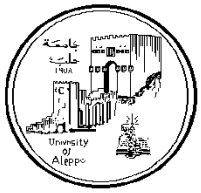
جامعة حلب
كلية الهندسة الكهربائية والالكترونية
قسم هندسة التحكم والأتمتة
مخبر التحكم



مقرر المتحكمات المصغرة

الجلسة السابعة

السنة الرابعة ميكاترونيك

2023/2022

Universal Asynchronous Serial Communications

USART/UART

محتويات الجلسة:

1- مقدمة
2- مفهوم UART و USART
3- أنماط العمل المختلفة للـ UART في متحكمات STM32
4- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ Polling
5- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt
6- التطبيق العملي الأول: مراقبة عمل أي كود (Debugging) من خلال نافذة الاتصال التسلسلي UART وباستخدام نمط الـ Polling
7- التطبيق العملي الثاني: إعادة التطبيق السابق باستخدام نمط الـ interrupt

الأدوات اللازمة للجلسة:

- البورد التطويري من شركة Hexabitz
- كبل Type-A to Mini-B
- ليدات
- كبل FTDI للاتصال التسلسلي بالبورد

1- مقدمة :

في هذه الأيام يوجد العديد من بروتوكولات الاتصال التسلسلية Serial communication Protocols ، حيث يركز أغلبها على سرعة نقل البيانات مثل USB 3.0 /USB2.0 وغيرها...، إحدى طرق الاتصال التي تم إنشاؤها قديماً ومازالت تستخدم حتى الان للربط بين المتحكمات المصغرة هي Universal Synchronous/Asynchronous Receiver/Transmitter Interface (USART).

Synchronous: هو الإرسال والاستقبال المتزامن المبني على وجود clock بين المرسل والمستقبل .

Asynchronous: لا يعتمد على clock وإنما يتم الاكتفاء بإرسال البيانات على خط الإرسال ويتم استقبالها على خط الاستقبال.

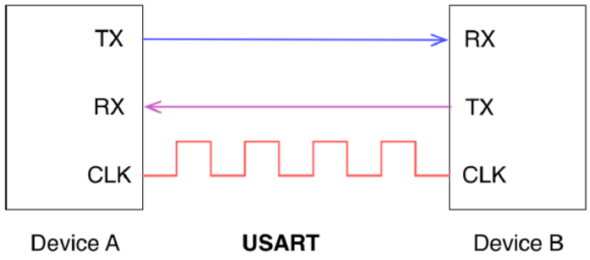
كل متحكم STM32 يحتوي على الأقل على وحدة طرفية UART واحدة، وأغلب متحكمات STM32 توفر على الأقل اثنتين من UART/USART ، وأخرى توفر لحد 8 وحدات طرفية UART/USART، حيث توفر لوحتنا التطويرية التي تحتوي على متحكم stm32G0B1CE على 6 وحدات طرفية للـ UART/USART.

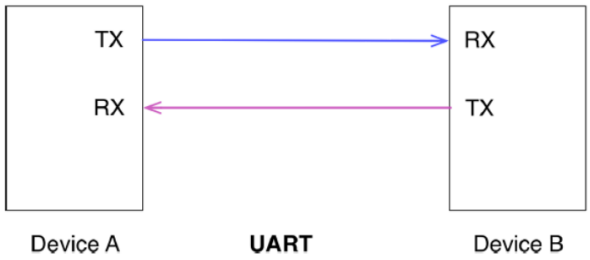
2- مفهوم UART و USART:

عندما تريد نقل البيانات ما بين جهازين أو أكثر، يوجد طريقتين لإرسال البيانات الأولى وهي:

- نقل البيانات على التوازي Parallel : في هذه الطريقة توجد مجموعة من خطوط البيانات على حسب طول البيانات التي سيتم إرسالها (مثلاً 8 خطوط بيانات في حال إرسال بيانات بطول 8بت).
- نقل البيانات على التسلسل Serial : وفي هذه الطريقة يتم إرسال البيانات على التوالي بت تلو الآخر باستخدام خط بيانات واحد حيث يكون أحد الجهازين المتصلين مرسل والآخر مستقبل.

في وضع Synchronous يتم مشاركة clock ما بين المرسل والمستقبل والتي يتم إنشاؤها دائماً باستخدام الجهاز الذي يدير الاتصال، أما في حالة الإرسال الغير متزامن (Asynchronous) يتم الاستغناء عن الـ clock - حيث يتم استخدام بت عند بداية الإرسال Start Bit و بت عن انتهاء الإرسال Stop Bit.





Pin number								Pin name (function upon reset)	Pin type	I/O structure	Note	Alternate functions	Additional functions
WLCSP25	UFQFPN28 - GP	UFQFPN28 - N	LQFP32 / UFQFPN32 - GP	LQFP32 / UFQFPN32 - N	LQFP48 / UFQFPN48	UFBGA64	LQFP64						
D4	7	7	8	8	12	H3	18	PA1	I/O	FT_a	-	SPI1_SCK/I2S1_CK, USART2_RTS_DE_CK, TIM2_CH2, USART4_RX, TIM15_CH1N, I2C1_SMBA, EVENTOUT	COMP1_INP, ADC_IN1
E4	8	8	9	9	13	G3	19	PA2	I/O	FT_a	-	SPI1_MOSI/I2S1_SD, USART2_TX, TIM2_CH3, UCPD1_FRSTX, TIM15_CH1, LPUART1_TX, COMP2_OUT	COMP2_INM, ADC_IN2, WKUP4,LSCO
C3	9	9	10	10	14	F3	20	PA3	I/O	FT_a	-	SPI2_MISO, USART2_RX, TIM2_CH4, UCPD2_FRSTX, TIM15_CH2, LPUART1_RX, EVENTOUT	COMP2_INP, ADC_IN3
-	-	-	-	-	15	H4	21	PA4	I/O	TT_a	-	SPI1_NSS/I2S1_WS, SPI2_MOSI, TIM14_CH1, LPTIM2_OUT, UCPD2_FRSTX, EVENTOUT	ADC_IN4, DAC_OUT1, RTC_OUT2

3- أنماط العمل المختلفة للـ UART في متحكمات STM32:

توجد 3 أنماط عمل مختلفة للـ UART وهي:

1. **نمط Polling Mode:**

يـ سمى أيـ ضاً Blocking Mode ، في هذا النمط يتم تفحص عملية إر سال وا ستقبال البيانات بـ شكل مـ ستمر ، حيث ينتظر المعالج لحين انتهاء عملية الإر سال مما يؤدي إلى تأخير معالجة باقي التعليمات وتنفيذ المهام ، وهو نمط العمل الأبسط من ناحية الكود ومن ناحية الـ Hardware ويستخدم عندما تكون كمية البيانات المتبادلة ليست كبيرة نسبياً ولا تمثل أهمية عالية من ناحية المعالجة.

2. **نمط المقاطعة Interrupt Mode:**

ويـ سمى أيـ ضاً non-Blocking Mode ، في هذا النمط لا يتم الانتظار وتفقد البيانات من حين لآخر للتأكد من عملية الارسال والاستقبال، حيث عند الانتهاء من إرسال البيانات يتم تفعيل مقاطعة تفيد بانتهاء عملية الإر سال ، وهذا النمط من العمل أف ضل من ناحية المعالجة ملائم عندما يكون معدل نقل البيانات صغير نسبياً (أقل من 38400Bps).

