Reportes de prácticas microprocesadores y microcontroladores 2

Michel Tcheky Lepicard Fiquet

Osvaldo Vega López

Axel Arriola

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## Índice

[**Practica 1. Contador binario ascendente y descendente con leds A**](#_gjebf7vv0w4n) **2**

[**Practica 2. Carro Control remoto por Bluetooth**](#_vo45wdlv7e1f) **2**

[**Práctica 3. Comunicación con putty puerto serial A**](#_cn82zh1k71j0) **2**

[**Práctica 4. Potenciómetro Polling A**](#_pjggparwdu) **2**

[**Práctica 5. Potenciómetro Interrupciones A**](#_aryd76geoovd) **2**

[**Práctica 6. Potenciómetro DMA j**](#_ultbq85385nx) **2**

[**Práctica 7. Conexión con LCD por medio de bus i2c**](#_holjgd2a2nd0) **2**

[**Práctica 8. Carro a control remoto por bluetooth con lcd**](#_w4jbmu2leiwh) **2**

## 

## 

## 

## Practica 1. Contador binario ascendente y descendente con leds A

Introducción

El objetivo de esta práctica es crear un contador binario ascendente y descendente con 4 LEDS, por lo tanto su valor máximo valor sería 16. La finalidad de este es aprender a utilizar las conexiones de los pines GPIO de la tarjeta STM32.

Materiales

Núcleo

Protoboard

Leds (4)

Software

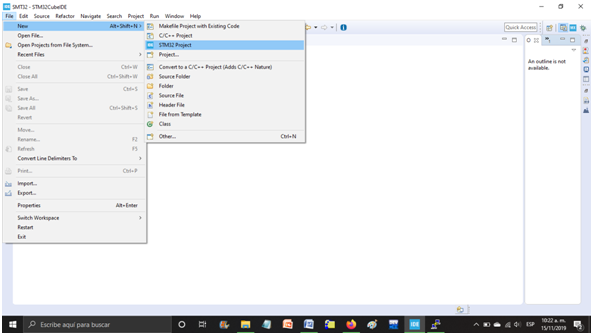
STM32CUBEIDE

STM32CUBEMX

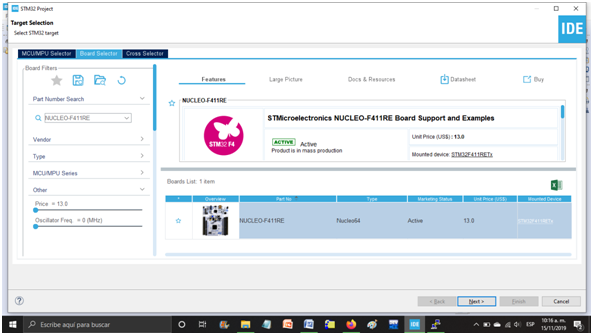
Procedimiento

**1.- Crear nuevo proyecto**

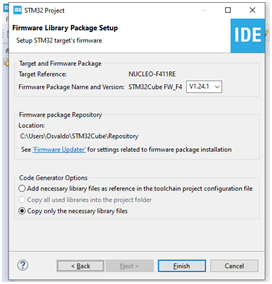
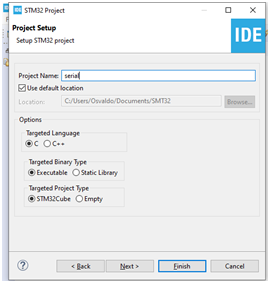
File -> New -> STM32 Project



Board Selector -> Buscar el nucleo F411RE y seleccionarlo

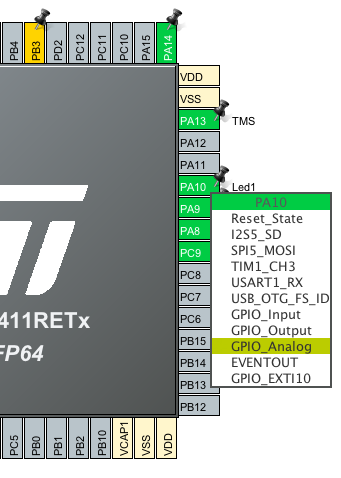


Poner nombre al proyecto -> Next ->Next ->Finish->Dar 'Yes' a todas ventanas emergentes



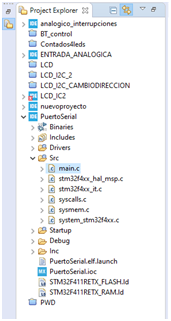
**2.- Modificar interfaz y código**

Dentro de la interfaz de STM32CUBEMX, se deben establecer los pines a usar, en este caso se usaron los pines PA10, PA9, PA8 y PC9; posteriormente dándole clic en el PIN como se muestra el la fig., se eligió GPIO\_Analog, y después dando clic secundario se agregó una etiqueta para cada uno y llamarlos Led1, Led2, Led3 y Led4 correspondientemente.



