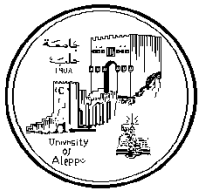
جامعة حلب
كلية الهندسة الكهربائية والالكترونية
قسم هندسة التحكم والأتمتة
مخبر التحكم



مقرر_____

الجلسة السادسة

السنة _____

2023⁄2202

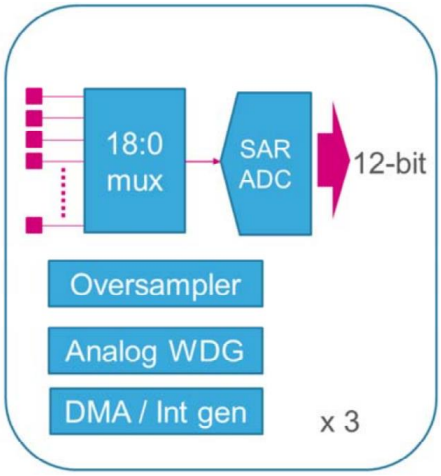
مدرس المقرر: _____

الغاية من الجلسة

1- التعرف على المبدلات التشابهية الرقمية في متحكماتSTM32
2- أنماط عمليات التحويل ADC conversion modes
3- طرق قراءة المبدل التشابهي الرقمي
4- التطبيق العملي الأول: التحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWM - من خلال مقاومة ضوئية مو صولة على أحد أقطاب الدخل الت شابهي با ستخدام نمط الـ Polling - با ستخدام البورد التطويري
5- التطبيق العملي الثاني: إعادة التطبيق السابق باستخدام نمط المقاطعة Interrupt
6- التطبيق العملي الثالث: مراقبة درجة حرارة الغرفة وعرضها على شاشة LCD

1- التعرف على المبدلات التشابهية الرقمية في متحكماتSTM32:

المبدلات الت شابهية الرقمية عبارة عن دارات الكترونية تقوم بتحويل الجهد الت شابهي على دخلها إلى قيمة رقمية بالنظام الثنائي مقابلة لم ستوى الجهد، فبمجرد قدح المبدل الت شابهي الرقمي يبدأ بأخذ العينات samples ويقوم بعملية تدعى التكميم ليقابل كل م ستوى من الجهد بما ينا سبه من القيم الرقمية.تحتوي متحكمات STM32G0 على م بدل ت شابهي رقمي وحيد من نوع Successive approximation ADC(SAR) بدقة 12بت وما يقارب الـ 19 قناة للمبدل



ت سمح المبدلات الت شابهية الرقمية للمتحكم STM32G0 با ستقبال القيم الت شابهية القادمة من الحـ سا سات، حيث تقوم بتحويلها إلى القيم الرقيمة المقابلة لها، فللمبدل ما يقارب الـ 19 - قناة تحويل أي 19 دخل ت شابهي للمتحكم بإمكانه استقبال القيم التشابهية من خلالها

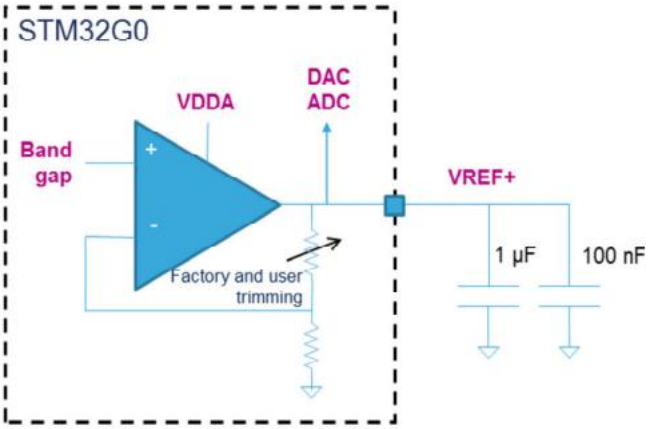
يتم حساب جهد الدخل التشابهي الناتج عن عملية التحويل من خلال العلاقة التالي:

$$Vin = ADC_{Res} * (reference\ voltage/4096)$$

حيث :

$$reference\ voltage = (V_{ref+}) - (V_{ref-})$$

يوجد داخل متحكمات STM32 مولد جهد مرجعي مدمج بداخلها يقوم بتوليد جهد مرجعي ثابت ومستقر حتى عند تغذيته من بطارية و يمكن استخدامه مع المبدل التشابهي الرقمي ADC والمبدل الرقمي التشابهي DAC ، و يعطي في خرجه قيمتين إما 2.5V أو 2.048V ، كما يمكنه أن يغذي أحمال خارجية باستجرار تيار لا يتجاوز الـ 4mA .



