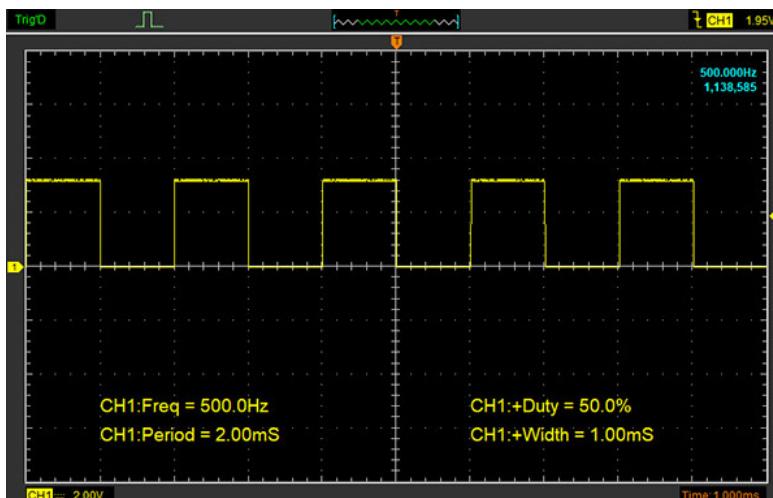


ຮູບທີ 1.26: ວັດຈະການຕ່ອທດລອງບັນບ່ອຮົດ STM32

PWM ຍ່ອມາຈາກ Pulse–Width Modulation ຕີ້ອກາຮສ້າງສັງຍານຮູບປັບເໜີ່ມ (Pulse) ແລະ ເຂົ້າຮ້າສັງຍານດ້ວຍຄວາມກວ້າງຂອງ Pulse + ໄລາຍໆຄົນທີ່ເລີ່ມ Arduino ເຊື່ວ່າເຄຍໃຫ້ກັນມາແລ້ວ ອາຈະໃຫ້ຄວບຄຸມຄວາມເຮົວມອເທົ່ອບັງ ທີ່ອສັ່ງຕຳແໜ່ງ Servo ອີ່ໂສ ສະເໝົາ

ການໃໝ່ງານ PWM ບັນ Arduino ທົ່ວໆໄປ ດ້ວຍຢ່າງນີ້ເວີ່ມຈາກການກຳໜັດ pinMode ໃຫ້ເປັນ output (pin ທີ່ຈະໃໝ່ງານ PWM ຕ້ອງມີສັງລັກຂ່າຍ) ໃນ ຕ້ວອຍ່າງໃໝ່ເປັນ D3 ແລະ ກຳໜັດ duty ດ້ວຍ ຕໍາສັ່ງ analogWrite ຜຶ່ງຄາທີ່ກຳໜັດຕະຫຼາງ 0–255 ນີ້ໜໍາຍຄວາມວ່າກໍ່າຕ້ອງການ duty 50

```
pinMode(3,OUTPUT);
analogWrite(3,127);
```



รูปที่ 1.27: สัญญาณที่สร้างโดย PWM

#### 1.7.4 การใช้งาน PWM บน STM32 arduino

สำหรับบอร์ด STM32 บน Arduino โดยทั่วไปนั้นสามารถเรียกใช้งานได้เหมือนบอร์ด Arduino แต่ pin ที่จะใช้เป็น PWM ต้องดูจากคู่มือของบอร์ดหรือเอกสารของซิพรุ่นที่ใช้ แต่ในบทความนี้จะมาดูว่า PWM สร้างขึ้นมาจากหลักการอะไรและการกำหนดรูปแบบของ PWM บน STM32 ทำได้อย่างไร จะอยู่ในหัวข้อถัดไป

```
void setup()
{
    pinMode(PA8,PWM);
    pwmWrite(PA8,32767);
}
```

ความแตกต่างระหว่างการใช้คำสั่ง digitalWrite กับ pwmWrite สำหรับ STM32 บน Arduino คือ pwmWrite จะกำหนดค่า duty ให้ตั้งแต่ 0 – 65535 เนื่องจาก Timer บน STM32 มีขนาด 16bit ต่างๆกัน แต่Timer บนบอร์ด Arduino ที่มีขนาด 8bit จะกำหนดค่าให้แค่ 0-255 ทำให้ STM32 สามารถกำหนดความละเอียดของ PWM ได้มากกว่า ส่วนคำสั่ง digitalWrite จะกำหนดค่า ช่วง 0-255 เมื่อกับบอร์ด Arduino แต่ Library ทำการ Map ค่า 0-255 เป็นค่า 0-65535 ให้เองเพื่อสะดวกในการใช้งานคำสั่ง Arduino

#### 1.7.5 Timer กับ PWM

Timer ถือเป็นส่วนสำคัญในการสร้างสัญญาณ PWM ซึ่งจะทำหน้าที่ในการเป็นตัวนับจำนวนตามสัญญาณนาฬิกาเริ่มจาก 0 ถึงค่าที่กำหนดไว้ซึ่งจะเรียกว่า Overflow