Posteriormente en STMCUBEIDE para modificar el código:

'Nombre proyecto'-> Src -> main.c



En el while del main(), modificarlo con el siguiente fragmento de código:

**int** contador = 0;

**while** (1)

**switch**(contador)

{

**case** 0:

HAL\_GPIO\_WritePin(GPIOA, LD2\_Pin|Led1\_Pin|Led2\_Pin|Led3\_Pin|Led4\_Pin, *GPIO\_PIN\_RESET*);

HAL\_Delay(1000);

**break**;

**case** 1:

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin|Led2\_Pin|Led3\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA, Led4\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 2:

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin|Led2\_Pin|Led4\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA, Led3\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 3:

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin|Led2\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA, Led4\_Pin|Led3\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 4:

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin|Led4\_Pin|Led3\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA, Led2\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 5:

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin|Led3\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA, Led4\_Pin|Led2\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 6:

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin|Led4\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA, Led2\_Pin|Led3\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 7:

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA, Led4\_Pin|Led2\_Pin|Led3\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 8:

HAL\_GPIO\_WritePin(GPIOA,Led4\_Pin|Led2\_Pin|Led3\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA,Led1\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

**case** 9:

HAL\_GPIO\_WritePin(GPIOA,Led2\_Pin|Led3\_Pin,*GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOA,Led4\_Pin|Led1\_Pin, *GPIO\_PIN\_SET*);

HAL\_Delay(800);

**break**;

}

contador++;

**if**(contador>9)

{

contador=0

}

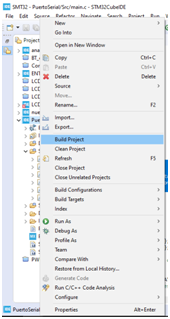
}

**return** 0;

Guardar los cambios realizados.

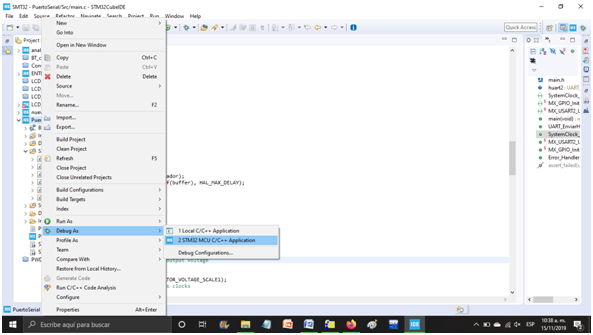
**3.- Compilar programa**

Conectar núcleo F411RE a la computadora



'Nombre proyecto' -> clic derecho -> Build Project

Si no marca error, 'Nombre proyecto' -> clic derecho -> Debug as -> 2 STM32 MCU



**4. Probar el programa**

Correr el programa ya compilado y sin errores.

## Practica 2. Carro Control remoto por Bluetooth

Introducción

El objetivo de esta práctica es poner en funcionamiento un carrito que cuenta con 2 motores de corriente directa, junto con el integrado “ “ para administrar el movimiento de las ruedas, y una conexión por medio de bluetooth para controlar el movimiento.

Materiales

Protoboard

Núcleo

Integrado l293d

Powerbank

Jumpers

Bluetooth HC06

Estructura de carro

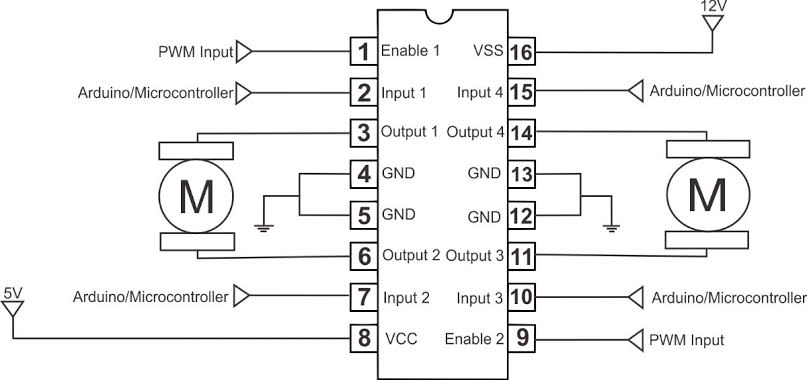
Software

STM32

Procedimiento

Comenzaremos por conectar el integrado a los motores, corriente y neutro.Es importante teneer en cuenta que la alimentación de los motores y el integrado debe ser completamente paralela al núcleo y HC06, por lo tanto se necesita la powerbank siguiendo el siguiente diagrama.