3. **نمط DMA:**

وهو النمط الأف ضل من ناحية إنتاجية نقل البيانات ومن ناحية سرعة نقل البيانات وعندما نريد تحرير المتحكم من الحمل الإ ضافي الذي ينتج عن من إحـ ضار البيانات من RAM ومعالجتها، فالـ DMA - يقوم بالوصول إلى الذاكرة RAM بدون احتياج أي جهد من المعالج لعمل ذلك، وبدون نمط الـ DMA - لا يمكن التعامل مع السرعات العالية في الـ UART.

سنشرح بالتفصيل نمطي الـ polling والـ Interrupt من خلال التطبيق العملي.

4- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ Polling :

- **دوال مكتبة HAL المستخدمة للتعامل مع منفذ الاتصال التسلسلي في وضع الـPolling :** سنستخدم دالتين رئيسيتين للتعامل مع المنفذ التسلسلي إحداهما للإرسال والأخرى للاستقبال:

1. **دالة الإرسال:**

```
HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t
Timeout);
```

حيث:

huart: وهو مؤشر يشير إلى Struct_UART_HandleTypeDef أي المنفذ التسلسلي المستخدم للاتـ صال مثلاً قد يكون &huart1 أو &huart2 أو &huart3 أو &huart4 باعتبار لدينا أربع منافذ تسلسلية في لوحة Nucleo.

pData: وهو مؤ شر أيـ ضاً يـ شير إلى البيانات التي سيتم إر سالها عبر UART وكما نرى نوعه uint8_t أي يقبل إر سال بيانات من نوع Unsigned int وبطول 8bit، مثال: قد تكون pData مصفوفة وليكن اسمها Data وتكون معرفة بالشكل التالي:

Uint8_t Data[]= {0,1,2,3,4,5,6,7,8,9};

حيث:

Size: وهو متغير يعبر عن حجم البيانات التي سيتم إرسالها أي pData وهي في المثال السابق 10.

Timeout: أقـ صى زمن يتم انتظاره بالميللي ثانية حتى يتم اكتمال عملية الإر سال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الإرسال سيتم قطع عملية الإرسال وتقوم الدالة بإرجاع HAL_TIMOUT ماعدا

ذ لك يـ تم ار جاع HAL_OK، و يـ مـ كن ا ستـ خدام هذه ا لدا لة مـ كان Timeout و هي HAL_MAX_DELAY ووظيفتها انتظار أقصى زمن ممكن لعملية الإرسال.

مثال: إذا أردنا إرسال المصفوفة التالية عبر المنفذ التسلسلي الأولUART1:

```
/* USER CODE BEGIN 0 */
uint8_t data[]={0,1,2,3,4,5,6,7,8,9};
/* USER CODE END 0 */
```

نستخدم الدالة التالية:

```
/* USER CODE BEGIN 3 */
 /* Infinite loop */
 while (1)
 {
   HAL_UART_Transmit(&huart1,data,10,1000);
 }
 /* USER CODE END 3 */
```

2. دالة الاستقبال:

إذا أردنا استقبال بيانات على UART باستخدام وضع Polling ومكتبات HAL نقوم باستدعاء الدالة التالية:

```
HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);
```

حيث:

Size: وهو متغير يعبر عن حجم البيانات التي سيتم استقبالها أي pData وهي في المثال التالي 10.

Timeout: أقـ صـى زمن يتم انتظاره بالميللي ثانية حتى يتم اكتمال عملية الا ستقبال، فإذا تم انتهاء هذا الزمن ولم تتم عملية الا ستقبال سيتم قطع عملية الا ستقبال وتقوم الدالة بإرجاع HAL_TIMOUT ما عدا ذ لك يتم ار جاع HAL_OK، ويمكن ا ستـ خدام هذه ا لدا لة مـ كان Timeout وهي HAL_MAX_DELAY ووظيفتها انتظار أقصى زمن ممكن لعملية الاستقبال.

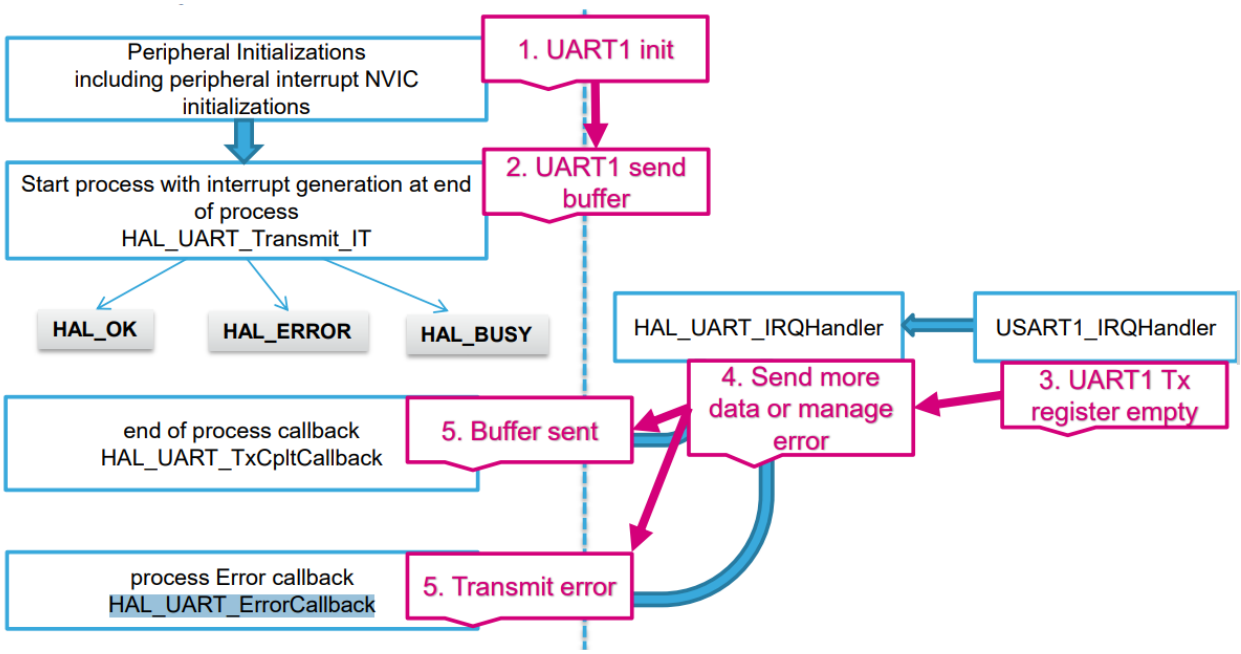
مثال: إذا أرنا استقبال مصفوفة أعداد صحيحة من المنفذ التسلسلي UART1:

```
/* USER CODE BEGIN 0 */
uint8_t data[10];
/* USER CODE END 0 */
```

```
/* USER CODE BEGIN 3 */
 /* Infinite loop */
 while (1)
 {
   HAL_UART_Receive(&huart1,data,10,1000);
 }
 /* USER CODE END 3 */
```

5- إنشاء اتصال عبر المنفذ التسلسلي باستخدام نمط الـ interrupt:

خطوات تفعيل واستخدام المقاطعة مع المنفذ التسلسلي: يتم تهيئة و استخدام المقاطعة مع المنفذ التسلسلي وفق المخطط التالي:



الخطوة الأولى:

تهيئة المنفذ التسلسلي من خلال اختيار رقم المنفذ التسلسلي و ضبط معدل نقل البيانات BaudRate وتفعيل المقاطعة على المنفذ وغيرها من الإعدادات ويتم ذلك با ستخدام أداة الـ CubeMx (التي أ صبحت مدمجة داخل بيئة STM32CubeIDE) ، وتلقائياً يتم إ ضافة سطور التهيئة عند توليد الكود ضمن ملف الـ main.c والملف stm32G0xx_hal_msp.c.