ตัวแปลล์ต่างๆในการกำหนดรูปแบบ PWM ประกอบด้วย

PrescaleFactor : ตัวหารความถี่ Overflow : กำหนดขนาด PWM (16bit) Frequency : ความถี่ PWM Period : ความยาวของ 1 ช่องสัญญาณ (1 Cycle)

จากตัวอย่างการเรียกใช้ PWM แบบง่าย จะถูกกำหนด ค่า PrescaleFactor และค่า Overflow ให้ซึ่งจะคำนวณได้ดังนี้

```
SystemClockFrequency      = 72000000
PrescaleFactor            = 2
Overflow                  = 65535
Frequency                 = SystemClockFrequency / PrescaleFactor * Overflow
                           (
                           = 72000000 / (2 * 65535)
                           = 549.32 Hz
                           = 1 / Frequency = 1.82e-3 = 1.82 ms.
```

รูปที่ 1.28: การกำหนดความถี่ PWM

หลังจากที่รู้หลักการคำนวณก็สามารถแทนค่าต่างๆในสูตรเพื่อกำหนดรูปแบบ PWM ตามที่ต้องการ เช่น หากต้องการ PWM ขนาด 20ms มี

เอกสารประกอบการสอน วิชาปฏิบัติไมโครคอนโทรลเลอร์ รวมโดย อาจารย์วรวิจาร์ เมืองใจ หน้าที่ 28-??

duty เท่ากับ 10% หรือ 2ms สามารถกำหนดได้ดังนี้

```
SystemClockFrequency = 72000000
Period = 20ms = 0.02
Frequency = 1 / 0.02 = 50 Hz
PrescaleFactor = 22 /*ค่ากำหนดเวลาโดยใช้ค่าที่ทำให Overflow ไม่เกิน 65535*/
Overflow = SystemClockFrequency / ( Frequency * PrescaleFactor )
= 72000000 / ( 50 * 22 )
= 65454

Duty = 65454 * 10 / 100 /*duty = 10%*/
= 6545
```

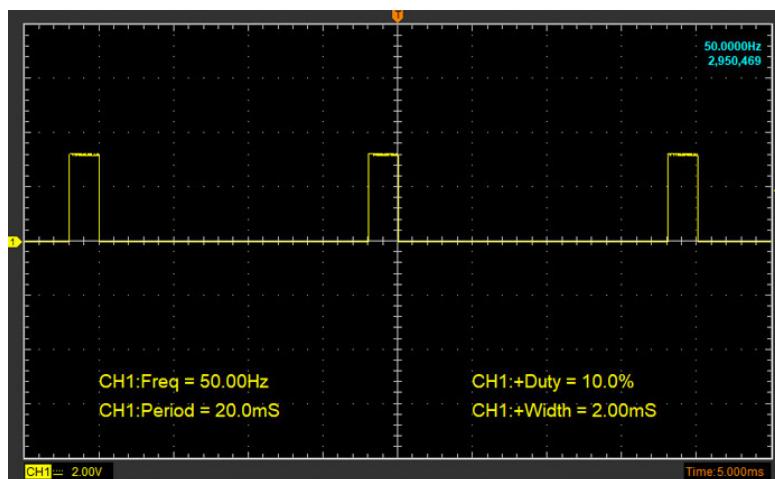
รูปที่ 1.29: การกำหนด duty PWM

### 1.7.6 โปรแกรมการทดลอง Main.c



```
File Edit Sketch Tools Help
sketch_sep18b §
void setup() {
pinMode(PA8,PWM);
Timer1.setPrescaleFactor(22);
Timer1.setOverflow(65454);
pwmWrite(PA8,6545);
}
```