عند استخدام مولد الجهد المرجعي الداخلي يجب وصل مكثفات على القطب +Vref وفي هذه الحالة لن تحتاج لوصل دارة خارجية لتوليد الجهد المرجعي، كما يدعم مولد الجهد المرجعي الداخلي كما ذكرنا قيمتي جهد هي:

- 2.5V: وهذا يتطلب أن يكون الجهد على الطرف VDDA>= 2.4
- 2.48V: وهذا يتطلب أن يكون الجهد على الطرف VDDA>= 2.8

2- طرق قراءة المبدل التشابهي الرقمي ADC conversion modes:

يوجد ثلاث طرق رئيسية لقراءة الـ ADC هي:

1. **Polling method:** تعتبر الطريقة الأسهل في كتابة الكود لقراءة القيمة القادمة من إحدى القنوات التشابهية، ولكنها ليست الأكثر فعالية ، حيث علينا أن نبدأ بعملية التحويل وتتوقف الـ CPU عن تنفيذ الكود وتنتظر لحين الانتهاء من عملية التحويل حينها يمكن للـ CPU استكمال تنفيذ الكود الرئيسي.
2. **The interrupt Method:** تعتبر هذه الطريقة طريقة فعالة لا ستخدام المبدل التشابهي الرقمي ، حيث نقوم بقدح المبدل فقط و يمكن للـ CPU أن تستكمل تنفيذ الكود والمهام المطلوبة منها لحين انتهاء عملية التحويل عندها سيقوم المبدل بطلب مقاطعة وستتوجه الـ CPU لبرنامج خدمة المقاطعة وتحتفظ بناتج عملية التحويل ليتم معالجته.
3. **DMA Method**

3- التطبيق العملي الأول: سنقوم في هذا التطبيق بالتحكم بشدة إضاءة ليد موصول على أحد أقطاب الـ PWM (القطب PA0) من خلال مقاومة ضوئية موصولة على أحد أقطاب الدخل التشابهي

سنقوم بتنفيذ المشروع وفقاً للتسلسل التالي:

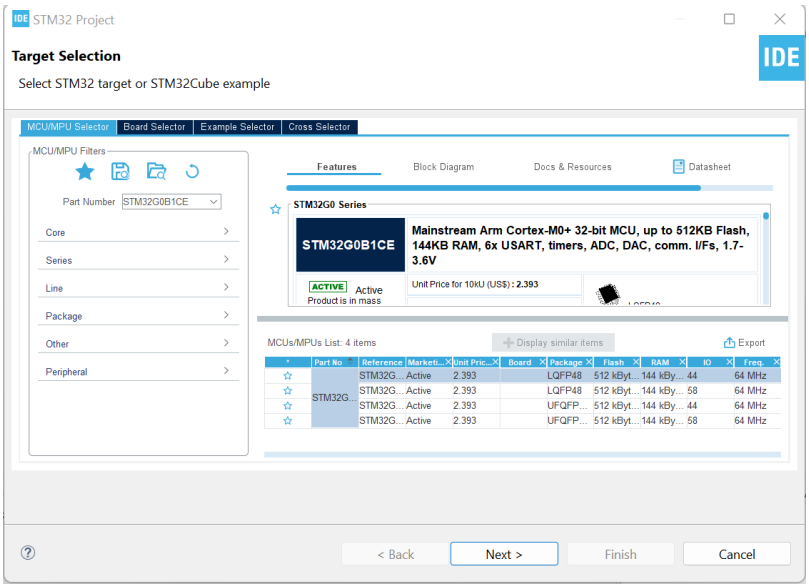
- ضبط تردد ساعة المتحكم
- ضبط قطب الدخل التشابهي (PA7) في نمط التحويل لمرة واحدة Single conversion mode حيث سنربط مقاومة ضوئية معه.
- ضبط الـ timer2 في نمط PWM على القناة CH1 (وسنربط معه ليد).