الخطوة الثانية:

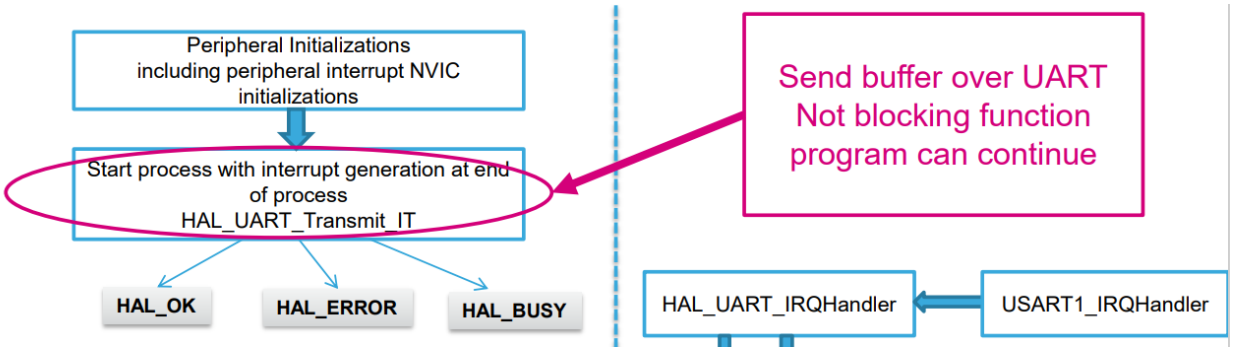
إرسال البيانات عبر المنفذ التسلسلي من خلال استخدام الدالة التالية:

```
HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
```

حيث يتم إدخال بارامترات هذه الدالة كما قمنا بالشرح سابقاً ، وكما تلاحظ فقد تم إضافة IT في اسم الدالة وأيضاً تم إزالة Timeout من بارامترات هذه الدالة مقارنةً بالدالة المستخدمة في نمط الـ Polling - ، لأنه لم يعد هناك زمن انتظار في نمط المقاطعة.

لاستقبال مجموعة من البايتات عبر المنفذ التسلسلي نستخدم الدالة التالية:

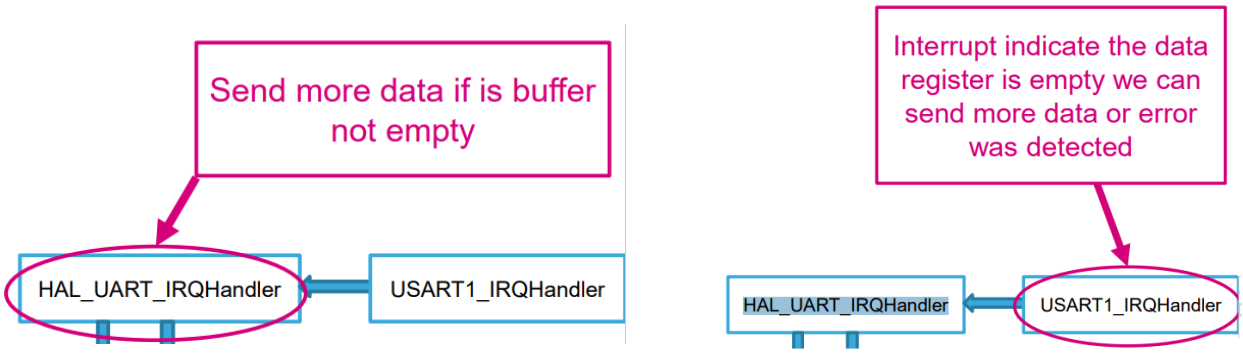
```
HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
```



تعيد دالة الإرسال أو الاستقبال إما HAL_OK في حال تمت عملية الإرسال/الاستقبال بنجاح، أو HAL_error في حال حدوث خطأ أثناء عملية الإرسال/ الاستقبال أو HAL_Busy .

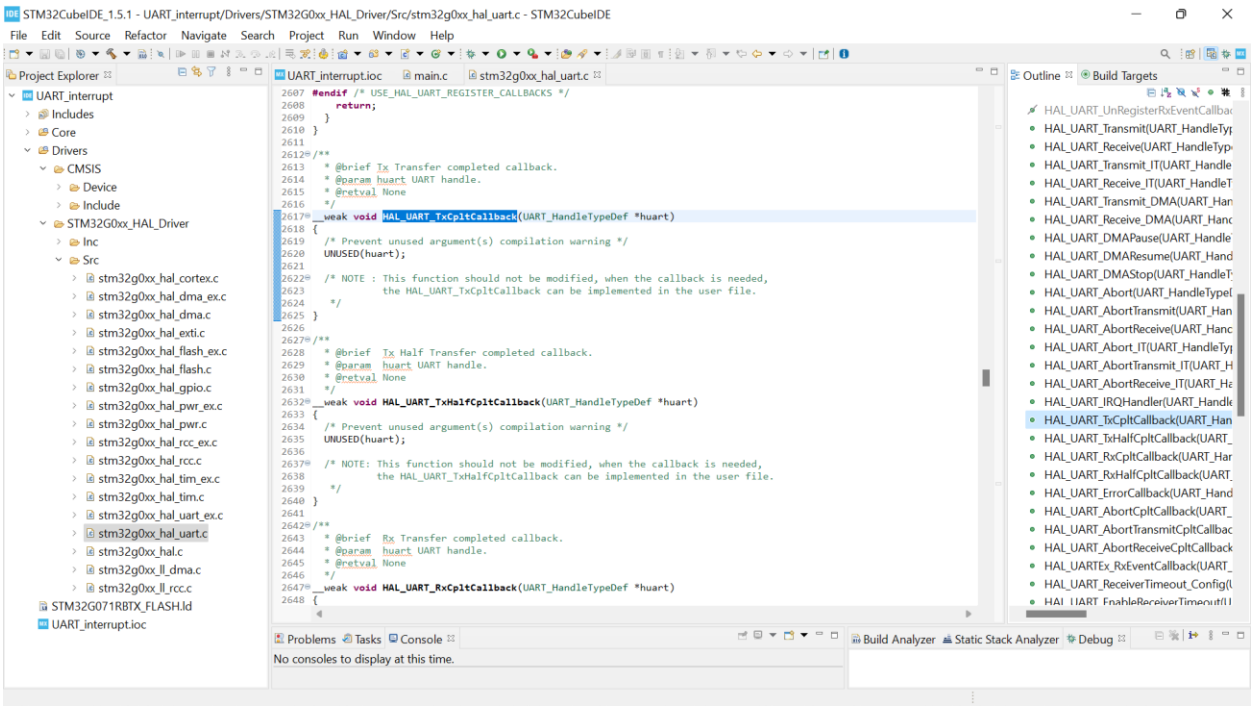
الخطوة الثالثة: تفحص فيما إذا أصبح مسجل البيانات UART1_TX فارغاً

الخطوة الرابعة: في حال كان مسجل البيانات UART1_TX فارغاً (يتم اكتشافه من خلال حدث المقاطعة وبالتالي استدعاء HAL_UART_IRQHandler والمعرف ضمن ملف stm32G0xx_hal_uart.c) فلدينا خيارين إما إرسال البيانات المتبقية في حال كان الـ Buffer - غير فارغ أو الإعلان عن انتهاء عملية الإرسال ، أو الكشف عن وجود خطأ في عملية الإرسال.