รูปที่ 1.30: ตัวอย่างโปรแกรม duty PWM



รูปที่ 1.31: รูปคลื่นโปรแกรมการทดลอง duty PWM

1.7.7 ข้อมูลการทดลอง Experimental Data

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

1.7.8 ผลการทดลองและข้อเสนอแนะ Results and Conclusions

---

---

---

---

---

---

---

---

---

---

---

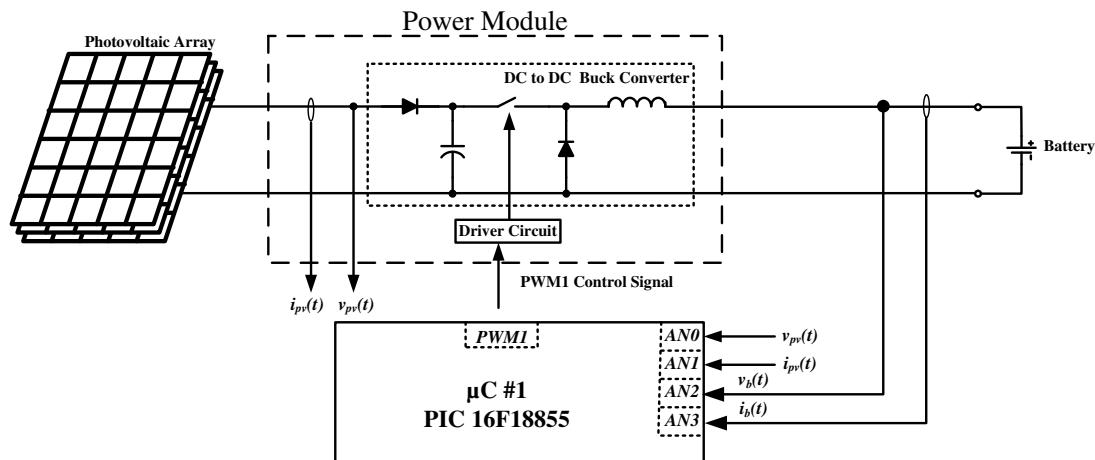
---

---

---

---

### 1.7.9 จิทย์การทำงานเพิ่มเติม



รูปที่ 1.32: การใช้งาน PWM เพื่อสร้าง Module Buck Converter

## 1.8 ໃບງານກារທດລອງ Interrupts

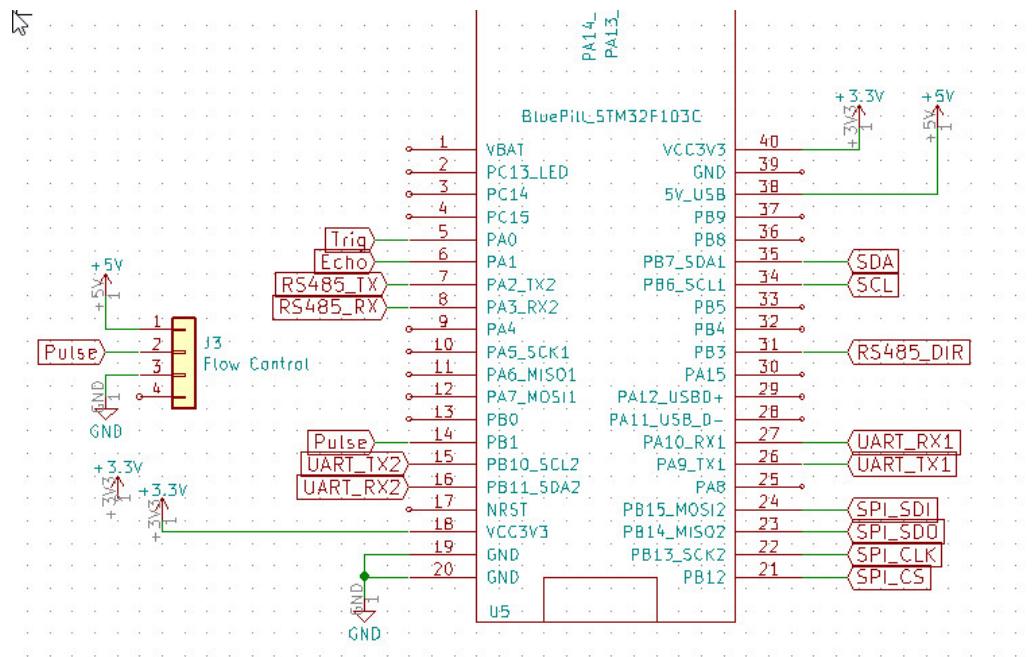
### 1.8.1 ວັດຖະສົງ

1. Introduction
2. Hardwares
3. I/O structure
4. Create a Interrupts switch, interrupt Timer with LED
5. Conclusion
6. Reference
7. FAQ

### 1.8.2 ອຸປກຮນທີ່ໃຊ້ໃນການທດລອງ

1. ບ່ອນທດລອງ
2. STM32F103C8T6 ບ່ອນ
3. ຄອມພິວເຕອີ່ ແລະ ອຸປກຮນຕ່ອື່ພ່ວງ

### 1.8.3 ວັດຈາກການທດລອງ Schematic



ຮູບທີ 1.33: ວັດຈາກການທດລອງ STM32

### 1.8.4 Digital Pins With Interrupts

The first parameter to attachInterrupt() is an interrupt number. Normally you should use digitalPinToInterrupt(pin) to translate the actual digital pin to the specific interrupt number. For example, if you connect to pin 3, use digitalPinToInterrupt(3) as the first parameter to attachInterrupt().

## Digital Pins With Interrupts

The first parameter to `attachInterrupt()` is an interrupt number. Normally you should use `digitalPinToInterrupt(pin)` to translate the actual digital pin to the specific interrupt number. For example, if you connect to pin 3, use `digitalPinToInterrupt(3)` as the first parameter to `attachInterrupt()`.