سنقوم بتنفيذ المـ شروع بطريقتي Polling ,interrupt، حيث سنقوم في البداية بقراءة القيمة القادمة من المقاومة المتغيرة الموجودة على القناة CH7 للمبدل التشابهي ومن ثم سنقوم بإسناد هذه القيمة لمسجل المؤقت CCR والذي من خلاله يتم تحديد دورة التشغيل dutycycle والتي تحدد شدة إضاءة الليد.

الطريقة الأولى: استخدام نمط الـ Polling:

سنقوم بضبط الإعدادات من خلال أداة CubeMx المدمجة داخل بيئة STM32CubeIDE وفقاً للخطوات التالية:

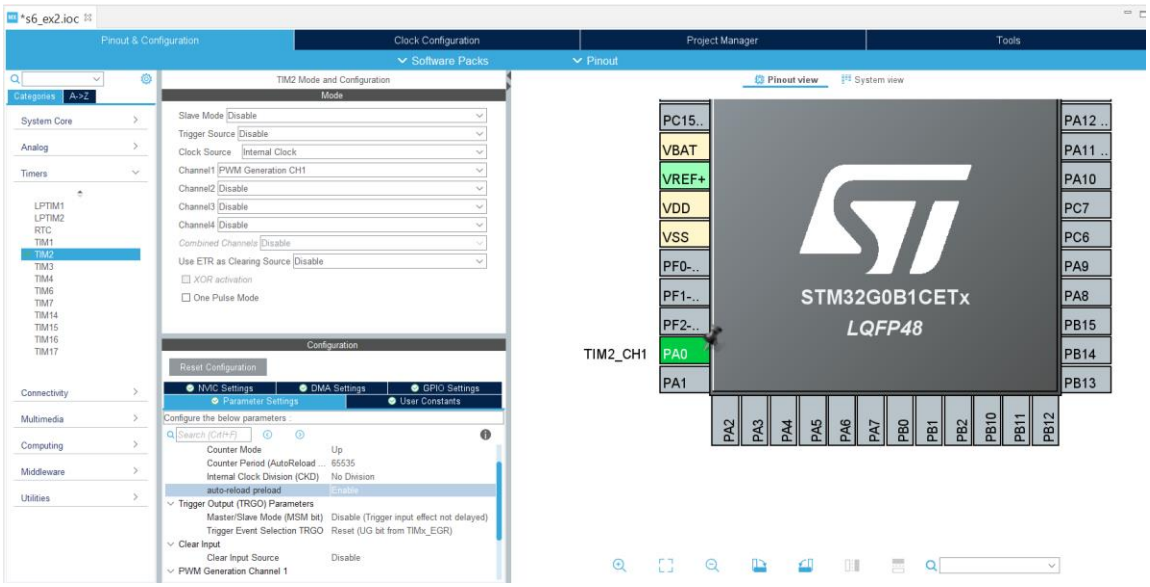
الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنـ شاء مـ شروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المـ صغر أو من خلال اختيار ا سم اللوحة المـ ستخدمة وهي في حالتنا STM32G0B1CE كما في الشكل التالي:



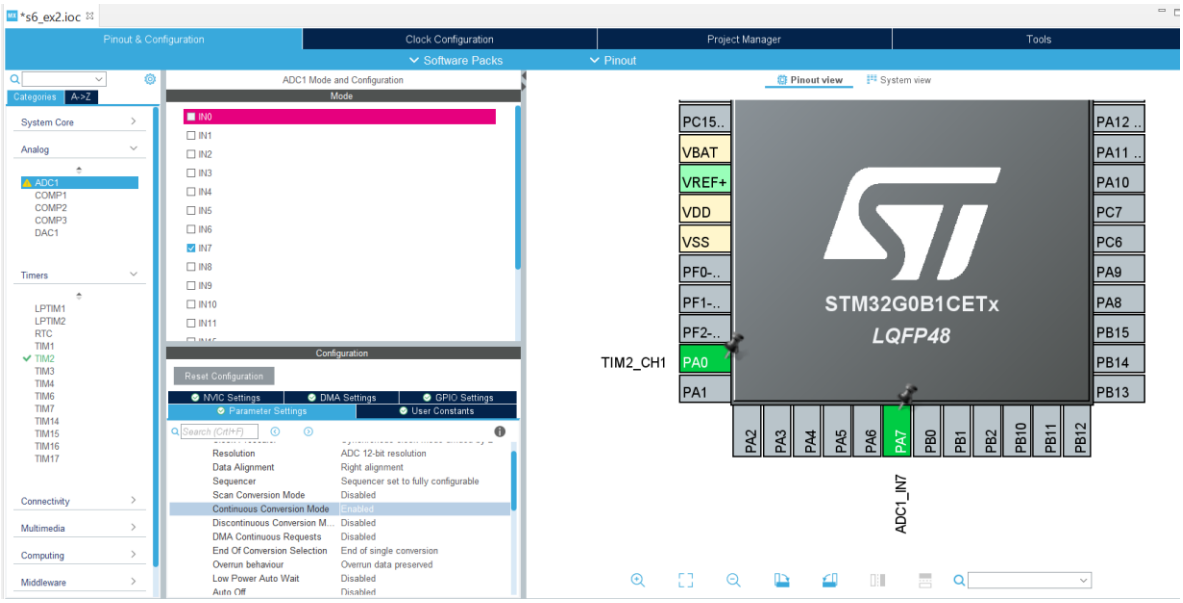
الخطوة الثانية: ضبط إعدادات المؤقت ليعمل في نمط PWM

نقوم بضبط مصدر الساعة للمؤقت على الساعة الداخلية للنظام internal clock ، نقوم بتفعيل القناة CH1 لتكون القناة التي سيتم إخراج إشارة الـ PWM عليها، نضبط القيمة العظمى للمسجل ARR على القيمة 65535 ليـ صبح تردد إ شارة PWM هوZ488.25H ، نفعل خا صية Auto Reload preload ونختار نمط إشارة الـ PWM

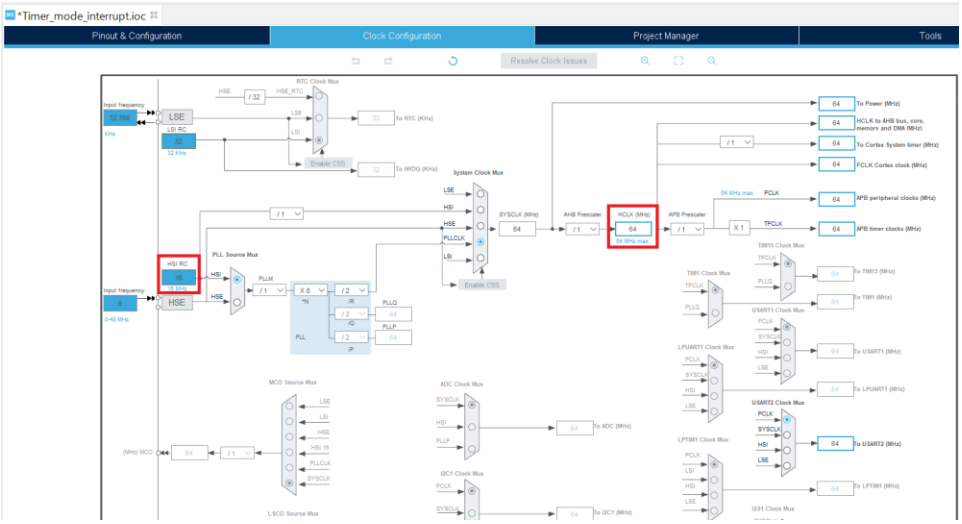
مدرس المقرر: =-----



الخطوة الثالثة: قم بضبط إعدادات المبدل التشابهي الرقمي ADC على القناة التشابهية السابعة CH7 بالشكل التالي:



الخطوة الرابعة: ضبط تردد ساعة المتحكم



الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s

يصبح الكود النهائي بالشكل التالي:

تعريف المبدل التشابهي ADC1 تعريف المؤقت TIM2	`#include "main.h"` `ADC_HandleTypeDef hadc1;` `TIM_HandleTypeDef htim2;`
التصريح عن الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2	`void SystemClock_Config(void);` `static void MX_GPIO_Init(void);` `static void MX_ADC1_Init(void);` `static void MX_TIM2_Init(void);`
التصريح عن متحول AD_RES لنخزن فيه لاحقاً القيمة الناتجة عن المحول التشابهي الرقمي	`int main(void)` `{` `  uint16_t AD_RES = 0;`
استدعاء الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2	`  HAL_Init();` `  SystemClock_Config();` `  MX_GPIO_Init();` `  MX_ADC1_Init();` `  MX_TIM2_Init();`
بدء عمل المؤقت في نمط الـ PWM وعلى القناة الأولى	`  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);`
بدء المعايرة الذاتية للمبدل التشابهي الرقمي adc1	`  // Calibrate The ADC On Power-Up For Better Accuracy` `  HAL_ADCEx_Calibration_Start(&hadc1);`

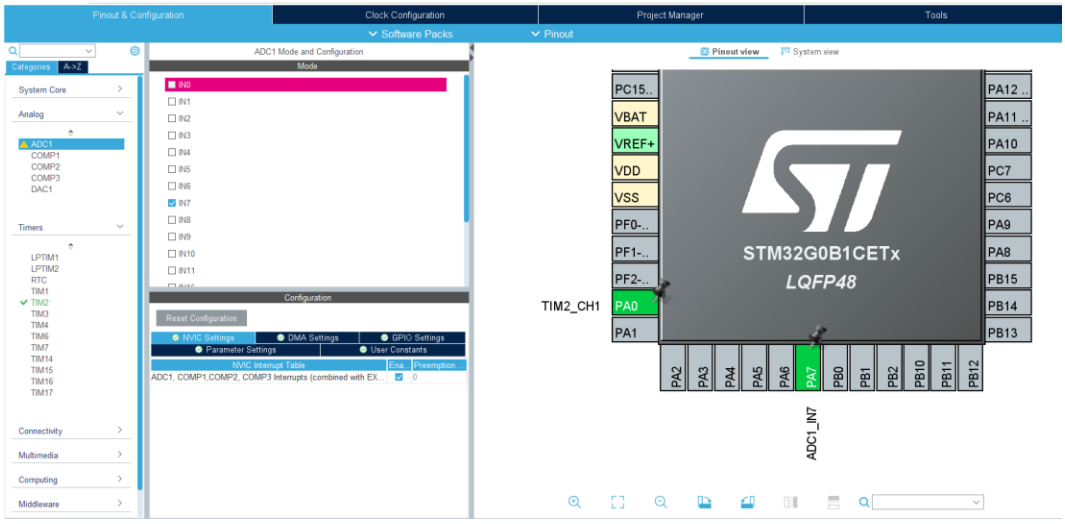
إعطاء إشارة البدء للمبدل التشابهي الرقمي ليكون جاهز فيما بعد لإجراء عمليات التحويل طلب عملية تحويل من المبدل التشابهي الرقمي إسناد ناتج عملية التحويل إلى المتغير AD_RES إسناد قيمة المتغير AD_RES التي تعبر عن قيمة المقاومة الضوئية ، إلى المسجل CCR1 من المؤقت TIM2 ، المسؤول عن دورة التشغيل dutycycle إجراء تأخير زمني بمقدار 1sec	``` while (1) { // Start ADC Conversion HAL_ADC_Start(&hadc1); // Poll ADC1 Perihperal & TimeOut = 1mSec HAL_ADC_PollForConversion(&hadc1, 1); // Read The ADC Conversion Result & Map It To PWM DutyCycle AD_RES = HAL_ADC_GetValue(&hadc1); TIM2->CCR1 = (AD_RES<<4); HAL_Delay(1); } } ```

التطبيق العملي الثاني: إعادة التطبيق الأول ولكن باستخدام نمط الـ Interrupt:

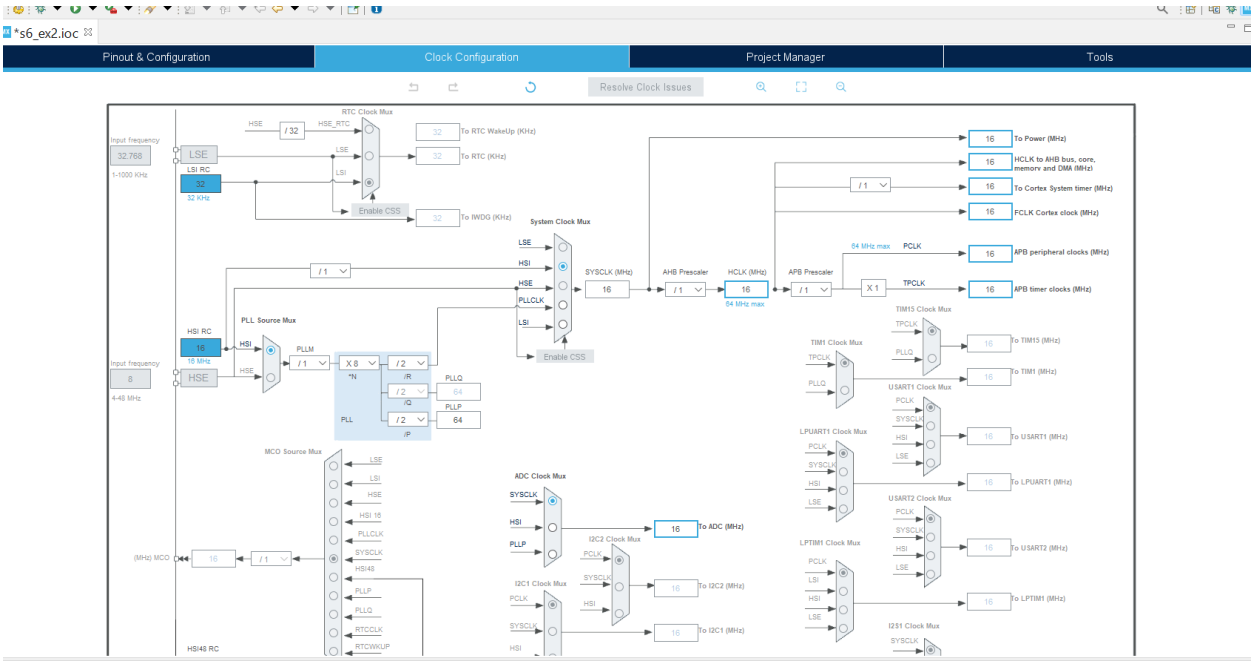
سنقوم بضبط الإعدادات من خلال أداة CubeMx المدمجة داخل بيئة STM32CubeIDE وفقاً للخطوات التالية:

الخطوة الأولى والثانية هي نفس الخطوات الموجودة في الطريقة الأولى

الخطوة الثالثة: ضبط إعدادات المبدل التشابهي بنفس الإعدادات السابقة ، فقط سنفعل المقاطعة الخاصة بالمبدل



الخطوة الرابعة: ضبط تردد ساعة المتحكم



الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s

يصبح الكود النهائي بالشكل التالي:

#include "main.h"	
uint16_t AD_RES = 0;	التصريح عن متحول عام (لأننا سنستخدمه في الدالة الرئيسية وفي برنامج خدمة المقاطعة) AD_RES لنخزن فيه لاحقاً القيمة الناتجة عن المحول التشابهي الرقمي
ADC_HandleTypeDef hadc1; TIM_HandleTypeDef htim2;	تعريف المبدل التشابهي ADC1 تعريف المؤقت TIM2
void SystemClock_Config(void); **static void MX_GPIO_Init(void);** **static void MX_ADC1_Init(void);** **static void MX_TIM2_Init(void);**	التصريح عن الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج ، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2
int main(void) {	
HAL_Init(); SystemClock_Config(); MX_GPIO_Init(); MX_ADC1_Init();	تهيئة مكتبة HAL استدعاء الدوال المسؤولة عن تهيئة ساعة المتحكم ، أقطاب الدخل/ الخرج