* Los pines 1, 8, 16, 9. Van conectados a corriente de 5v
* Los pines 2,7,10,15 Se conectan al núcleo
  + Renombramos estos pines a
    - MA1=2
    - MA2=7
    - MB1=15
    - MB2=10
* Pines 4,5,12,13 se conectan a GND
* Pines 3,6,11,14 corresponden a los outputs y estos estarán conectados a los motores en pares
  + 3,6
  + 11,14

****

Una vez conectado el integrado con los motores podemos proseguir con la preparación al Núcleo. existen 6 pines que debemos modificar para proseguir con el proyecto

Output

MA1

MA2

MB1

MB2

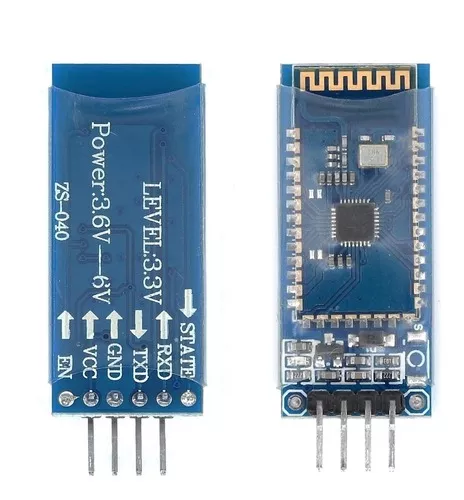
Usart

RX

TX

La necesidad de activar la conectividad por USART es para poder conectar el bluetooth, en el nuevo proyecto se debe ir a la pestaña de conectividad y activar USART2, seguido de seleccionar los pines en donde se requiere este tipo de protocolo, en este caso se seleccionó los pines PA9, PA10.

Nota importante, al conectar al dispositivo bluetooth, se deben de invertir RX-TX, TX-RX, y conectar a corriente y tierra normalmente. Estas conexiones pueden ir unidas al núcleo, así que se pueden aprovechar los pines de 5v y GND.



## 

Una vez asignados los pines podemos guardar el proyecto, y esto generará el template del código.

Código

Para hacer la lectura de nuestra conexión bluetooth necesitamos declarar una función

char Leer\_Serial(){

char entrada[1];

HAL\_UART\_Receive(&huart1, entrada, sizeof(entrada),HAL\_MAX\_DELAY);

HAL\_Delay(10);

HAL\_UART\_Transmit(&huart2, entrada,sizeof(entrada),HAL\_MAX\_DELAY);

HAL\_Delay(10);

return entrada[0];

}

y adentro del while que se encuentra en main() se necesita insertar el siguiente codigo

char lectura[1];

while (1)

{

/\* USER CODE END WHILE \*/

lectura[0]=Leer\_Serial();

if(lectura[0]=='a'){ //izq

HAL\_GPIO\_WritePin(GPIOC, MA2\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOC,MA1\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MB2\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MB1\_Pin, GPIO\_PIN\_RESET);

}

else if(lectura[0]=='b'){ //der

HAL\_GPIO\_WritePin(GPIOC,MB2\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOC,MA2\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MB1\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MA1\_Pin, GPIO\_PIN\_RESET);

}

else if(lectura[0]=='c'){ //adelante

HAL\_GPIO\_WritePin(GPIOC,MB2\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOC,MA2\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOC,MB1\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MA1\_Pin, GPIO\_PIN\_RESET);

}

else if(lectura[0]=='d'){ //atras

HAL\_GPIO\_WritePin(GPIOC,MB1\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOC,MA1\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOC,MB2\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MA2\_Pin, GPIO\_PIN\_RESET);

}

else if(lectura[0]=='e'){ //Parar

HAL\_GPIO\_WritePin(GPIOC,MB2\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MA2\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MB1\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOC,MA1\_Pin, GPIO\_PIN\_RESET);

}

}

}

Estos 5 condicionales if, se encargan de recibir una letra de la conexión bluetooth y prender a apagar los pines correspondientes, que provocan el movimiento correspondiente.

Conexiones finales

Se debe conectar el bluetooth, al núcleo teniendo en cuenta la observación anterior y conectar los pines del núcleo

* MA1=2
* MA2=7
* MB1=15
* MB2=10

Al igual que los output del integrado l293d, alimentar la protoboard con la power bank e iniciar el programa en STM32

## Práctica 3. Comunicación con putty puerto serial A

MATERIALES:

· Tarjeta nucleo F411RE

· Laptop

SOFTWARE:

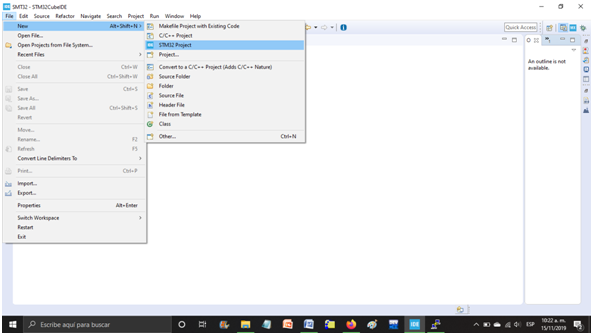
· STM32

· PuTTY

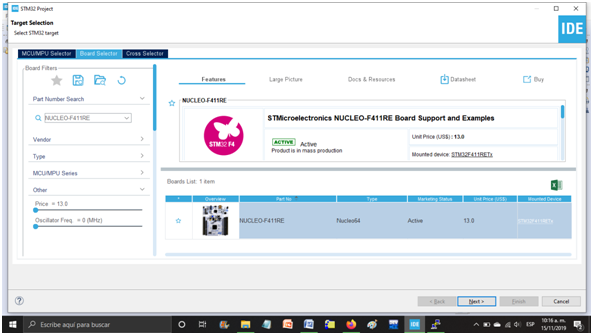
**Procedimiento**

**1.- Crear nuevo proyecto**

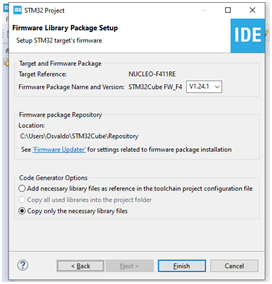
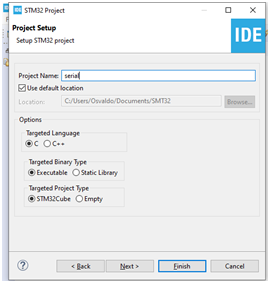
File -> New -> STM32 Project



Board Selector -> Buscar el nucleo F411RE y seleccionarlo

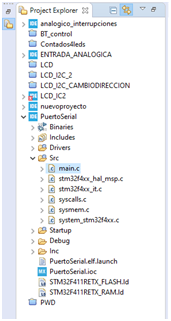


Poner nombre al proyecto -> Next ->Next ->Finish->Dar 'Yes' a todas ventanas emergentes



**2.- Modificar código**

'Nombre proyecto'-> Src -> main.c



En el while del main(), modificarlo con el siguiente fragmento de código:

**int** contador = 0;

**while** (1)

{

/\* USER CODE END WHILE \*/

UART\_EnviarHolaMundo(contador);

contador++;

**if**(contador>9){

contador = 0;

}

/\* USER CODE BEGIN 3 \*/

}

Agregar la siguiente función abajo del main():

**void** **UART\_EnviarHolaMundo**(**int** contador){

**char** buffer[50]="";

**sprintf**(buffer,"Hola Mundo %d \n\r",contador);

HAL\_UART\_Transmit(&huart2, buffer, **sizeof**(buffer), HAL\_MAX\_DELAY);

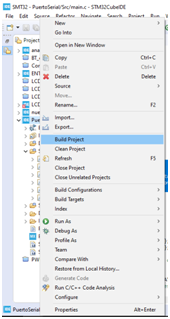
HAL\_Delay(200);

}

Guardar los cambios realizados.

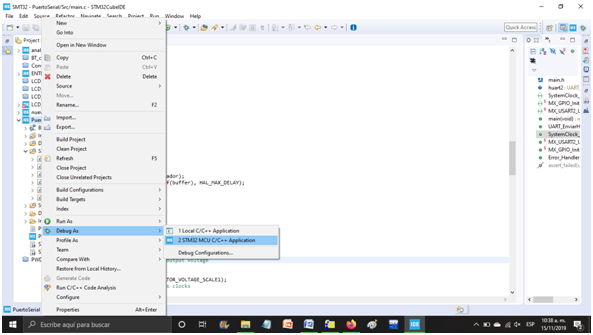
**3.- Compilar programa**

Conectar núcleo F411RE a la computadora



'Nombre proyecto' -> clic derecho -> Build Project

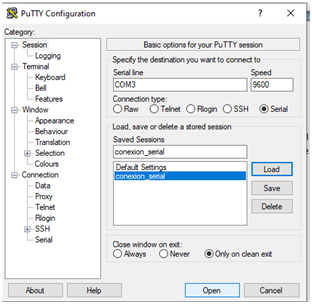
Si no marca error, 'Nombre proyecto' -> clic derecho -> Debug as -> 2 STM32 MCU



**4. Probar el programa**

Correr el programa ya compilado y sin errores.