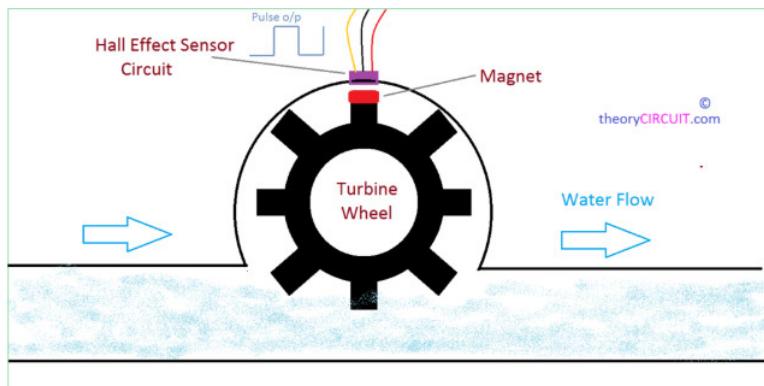
BOARD	DIGITAL PINS USABLE FOR INTERRUPTS
Uno, Nano, Mini, other 328-based	2, 3
Uno WiFi Rev.2	all digital pins
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR Family boards	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due	all digital pins
101	all digital pins (Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with CHANGE)

รูปที่ 1.34: ขات่างๆ ของ MCU ที่สามารถใช้งาน Interrupt ได้

### 1.8.5 การใช้งาน Interrupts บน STM32 arduino

สำหรับบอร์ด STM32 บน Arduino โดยทั่วไปนั้นสามารถเรียกใช้งานได้เหมือนบอร์ด Arduino ชาใช้งาน Interrupt จะสามารถใช้งานได้ทุกขา GPIO

ในการทดลองนี้จะเป็นการต่อเพื่อวัดค่าสัญญาณพลังที่มาจาก flow sensor ดังรูป 1.35 หลังจากนั้นจึงนำมาเปรียบเทียบกับช่วงเวลาที่ทำการรับค่า



รูปที่ 1.35: ขات่างๆ ของ MCU ที่สามารถใช้งาน Interrupt ได้

### 1.8.6 โปรแกรมการทดลอง Main.c

---

```

byte sensorInterrupt = 0;
byte sensorPin    = PB1;
float calibrationFactor = 0.24;
volatile byte pulseCount;
float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;
int total1;
unsigned long oldTime;

void pulseCounter() {
    pulseCount++;
}

void readData(){
    detachInterrupt(sensorInterrupt);
    flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) * calibrationFactor;
    oldTime = millis();
    flowMilliLitres = (flowRate / 60) * 1000;
    totalMilliLitres += flowMilliLitres;
    total1 = totalMilliLitres;
    pulseCount = 0;
    attachInterrupt(sensorPin, pulseCounter, FALLING);

    Serial.print("flowrate ");Serial.println(flowRate);Serial.print(" mL/min. ");
}

void setup() {
    pinMode(sensorPin, INPUT);
    digitalWrite(sensorPin, HIGH);
    Serial.begin(9600);

    pulseCount      = 0;
    flowRate        = 0.0;
    flowMilliLitres = 0;
    totalMilliLitres = 0;
    oldTime         = 0;
    attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
}

void loop() {
    // put your main code here, to run repeatedly:
    readData();
    delay(5000);
}