MX_TIM2_Init();	، تهيئة المبدل التشابهي الرقمي ADC1 ، تهيئة المؤقت TIM2
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1); // Calibrate The ADC On Power-Up For Better Accuracy HAL_ADCEx_Calibration_Start(&hadc1);	بدء عمل المؤقت في نمط الـ PWM وعلى القناة الأولى بدء المعايرة الذاتية للمبدل التشابهي الرقمي adc1
while (1) { // Start ADC Conversion HAL_ADC_Start_IT(&hadc1); // Update The PWM Duty Cycle With Latest ADC Conversion Result TIM2->CCR1 = (AD_RES<<4); HAL_Delay(1); } }	إعطاء إشارة البدء للمبدل التشابهي الرقمي ليكون جاهز فيما بعد لإجراء عمليات التحويل إسناد قيمة المتغير AD_RES التي تعبر عن قيمة المقاومة الضوئية ، إلى المسجل CCR1 من المؤقت TIM2 ، المسؤول عن دورة التشغيل dutycycle إجراء تأخير زمني بمقدار 1sec
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) { // Read & Update The ADC Result AD_RES = HAL_ADC_GetValue(&hadc1); }	برنامج خدمة المقاطعة والذي يتم استدعاؤه عند إتمام عملية التحويل التشابهي الرقمي إسناد ناتج عملية التحويل إلى المتغير AD_RES (نحصل على ناتج التحويل التشابهي الرقمي من برنامج خدمة المقاطعة)

4- التطبيق العملي الثالث: سنقوم في هذا التطبيق بمراقبة درجة حرارة الغرفة با ستخدام حـ ساس حرارة LM35 موصول على أحد أقطاب الدخل التشابهي ADC وعرض درجة الحرارة على شاشة lcd.

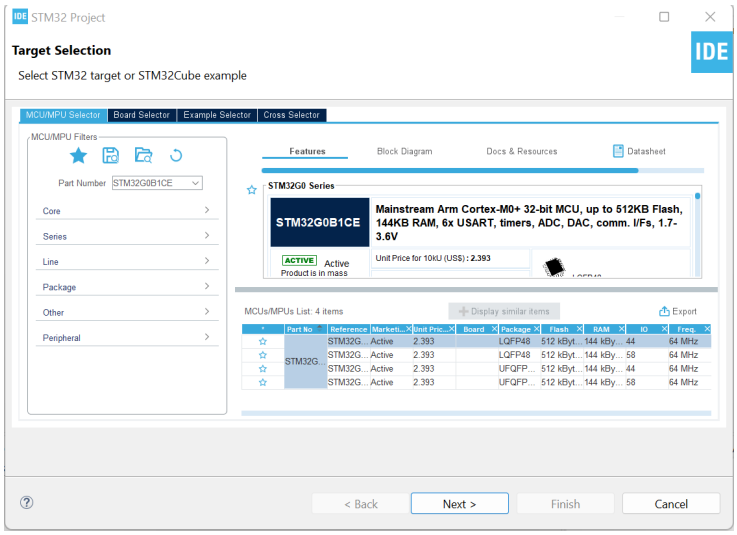
سنقوم بتنفيذ المشروع وفقاً للتسلسل التالي:

- ضبط تردد ساعة المتحكم
- ضبط قطب الدخل التشابهي (PA7) في نمط المقاطعة حيث سنصل معه حساس الحرارة lm35
- إضافة مكتبة الـ lcd إلى الكود

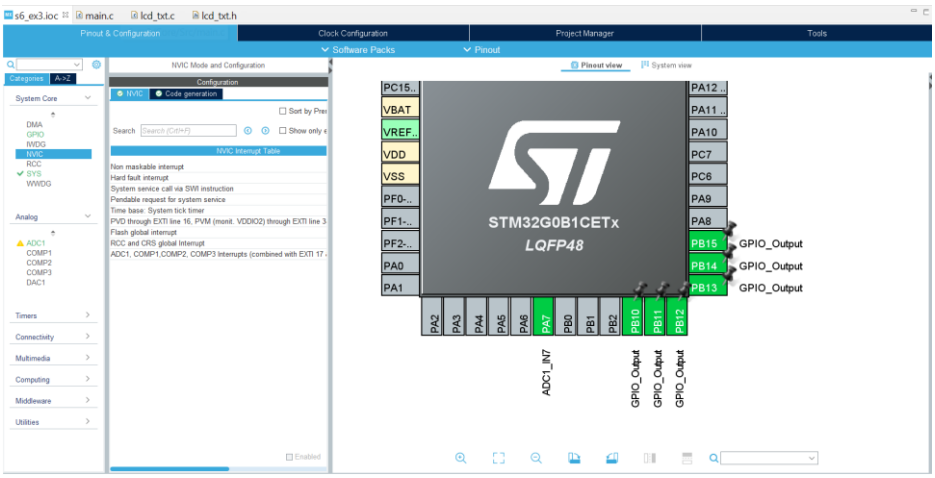
ضبط إعدادات المشروع:

الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنـ شاء مـ شروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المـ صغر أو من خلال اختيار ا سم اللوحة المـ ستخدمة وهي في حالتنا STM32G0B1CE كما في الشكل التالي:

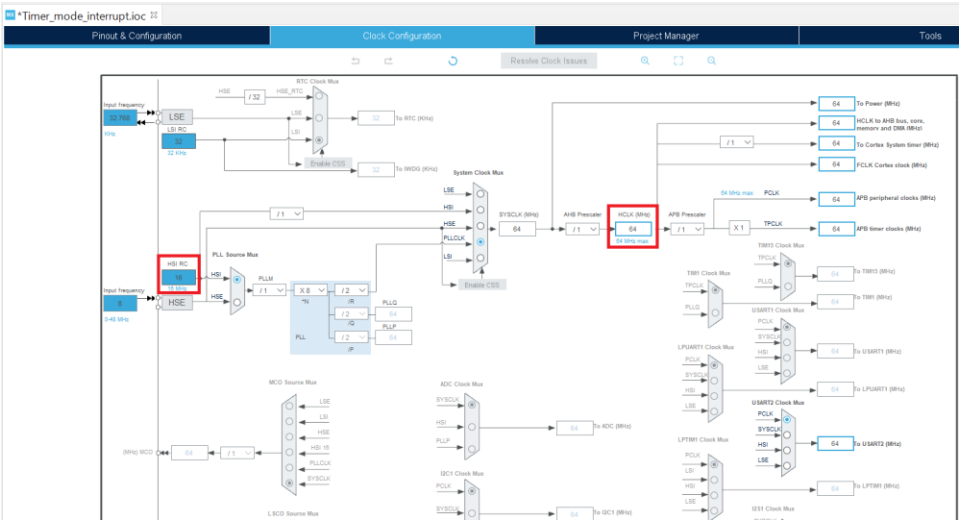
-



الخطوة الثالثة: قم بضبط إعدادات المبدل التشابهي الرقمي ADC على القناة التشابهية السابعة CH7 ، اختر الأقطاب PB10:PB15 لضبطها كأقطاب خرج ليتم وصل شاشة الـ lcd معها، بالشكل التالي:



الخطوة الرابعة: ضبط تردد ساعة المتحكم



الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s، ثم إضافة مكتبة الـ lcd والتعليمات الخاصة ببدء المبدل التشابهي ومعايرته و أخذ القراءات من الحساس وعرضها على شاشة الـ lcd

```
#include "main.h"
#include "lcd_txt.h"
ADC_HandleTypeDef hadc1;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
int main(void)
{
    uint16_t  adc_value= 0;
float voltage=0.f;
float  temp=0.f;
unsigned char buffer ;
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_ADC1_Init();
    lcd_init();
    lcd_clear();
    // Calibrate The ADC On Power-Up For Better Accuracy
    HAL_ADCEx_Calibration_Start(&hadc1);
    while (1)
    {
       // Start ADC Conversion
        HAL_ADC_Start(&hadc1);
       // Poll ADC1 Perihperal & TimeOut = 1mSec
        HAL_ADC_PollForConversion(&hadc1, 1);
       // Read The ADC Conversion Result & Map It To PWM DutyCycle
      adc_value = HAL_ADC_GetValue(&hadc1);
      voltage= adc_value*(3.3/4096);
      temp=voltage*100;
      snprintf(buffer, 10, "%d", temp);
      lcd_puts(0,0,"temp=");
      lcd_puts(0,10,buffer);
      HAL_Delay(1);
    }
}
```