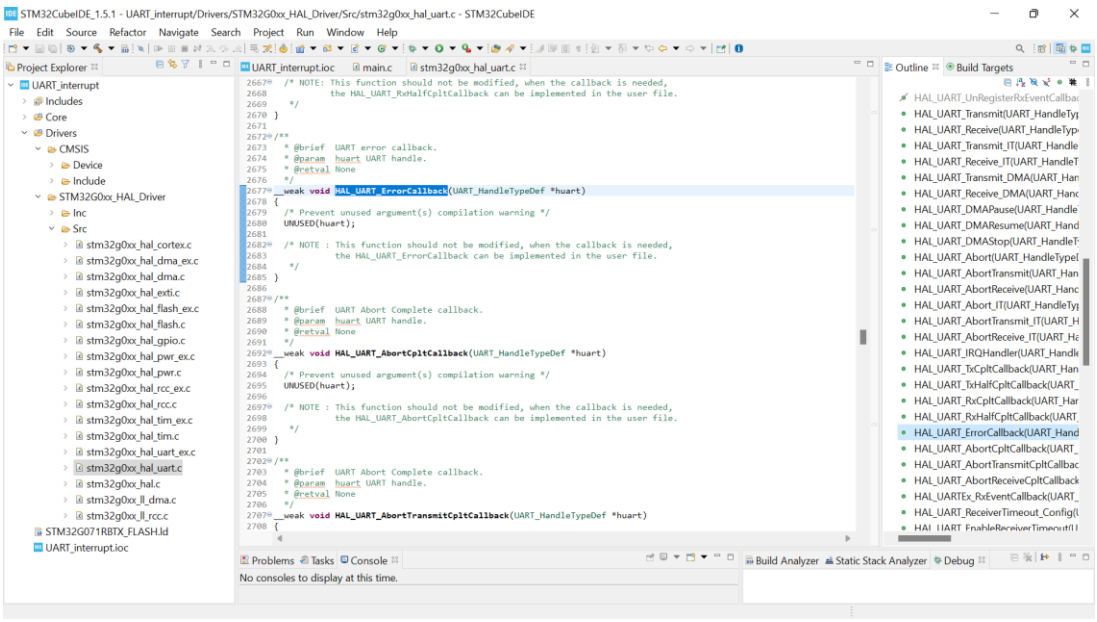


الخطوة الخامسة: لدينا خيارين:

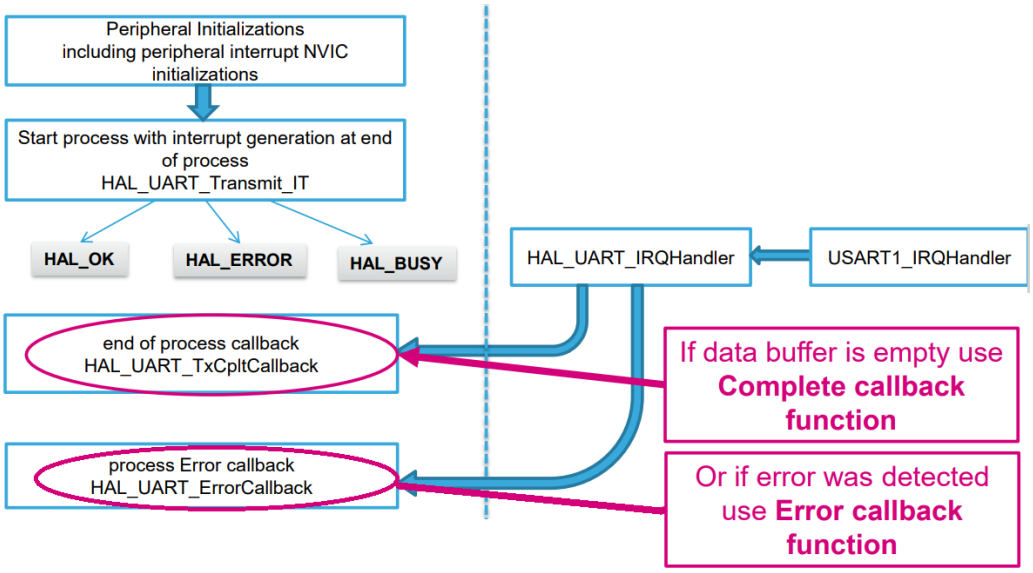
في حال تم إر سال البيانات بنجاح عندها يتم ا ستدعاء الدالة HAL_UART_TxCpltCallback والتي تكون معرفة ك ـ weak_ (أي يتم ا ستدعائها في حال لم يتم إعادة تعريفها ضمن البرنامج الرئي سي) ضمن ملف stm32G0xx_hal_uart.c ضمن مجلد stm32G0xx_HAL_Driver ثم المجلد Src ، نقوم بن سخ الدالة HAL_UART_TxCpltCallback ووضعها ضمن ملف البرنامج الرئيسي main.c لاستخدامها لاحقاً.



في حال تم اكت شاف خطأ أثناء إر سال البيانات يتم ا ستدعاء الدالة HAL_UART_ErrorCallback والتي تكون معرفة ك ـ weak_ (أي يتم استدعائها في حال لم يتم إعادة تعريفها ضمن البرنامج الرئيسي) ضمن ملف stm32G0xx_hal_uart.c ضمن مجلد stm32G0xx_HAL_Driver ثم المجلد Src كما في الـ شكل التالي:

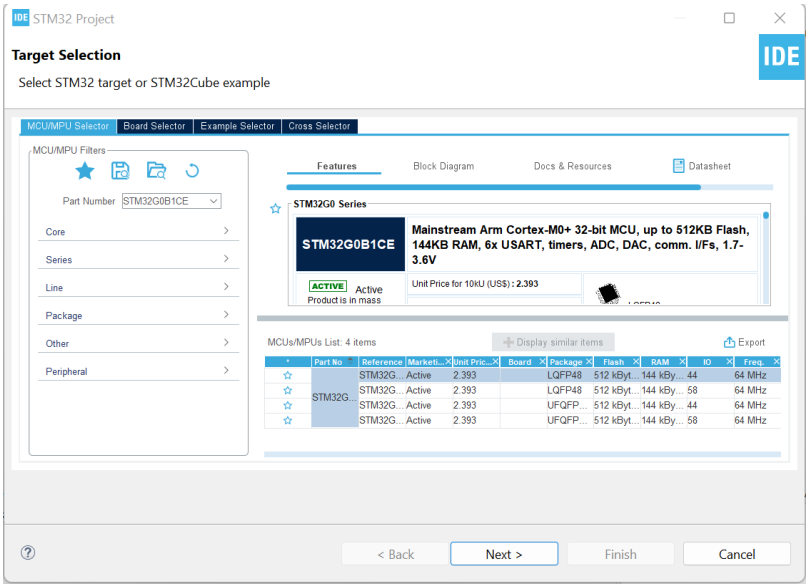


نقوم بنسخ الدالة HAL_UART_ErrorCallback ووضعها ضمن ملف البرنامج الرئيسي main.c لاستخدامها لاحقاً.