```

---

1.8.7 ข้อมูลการทดลอง Experimental Data

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

1.8.8 ผลการทดลองและข้อเสนอแนะ Results and Conclusions

---

---

---

---

---

---

---

---

---

---

---

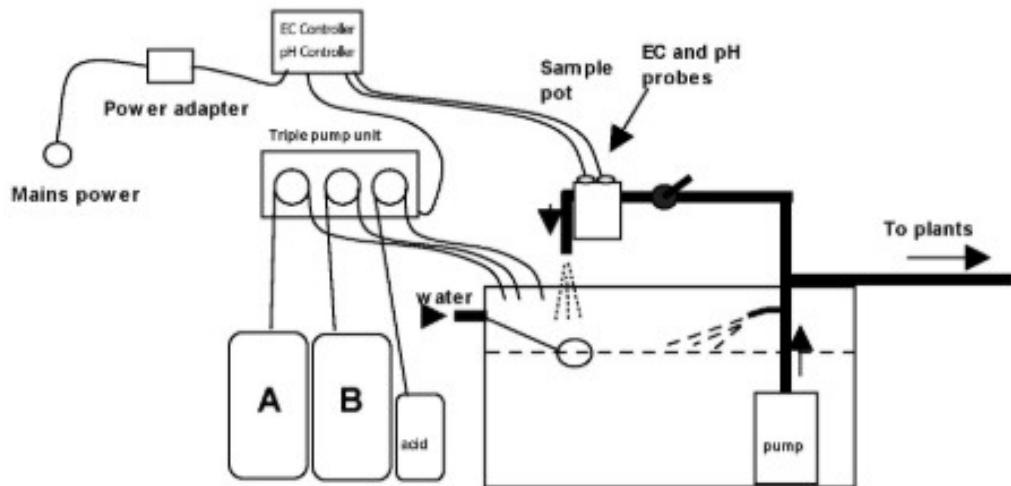
---

---

---

---

### 1.8.9 โจทย์การทำงานเพิ่มเติม



รูปที่ 1.36: การผสมปุ๋ย A B อัตโนมัติ

---

---

---

---

---

---

---

---

---

---

---

## 1.9 ใบงานการทดสอบ SPI I2C

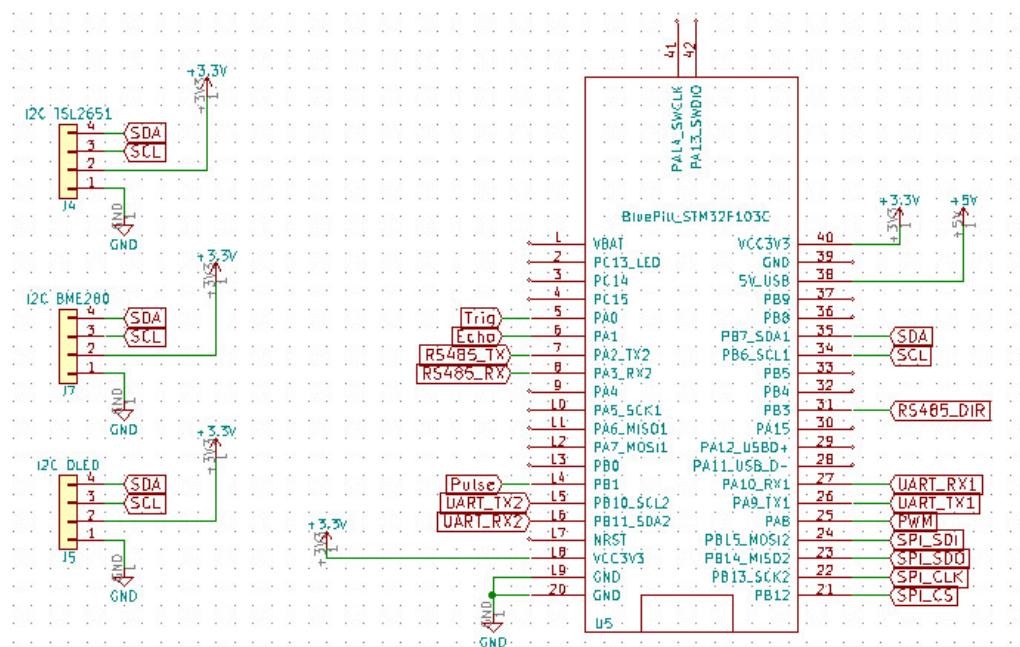
### 1.9.1 ວັດທະນາປະສົງຄໍ

1. Introduction
  2. Hardwares
  3. I/O structure
  4. Create a SPI I2C with OLED, Real Time Clock and Temperature Sensor program
  5. Conclusion
  6. Reference
  7. FAQ

### 1.9.2 อุปกรณ์ที่ใช้ในการทดสอบ

1. บอร์ดทดลอง
  2. จอ OLED ชนิด SPI I2C
  3. STM32 Board
  4. คอมพิวเตอร์ และอุปกรณ์ต่อพ่วง

### 1.9.3 ວິຊາການທົມຄອງ Schematic |2C



รูปที่ 1.37: การต่อวงจรเพื่อทดสอบการใช้งาน I2C



รูปที่ 1.38: I2C ที่ใช้งานกับเซนเซอร์ต่างๆ

#### 1.9.4 โปรแกรมการทดลอง

```
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <Arduino.h>

#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <SSD1306Ascii.h>
#include <SSD1306AsciiWire.h>
#include "TSL2561.h"

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C #define BME280_ADDRESS (0x76)

TSL2561 tsl(TSL2561_ADDR_FLOAT);
```

```

#define I2C_ADDRESS 0x3C
#define RST_PIN -1
SSD1306AsciiWire oled;

float temperature = 0;
float humidity = 0;
uint16_t tempC_int = 0;
uint8_t hum_int = 0;
uint16_t lux_int = 0;

int vbat_int;

void readData(){
    uint32_t lum = tsl.getFullLuminosity();
    uint16_t ir, full;

    ir = lum >> 16;
    full = lum & 0xFFFF;
    lux_int = tsl.calculateLux(full, ir);

    temperature = bme.readTemperature();