Abrir PuTTY-> seleccionar la opción de 'serial' -> en 'serial line' indicar el puerto usb al que esté conectado la tarjeta núcleo(se puede revisar en el administrador de dispositivos de windows en los puertos COM), en 'Speed' colocar el valor 9600 o 115200 -> Open



En la pantalla que se abre se deberá mostrar un ciclo de mensajes, los cuales contiene 'Hola mundo' y un contador

## Práctica 4. Potenciómetro Polling A

**MATERIALES:**

* TARJETA NUCLEO F411RE
* Laptop
* Protoboard
* Potenciómetro
* Jumpers

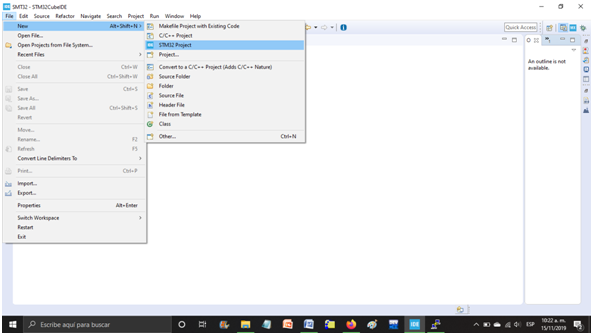
**SOFTWARE:**

* STM32
* PuTTY

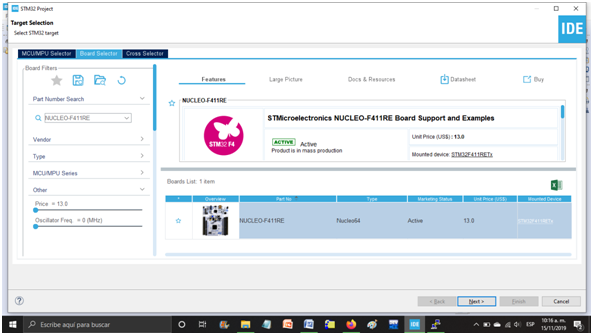
**PROCESO DE LA PRÁCTICA**

**1.- Crear nuevo proyecto**

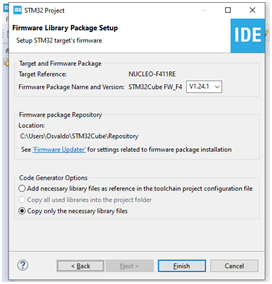
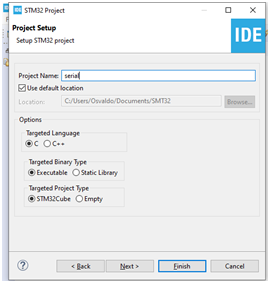
File -> New -> STM32 Project