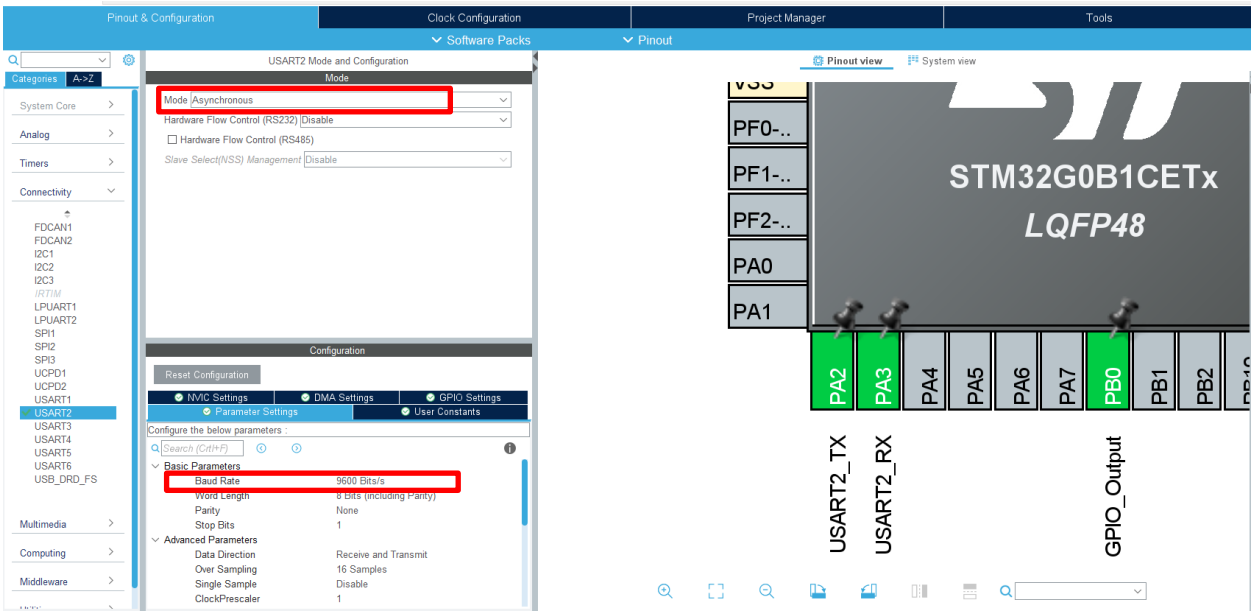


6- التطبيق العملي الأول: مراقبة عمل أي كود (Debugging) من خلال نافذة الاتصال التسلسلي UART وبا ستخدام نمط الـ Polling: للاتصال باللوحة من خلال الـ Serial - نحتاج إلى دارة FT232RL.

الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنشاء مشروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المصغر أو من خلال اختيار اسم اللوحة المستخدمة وهي في حالتنا G0B1CE كما في الشكل التالي:



الخطوة الثانية: قم بضبط إعدادات المنفذ UART2 ، كما في الشكل التالي:



من ضمن الإعدادات الموجودة Basic Rate ما يلي:

Baudrate: يمثل معدل نقل البيانات بواحدة Bit/Sec بين المرسل والمستقبل، ولها قيم قياسية يتم الاختيار منها، حيث تعتمد هذه القيم على وحدة الـ Clock Peripheral الخاصة بالـ USART - وهي عبارة عن ساعة المتحكم المصغرة على رقم ثابت، لكن ليس كل الـ BaudRates المتاحة يمكن استخدامها فقد ينتج عن المعدلات العالية أخطاء، حيث يوضح الشكل التالي الـ BaudRates القياسية والأخطاء التي قد تنجم عن كل BaudRate، حيث تختلف قيم الـ BaudRates المتاحة من متحكم لآخر بناءً على Clock Peripheral التي يوفرها للـ USART فقد يدعم متحكم BaudRates أكثر أو أقل.

Baud rate		Oversampling by 16		Oversampling by 8	
S.No	Desired (Bps)	Actual	%Error	Actual	%Error
2	2400	2400	0	2400	0
3	9600	9600	0	9600	0
4	19200	19200	0	19200	0
5	38400	38400	0	38400	0
6	57600	57620	0.03	57590	0.02
7	115200	115110	0.08	115250	0.04
8	230400	230760	0.16	230210	0.8
9	460800	461540	0.16	461540	0.16
10	921600	923070	0.16	923070	0.16
11	2000000	2000000	0	2000000	0
12	3000000	3000000	0	3000000	0
13	4000000	N.A.	N.A.	4000000	0
14	5000000	N.A.	N.A.	5052630	1.05
15	6000000	N.A.	N.A.	6000000	0

- اختر معدل نقل البيانات 9600 (عند ا ستخدام م صدر الـ ساعة الداخلية للمتحكم فلا يمكن للمتحكم نقل البيانات عبر منفذ الـ UART بسرعة عالية)

WordLength: وتعني عدد البتات التي يتم إر سالها أو ا ستقبالها في Frame (في المرة الواحدة)، وتوفر 3 قيم يمكن الاختيار بينها 7bit,8bit,9bit حيث لا يتضمن هذا الرقم البتات الخاصة بـ Start و الـ Stop وغيرها.

StopBits: يحدد عدد البتات الخاصة بالـ Stop - التي سيتم إر سالها، ويمكن الاختيار بين 1 و 2 أي بت واحد أو 2bit في نهاية الإشارة.

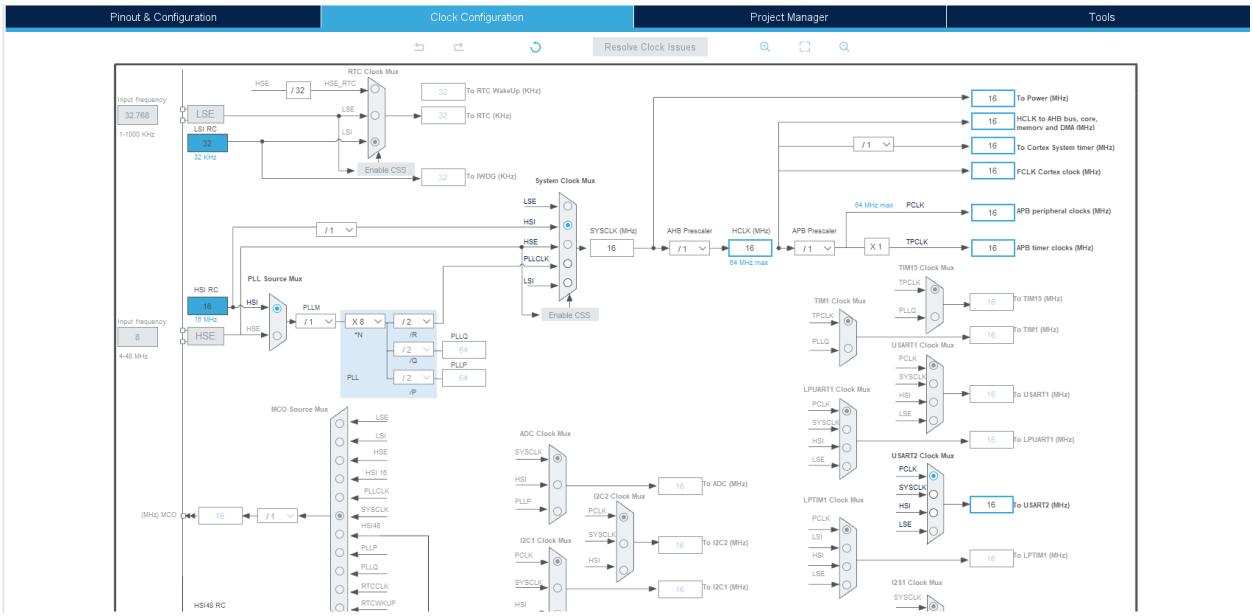
Parity: هو عبارة عن اختبار يـ ستخدم لاكت شاف الأخطاء أثناء عملية الإر سال والا ستقبال للبيانات من خلال الـ USART ، وهو عبارة عن بت يكون مكانه عند البت الأكثر أهمية MSB بحيث لو تم ا ستخدام Word Length بـ 8-bit يكون مكانه في البت الثامن، أما لو تم استخدام 9-bit يكون مكانه هو في البت التاسع، ولها نمطين:

- فردي Odd: وتكون قيمة بت الـ Parity - مـ ساوٍ للواحد المنطقي عندما يكون عدد الواحدات الموجودة في الكلمة المراد إر سالها زوجي، و صفر منطقي في حال كان عدد الواحدات الموجودة في الكلمة المراد إر سالها فردي.
- زوجي Even: وتكون قيمة بت الـ Parity - مـ ساوٍ للواحد المنطقي عندما يكون عدد الواحدات الموجودة في الكلمة المراد إر سالها فردي، و صفر منطقي في حال كان عدد الواحدات الموجودة في الكلمة المراد إر سالها زوجي.

على سبيل المثال: عندما تريد إر سال أي بيانات يتم تحويلها لـ Binary - فمثلاً إذا كنا نريد إر سال الكلمة التالية 0b01101110 فمن خلال الـ Parity - يتم حساب عدد الواحدات الموجودة ضمن هذه الكلمة المراد إرسالها وهي في هذه الحالة 5، ففي حال كنت تـ ستخدم نمط الفردي ستكون قيمة الـ Parity - صفر، أما في حال كنت تـ ستخدم نمط الزوجي ستكون قيمة الـ Parity واحد.

ويتم إرسال قيمة البت الخاص بالـ Parity من المرسل إلى المستقبل، فإن لم يحصل تطابق بين قيمته عند المرسل مع قيمته عند المستقبل فهذا يعني وجود خطأ ما في الإرسال حيث يتم طلب إعادة الإرسال.

الخطوة الثالثة: ضبط تردد الساعة للمتحكم (تذكر أن لوحة Nucleo التي نستخدمها لا تحتوي على كريستالة خارجية ولكن بإمكانك إضافة واحدة) سنختار مصدر الساعة الداخلي HSI:



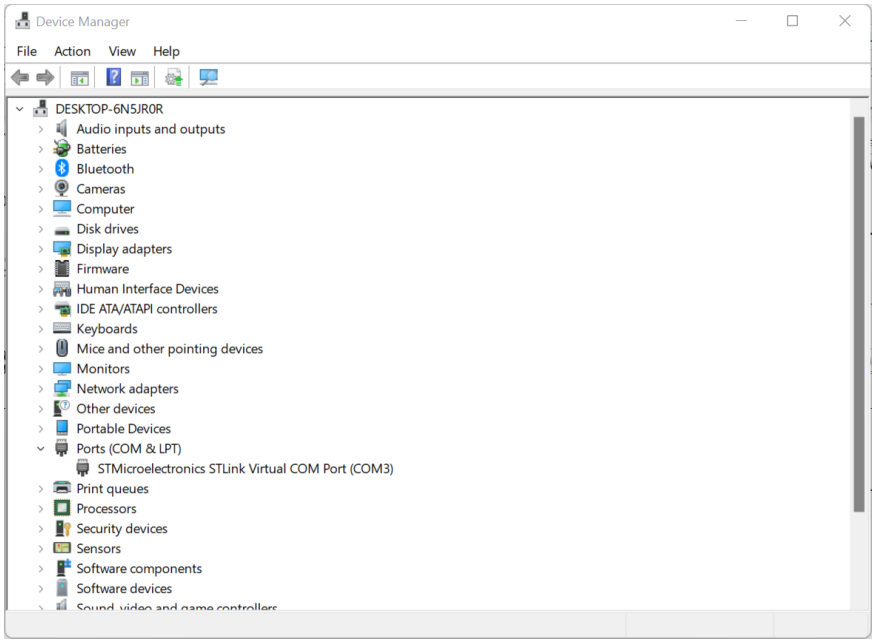
الخطوة الرابعة: توليد الكود اعتماداً على الإعدادات التي تم اختيارها من Project...Generate code

الخطوة الخامسة: كتابة الكود المناسب، حيث سنقوم بطباعة عبارة ("Hello Dudes! Tracing X =") وسنقوم بتعريف عداد X تزداد قيمته في كل مرة يعيد فيها المعالج تنفيذ الحلقة اللانهائية While(1)، ويكون الكود بالشكل التالي:

تضمين المكتبة الرئيسية main.h	#include "main.h"
تعريف منفذ الاتصال المستخدم وهو في حالتنا UART2	UART_HandleTypeDef huart2;
تعريف الدالة المستخدمة لضبط إعدادات ساعة المتحكم	void SystemClock_Config(void);
تعريف الدالة المستخدمة لضبط إعدادات أقطاب الدخل والخرج للمتحكم	static void MX_GPIO_Init(void);
تعريف الدالة المستخدمة لضبط إعدادات المنفذ التسلسلي للمتحكم	static void MX_USART2_UART_Init(void);
البرنامج الرئيسي	int main(void) {
تصريح عن مصفوفة محارف بعدد 35 محرف تصريح عن متحول من نوع uint_8t وإسناد قيمة صفرية له كقيمة ابتدائية	uint8_t MSG[35] = {'\0'}; uint8_t X = 0;

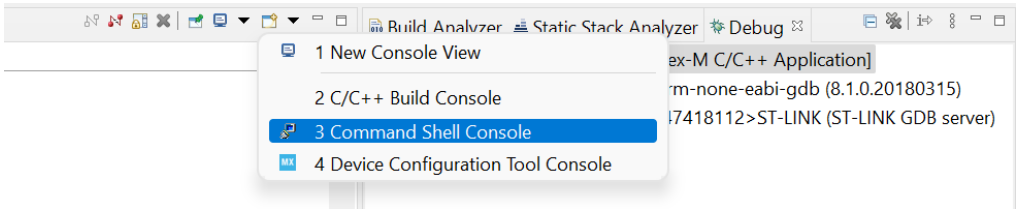
استدعاء الدالة المسؤولة عن تهيئة مكتبة HAL استدعاء الدالة المسؤولة عن ضبط إعدادات ساعة المتحكم استدعاء الدالة المسؤولة عن ضبط أقطاب الدخل والخرج للمتحكم استدعاء الدالة المستخدمة لضبط إعدادات المنفذ التسلسلي للمتحكم حلقة While(1) اللانهائية تشكيل الجملة المراد إرسالها على المنفذ التسلسلي إرسال السلسلة MSG إلى المنفذ التسلسلي UART2 إضافة تأخير زمني بين عمليات الإرسال المتكررة بمقدار 500 ميللي ثانية زيادة المتحول X في كل مرة	HAL_Init(); SystemClock_Config(); MX_GPIO_Init(); MX_USART2_UART_Init(); while (1) { sprintf(MSG, "Hello Dudes! Tracing X = %d\r\n", X); HAL_UART_Transmit(&huart2, MSG, sizeof(MSG), 100); HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0); HAL_Delay(500); X++; }}