    humidity = bme.readHumidity();
    tempC_int = temperature*10;
    hum_int = humidity*2;
    adc_enable(ADC1);
    vbat_int = 120 * 4096 / adc_read(ADC1, 17);
    adc_disable(ADC1);

    oled.clear();
    oled.println("Sensor Test");
    oled.setCursor(0,2);
    oled.print(temperature,1);oled.print("C ");oled.print(humidity,1);oled.print("%");
    oled.setCursor(0,3);
    oled.print("lux:");oled.print(lux_int);oled.print(" Batt:");oled.print(vbat_int*0.01);

#ifdef DEBUG
Serial.print(temperature);Serial.print(" ");Serial.print(humidity);Serial.print(" ");
    " );Serial.print(vbat_int*0.01);Serial.print(" ");Serial.println(tsl.calculateLux(full, ir));
    Serial.println();
#endif
}

void setup_vdd_sensor() {
    adc_reg_map *regs = ADC1->regs;
    regs->CR2 |= ADC_CR2_TSVREFE; // enable VREFINT and temp sensor
    regs->SMPR1 = (ADC_SMPR1_SMP17 /* | ADC_SMPR1_SMP16 */); // sample rate for VREFINT ADC channel
}

void setup() {
    setup_vdd_sensor();
    Serial.begin(9600);

    if (tsl.begin()) {
        Serial.println("Found TLS2561 sensor");
    } else {
        Serial.println("No TLS2561 sensor?");
        while (1);
    }

    tsl.setGain(TSL2561_GAIN_16X); // set 16x gain (for dim situations)
}

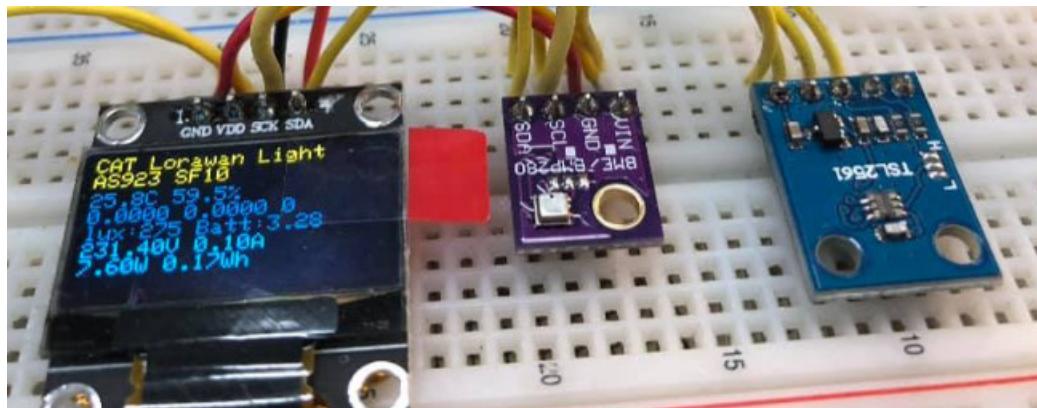
```

```
tsl.setTiming(TSL2561_INTEGRATIONTIME_13MS); // shortest integration time (bright light)

bool status = bme.begin();
if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
}
#if RST_PIN >= 0
oled.begin(&Adafruit128x64, I2C_ADDRESS, RST_PIN);
#else // RST_PIN >= 0
oled.begin(&Adafruit128x64, I2C_ADDRESS);
#endif // RST_PIN >= 0
oled.setFont(Adafruit5x7);
}

void loop() {
    readData();
    delay(5000);
}
```