Board Selector -> Buscar el nucleo F411RE y seleccionarlo

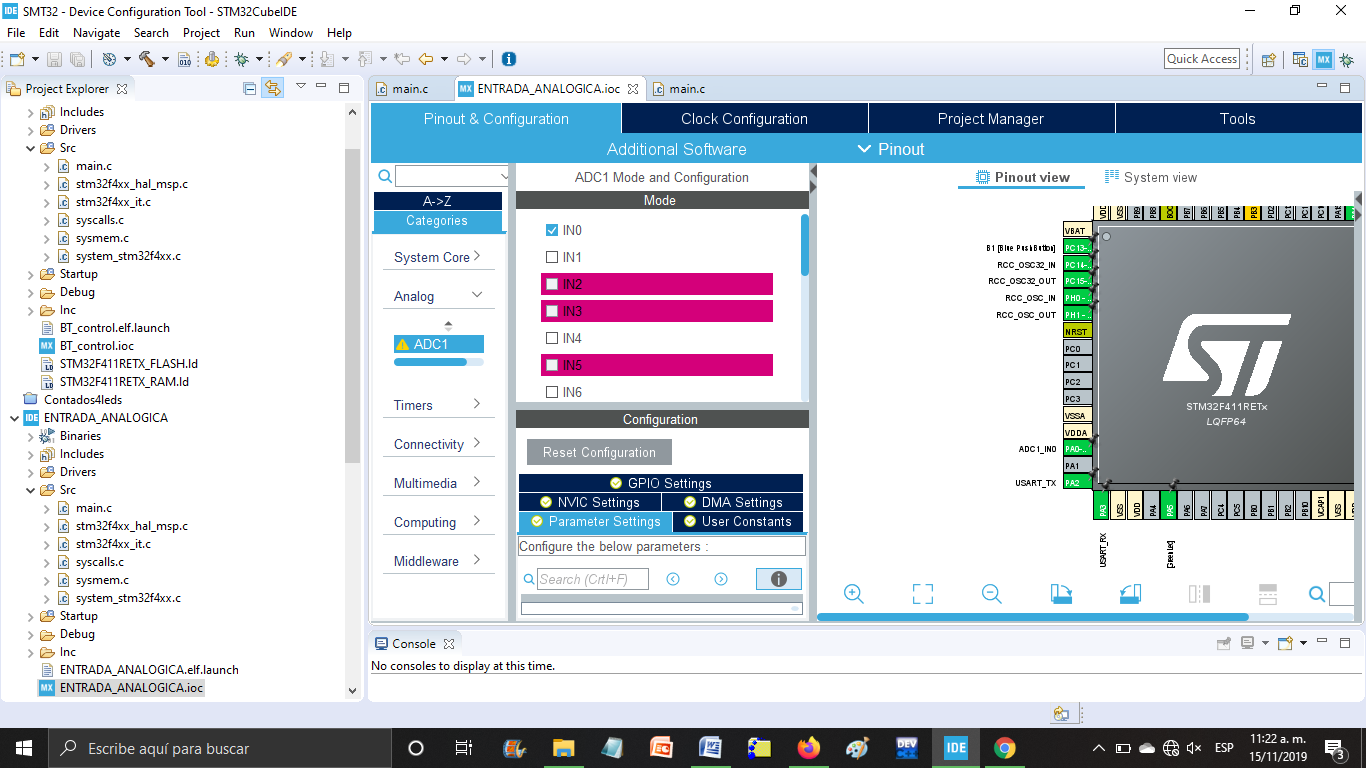


Poner nombre al proyecto -> Next ->Next ->Finish->Dar 'Yes' a todas ventanas emergentes



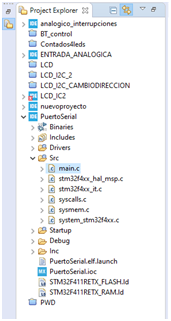
**2.- Configurar la tarjeta**

Analog -> ADC1 -> activar IN0



**3.- Modificar código**

'Nombre proyecto'-> Src -> main.c



* Verificar que al inicio del código estén declaradas las siguientes variables, si llega a faltar una agregarla:

ADC\_HandleTypeDef hadc1;

UART\_HandleTypeDef huart2;

**void** **imprimir**(**float** valor2);

uint32\_t valor;

**float** voltaje;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_USART2\_UART\_Init**(**void**);

**static** **void** **MX\_ADC1\_Init**(**void**);

* Agregar la siguiente función arriba del main():

**void** **imprimir**(**float** valor2){

**char** buffer[50]="";

**sprintf**(buffer," %f \n\r",valor2); // @suppress("Float formatting support")

HAL\_UART\_Transmit(&huart2, buffer, **sizeof**(buffer), HAL\_MAX\_DELAY);

HAL\_Delay(200);

}

* Modificar el contenido del main() con el siguiente código:

/\* 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();

MX\_USART2\_UART\_Init();

MX\_ADC1\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

HAL\_ADC\_Start(&hadc1);

HAL\_ADC\_PollForConversion(&hadc1,1000);

valor = HAL\_ADC\_GetValue(&hadc1);

voltaje = (3.3/4096) \* valor;

imprimir(valor);

HAL\_ADC\_Stop(&hadc1);