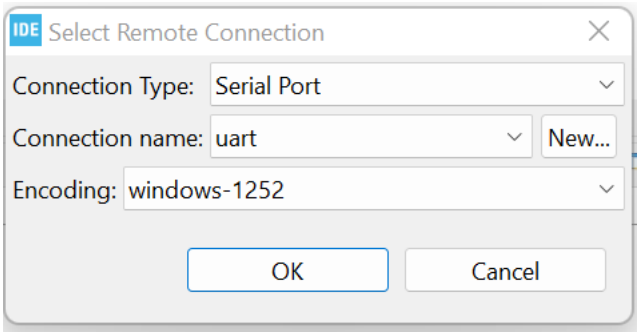
الخطوة السادسة: ترجمة الكود وإرساله إلى البورد التطويري ، ثم قم بفتح Device Manager لتعرف رقم المنفذ التسلسلي المستخدم من الحاسب للاتصال مع اللوحة:



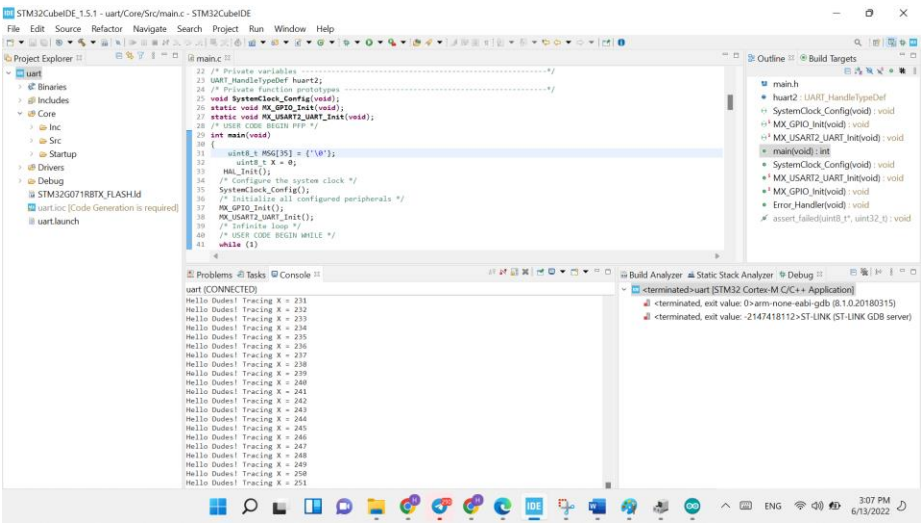
الخطوة السابعة: من القائمة Windows اختر show view ثم console فتجد أن الـ console ظهرت في أسفل الشاشة ، ثم من console نختار command shell console ثم نختر نوع منفذ الاتصال وهو Serial port

ونعطيه اسم مثلاً uart كما في الشكلين التاليين:



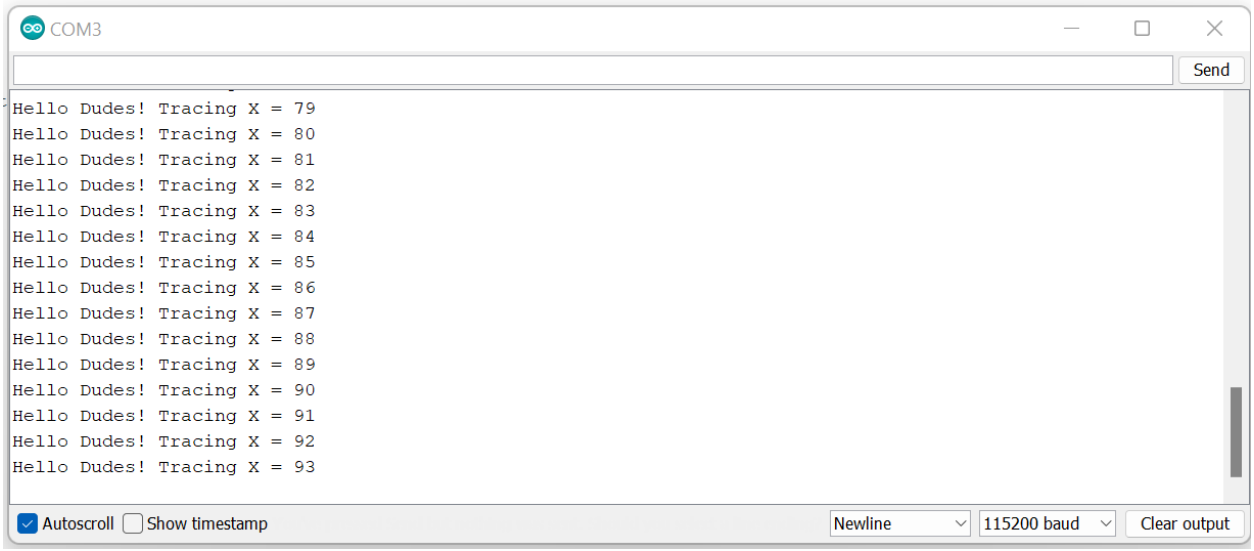


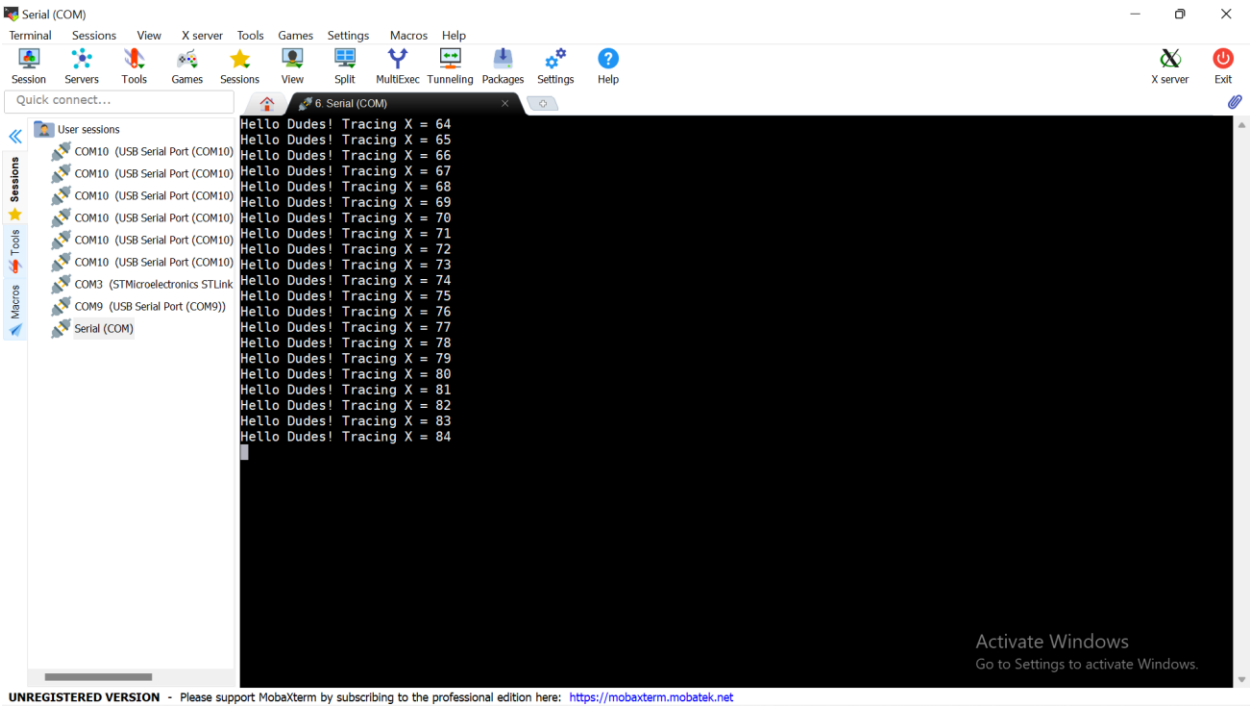
فتبدأ عملية طباعة الجملة المرادة كما في الشكل التالي:



ملاحظة:

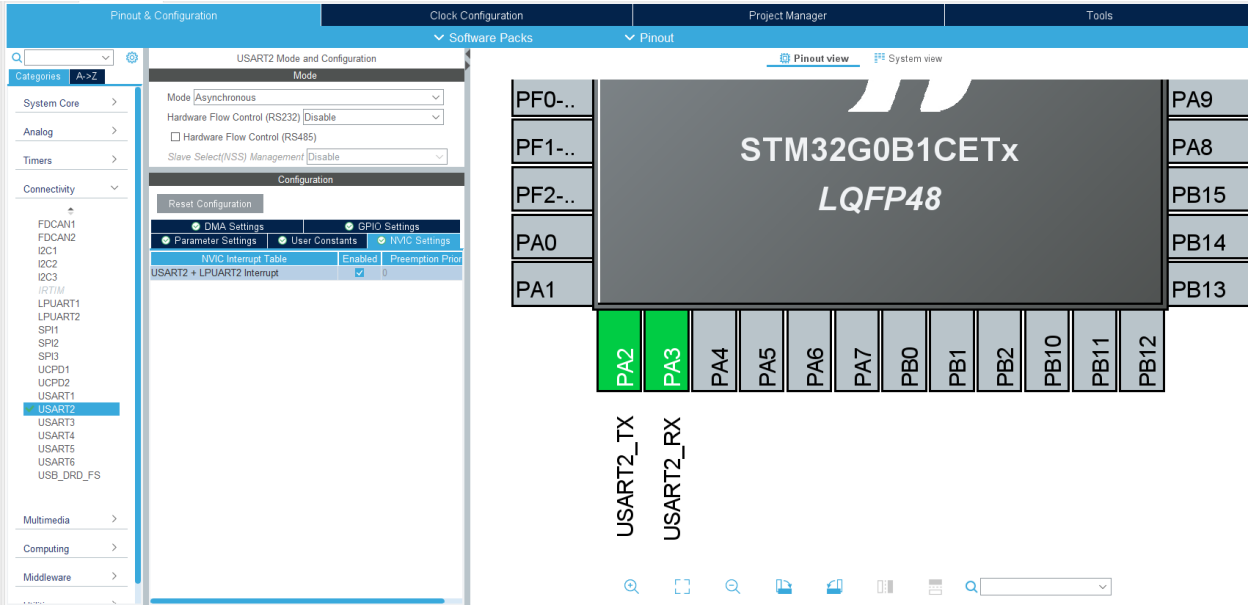
بإمكانك ا ستخدام أي أداة للتخاطب مع المنفذ التـ سلـ سلي على سبيل المثال Mobaxterm أو حتى من خلال بيئة Arduino بعد ضبط الإعدادات الأ سا سية مثل رقم المنفذ التـ سلـ سلي للحا سب المتـ صل باللوحة وأي ضاً معدل نقل البيانات BaudRate كما في الشكل التالي:





7- تطبيق عملي لاستخدام المنفذ التسلسليUSART من خلال نمط الـ interrupt:

قم بإعادة الخطوتين الأولى والثانية كما في التطبيق السابق ثم قم بتفعيل المقاطعة للمنفذ التسلسلي



الخطوة الثالثة: نقوم بضبط إعدادات ساعة النظام كما في التطبيق السابق، ثم قم بتوليد الكود من خلال الضغط على ctrl+s فيتم بتوليد الكود آلياً.

الخطوة الرابعة: قم بكتابة الكود بالشكل التالي(طبعاً هناك قسم كبير من الكود تم توليده من خلال MX وقمنا نحن فقط بإكمال الكود بما يناسب التطبيق)

المنفذ التسلسلي الذي سيتم استخدامه هو UART2 وتم تسميته بـhuart2	`#include "main.h"` `UART_HandleTypeDef huart2;`
تصريح عن مصفوفة محارف بعدد 35 محرف تصريح عن متحول ليعمل كعداد من نوع uint_8t وإسناد قيمة صفرية له كقيمة ابتدائية	`uint8_t MSG[35] = {'\0'};` `uint8_t X = 0;`
تعريف الدالة الخاصة بتهيئة ساعة النظام	`void SystemClock_Config(void);`
تعريف الدالة الخاصة بتهيئة أقطاب الدخل والخرج	`static void MX_GPIO_Init(void);`
تعريف الدالة الخاصة بتهيئة المنفذ التسلسلي	`static void MX_USART2_UART_Init(void);`
بداية البرنامج الرئيسي تهيئة مكتبة HAL	`int main(void)` `{` `  HAL_Init();`
تشكيل الجملة المراد إرسالها على المنفذ التسلسلي	`  SystemClock_Config();` `  MX_GPIO_Init();` `  MX_USART2_UART_Init();` `  sprintf(MSG, "Hello Dudes! Tracing X = %d\r\n", X);`
إرسال السلسلة MSG إلى المنفذ التسلسلي UART2	`  HAL_UART_Transmit_IT(&huart2,MSG, sizeof(MSG));`

زيادة المتحول X في كل مرة إضافة تأخير زمني بين عمليات الإرسال المتكررة بمقدار 100 ميللي ثانية الدالة التي يتم استدعائها عند إتمام عملية ارسال البيانات(قمنا بنسخ اسمها من (stm32g0xx_hal_uart.c حيث تكون معرفة كـ weak) لاحظ أننا قمنا بتغيير الحالة المنطقية لليد وإعادة إرسال البيانات مجدداً الدالة التي يذهب إليها المعالج في حال لم تتم عملية الاستقبال بشكل صحيح لأي سبب من الأسباب وضعنا هذه التعليمة فقط من أجل الـ debug.	``` while (1) { X++; HAL_Delay(100); } } void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) { HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0) ; sprintf(MSG, "Hello Dudes! Tracing X = %d\r\n", X); HAL_UART_Transmit_IT(&huart2, MSG, sizeof (MSG)); } void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart) { __NOP(); } ```