---



รูปที่ 1.39: การต่อวงจรเพื่อทดลองการใช้งาน I2C

1.9.5 ข้อมูลการทดลอง Experimental Data

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

1.9.6 ผลการทดลองและข้อเสนอแนะ Results and Conclusions

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

### 1.9.7 โจทย์การทำงานเพิ่มเติม

---



---



---



---



---



---



---

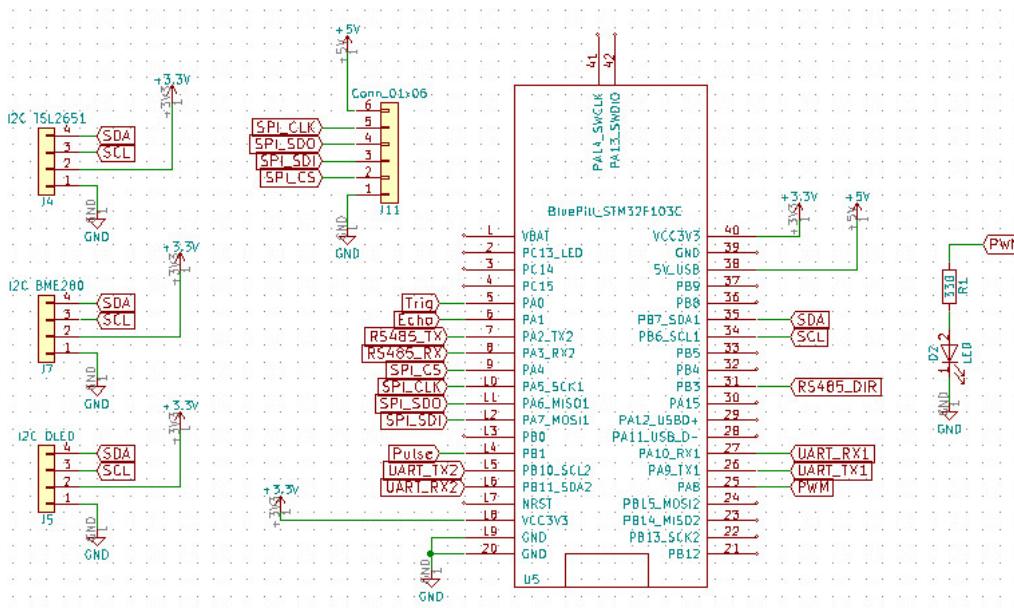


---



---

### 1.9.8 วงจรการทดลอง Schematic SPI

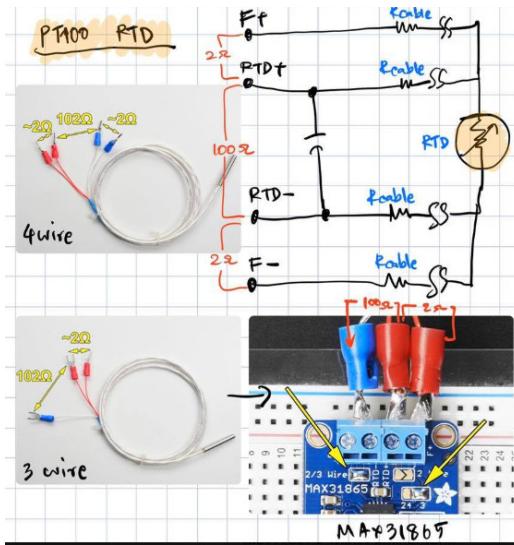


รูปที่ 1.40: การต่อวงจรเพื่อทดลองการใช้งาน SPI

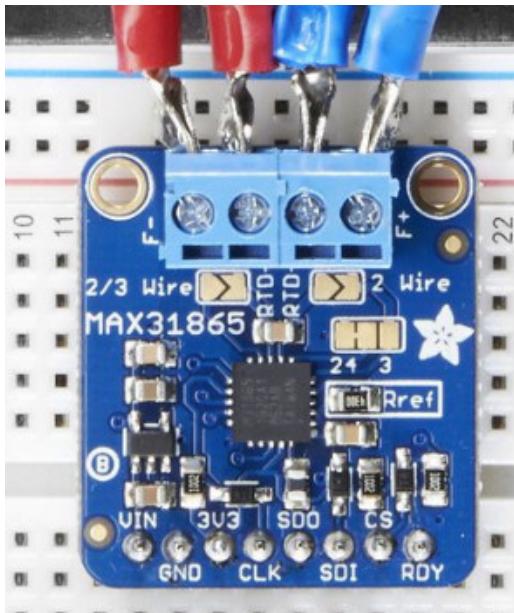
For precision temperature sensing, nothing beats a Platinum RTD. Resistance temperature detectors (RTDs) are temperature sensors that contain a resistor that changes resistance value as its temperature changes, basically a kind of thermistor. In this sensor, the resistor is actually a small strip of Platinum with a resistance of 100 or 1000 ohms at 0°C, thus the name PT100/PT1000. Compared to most NTC/PTC thermistors, the PT type of RTD is much more stable and precise (but also more expensive) PT's have been used for many years to measure temperature in laboratory and industrial processes, and have developed a reputation for accuracy (better than thermocouples), repeatability, and stability.

However, to get that precision and accuracy out of your PT100x RTD you must use an amplifier that is designed to read the low resistance. Better yet, have an amplifier that can automatically adjust and compensate for the resistance of the connecting wires. If you're looking for a great RTD sensor, today is your lucky day because we have a lovely Adafruit RTD Sensor Amplifier with the MAX31865 sensor.

1.41 1.42



รูปที่ 1.41: การต่อวงจรการใช้งาน SPI กับ IC MAX31865 เพื่อวัดค่าอุณหภูมิจากหัววัด PT100



รูปที่ 1.42: IC MAX31865 เพื่อวัดค่าอุณหภูมิจากหัววัด PT100

### 1.9.9 โปรแกรมการทดลอง SPI MAX31865 PT100

```
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <Arduino.h>

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <SSD1306Ascii.h>
```

```

#include <SSD1306AsciiWire.h>
#include <Adafruit_MAX31865.h>

#define SPI_NSS_PIN PA4
SPIClass SPI_2(2);

Adafruit_MAX31865 max = Adafruit_MAX31865(PA4);

#define RREF    430.0
// The 'nominal' 0-degrees-C resistance of the sensor
// 100.0 for PT100, 1000.0 for PT1000
#define RNOMINAL 100.0

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C #define BME280_ADDRESS

#define I2C_ADDRESS 0x3C
#define RST_PIN -1
SSD1306AsciiWire oled;

float temperature = 0;
float humidity = 0;
uint16_t tempC_int = 0;
uint8_t hum_int = 0;
uint16_t lux_int = 0;
int vbat_int;

void setup_vdd_sensor() {
    adc_reg_map *regs = ADC1->regs;
    regs->CR2 |= ADC_CR2_TSVREFE; // enable VREFINT and temp sensor
    regs->SMPR1 = (ADC_SMPR1_SMP17 /* | ADC_SMPR1_SMP16 */); // sample rate for VREFINT ADC channel
}

float extT;

void readData(){

    uint16_t rtd = max.readRTD();
    extT = max.temperature(RNOMINAL, RREF);
    //Serial.print("RTD value: "); Serial.println(rtd);
    float ratio = rtd;
    ratio /= 32768;
    //Serial.print("Ratio = "); Serial.println(ratio,8);
    //Serial.print("Resistance = "); Serial.println(RREF*ratio,8);

    temperature = bme.readTemperature();
    humidity = bme.readHumidity();
    adc_enable(ADC1);
    vbat_int = 120 * 4096 / adc_read(ADC1, 17);
    adc_disable(ADC1);

    oled.clear();
    oled.println("PT100 Demo");
    oled.setCursor(0,2);
    oled.print("T:");oled.print(temperature,1);oled.print(" RH:");oled.print(humidity,1);
    oled.setCursor(0,4);oled.print("TR Temp:");oled.print(extT,2);

    Serial.print(temperature);Serial.print(" ");
}