/\* USER CODE END WHILE \*/

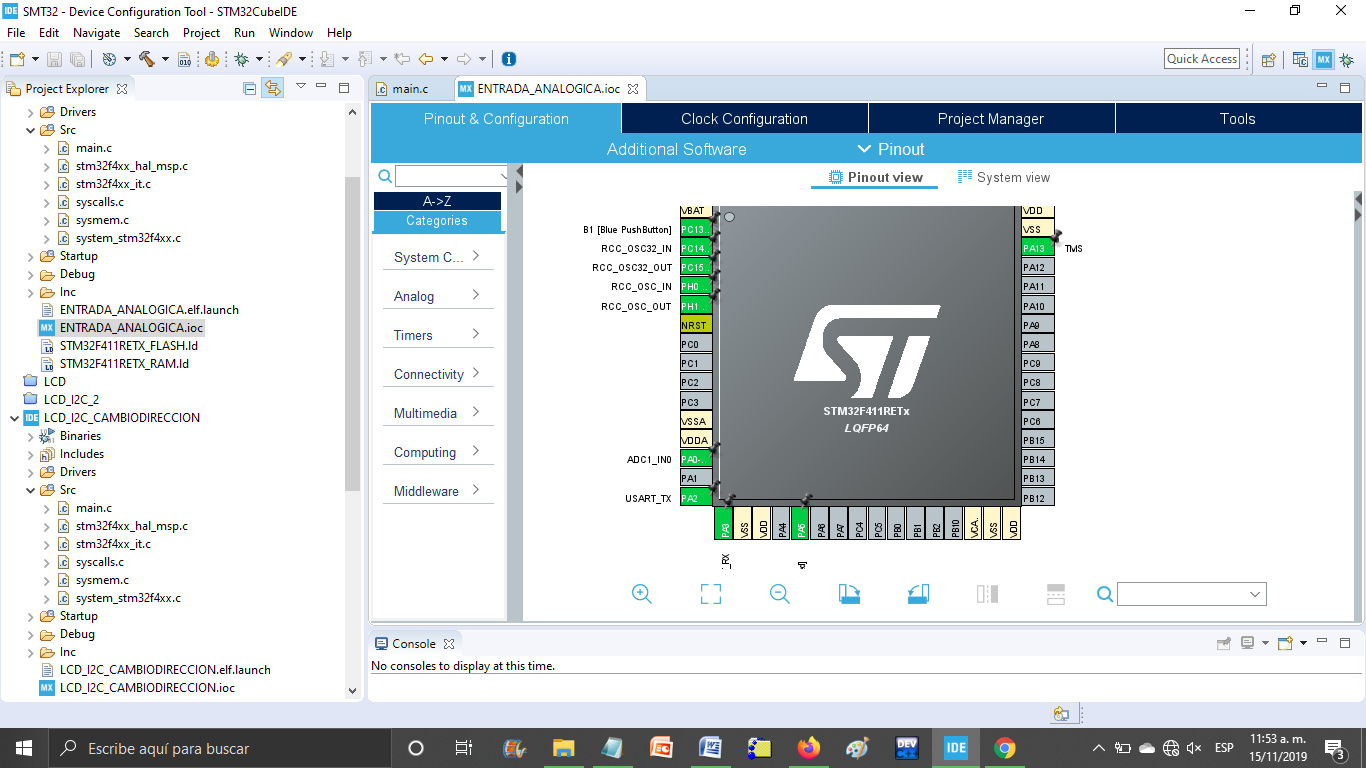
/\* USER CODE BEGIN 3 \*/

}

Guardar cambios.

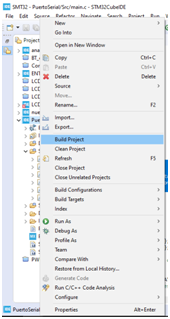
**4. Armar el protoboard**

* Poner el potenciómetro en el protoboard.
* Uno de los extremos del potenciómetro conectarlo a tierra de la tarjeta núcleo.
* Otro de los extremos del protoboard conectarlo a voltaje por medio de una resistencia a la tarjeta núcleo.
* La pata de en medio del potenciómetro conectarla al puerto de la tarjeta núcleo que está indicado en la configuración de la tarjeta del STM32, el puerto debe tener el nombre de ADC1\_IN0:



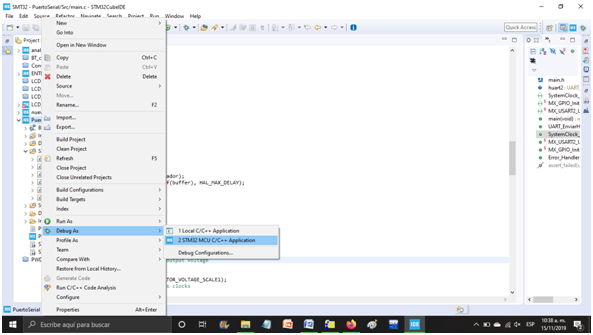
**5.- Compilar programa**

Conectar la núcleo F411RE a la computadora



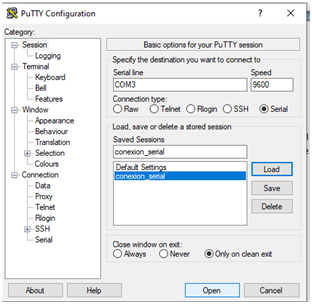
'Nombre proyecto' -> clic derecho -> Build Project

Si no marca error, 'Nombre proyecto' -> clic derecho -> Debug as -> 2 STM32 MCU



**6. Probar el programa**

* Correr el programa ya compilado y sin errores.
* Abrir PuTTY-> seleccionar la opción de 'serial' -> en 'serial line' indicar el puerto usb al que esté conectado la tarjeta núcleo(se puede revisar en el administrador de dispositivos de windows en los puertos COM), en 'Speed' colocar el valor 9600 o 115200 -> Open



En la pantalla que se abre se deberá mostrar un ciclo de números que cambiarán al momento de que se esté moviendo el potenciómetro.