```

```

Serial.print(humidity);Serial.print(" ");
Serial.print(vbat_int);Serial.print(" ");
Serial.print(extT,2);Serial.print(" ");
}

void setup(){
  Serial.begin(9600);
  delay(1000);

  SPI.begin();
  pinMode(SPI_NSS_PIN, OUTPUT);

  max.begin(MAX31865_3WIRE);

  bool status = bme.begin();
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  #if RST_PIN >= 0
  oled.begin(&Adafruit128x64, I2C_ADDRESS, RST_PIN);
  #else // RST_PIN >= 0
  oled.begin(&Adafruit128x64, I2C_ADDRESS);
  #endif // RST_PIN >= 0

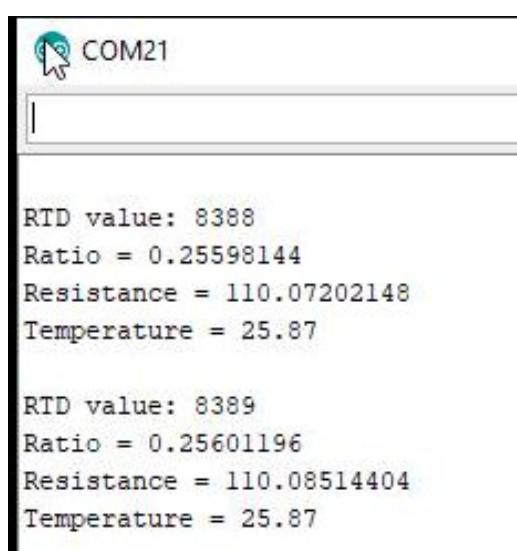
  oled.setFont(Adafruit5x7);
}

unsigned long delTime = 10000;

void loop()
{
  readData();
  delay(5000);
}

```

---



รูปที่ 1.43: IC MAX31685 ค่าadcการวัดอุณหภูมิจากหัววัด PT100

1.9.10 ข้อมูลการทดลอง Experimental Data

---

---

---

---

---

---

---

---

---

---

---

---

---

1.9.11 ผลการทดลองและข้อเสนอแนะ Results and Conclusions

---

---

---

---

---

---

---

---

---

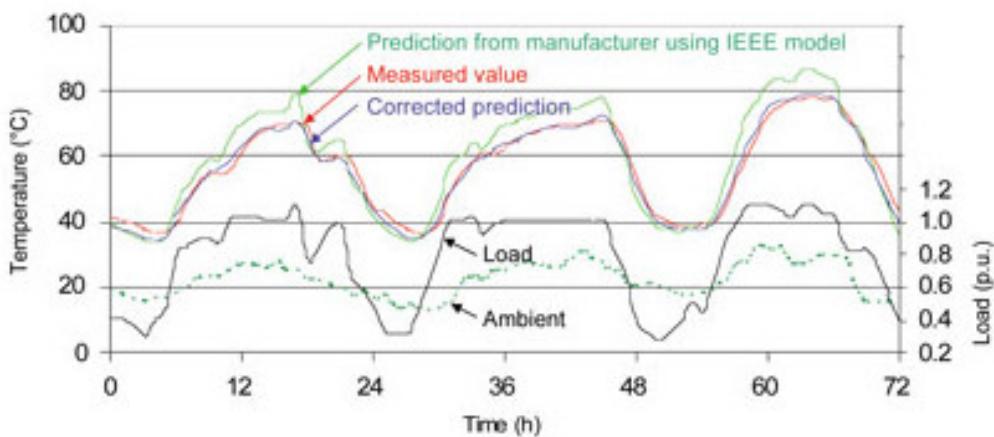
---

---

---

---

### 1.9.12 โจทย์การทำงานเพิ่มเติม



รูปที่ 1.44: Winding Hot-Spot Temperature Model

A better approach would be to calculate the economic benefit that would ensue from using the additional loading capability to take advantage of market opportunities. As a numerical example, assume that a 100MVA transformer could be requested to carry a 10% overload, for about 5% of the time, when the market condition are attractive. Assuming an ambient temperature of 30°C, the calculation of benefit, including allocation for loss of life would unfold as follows:

1 Transformer rated power (MVA)	100
2 Overloading margin made available by monitoring (%)	10
3 Probability of overloading opportunity (h/year)	450
4 Financial benefit from energy transmitted (\$/MWh)	80
5 Replacement cost of transformer (\$)	2,000,000
6 Transformer normal life duration at 110°C (h)	150,000
7 Assumed ambient temperature (°C)	30
8 Estimated winding hot-spot temperature (°C)	127
9 Aging acceleration factor during overload	5.5
 Yearly benefit from extra loading ( $1 \times 2 \times 3 \times 4$ )	\$360,000
Yearly cost for additional loss of life ( $5 \times 1/6 \times 3 \times 9$ )	\$33,000
 Net yearly benefit from overloading with on-line monitoring	\$327,000

รูปที่ 1.45: the calculation of benefit, including allocation for loss of life of transformer