## Práctica 5. Potenciómetro Interrupciones A

**MATERIALES:**

* TARJETA NUCLEO F411RE
* Laptop
* Protoboard
* Potenciómetro
* Jumpers

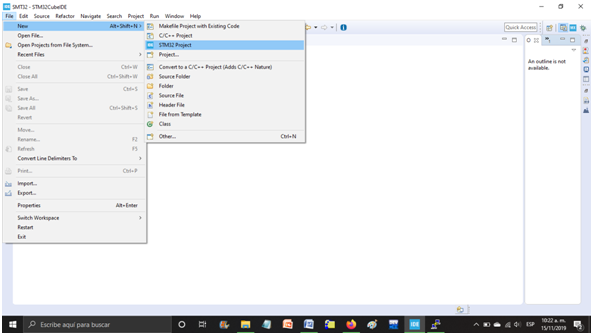
**SOFTWARE:**

* STM32
* PuTTY

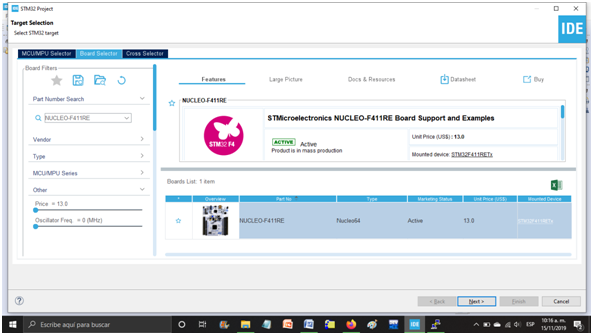
**PROCESO DE LA PRÁCTICA**

**1.- Crear nuevo proyecto**

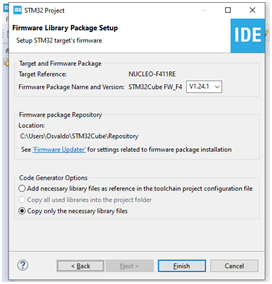
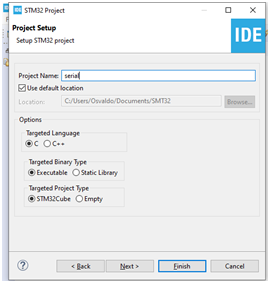
File -> New -> STM32 Project



Board Selector -> Buscar el nucleo F411RE y seleccionarlo

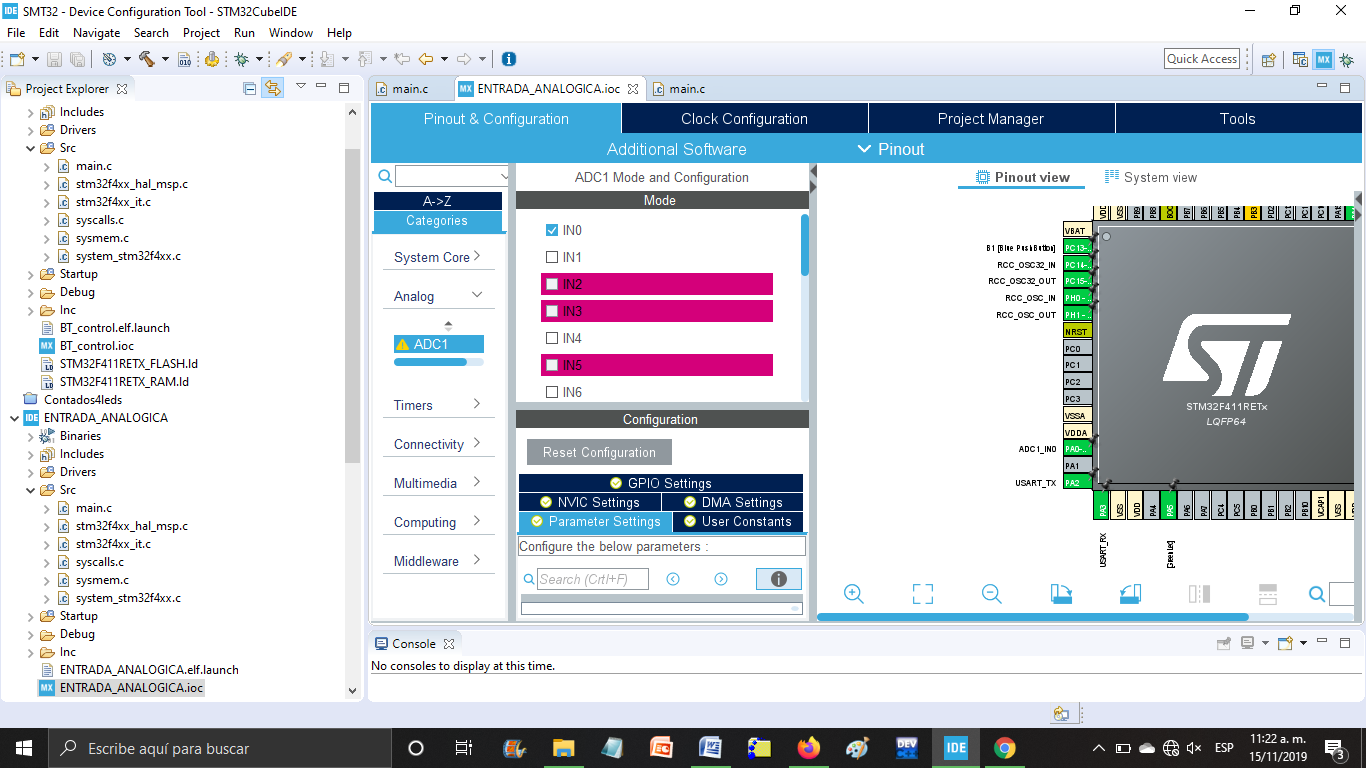


Poner nombre al proyecto -> Next ->Next ->Finish->Dar 'Yes' a todas ventanas emergentes

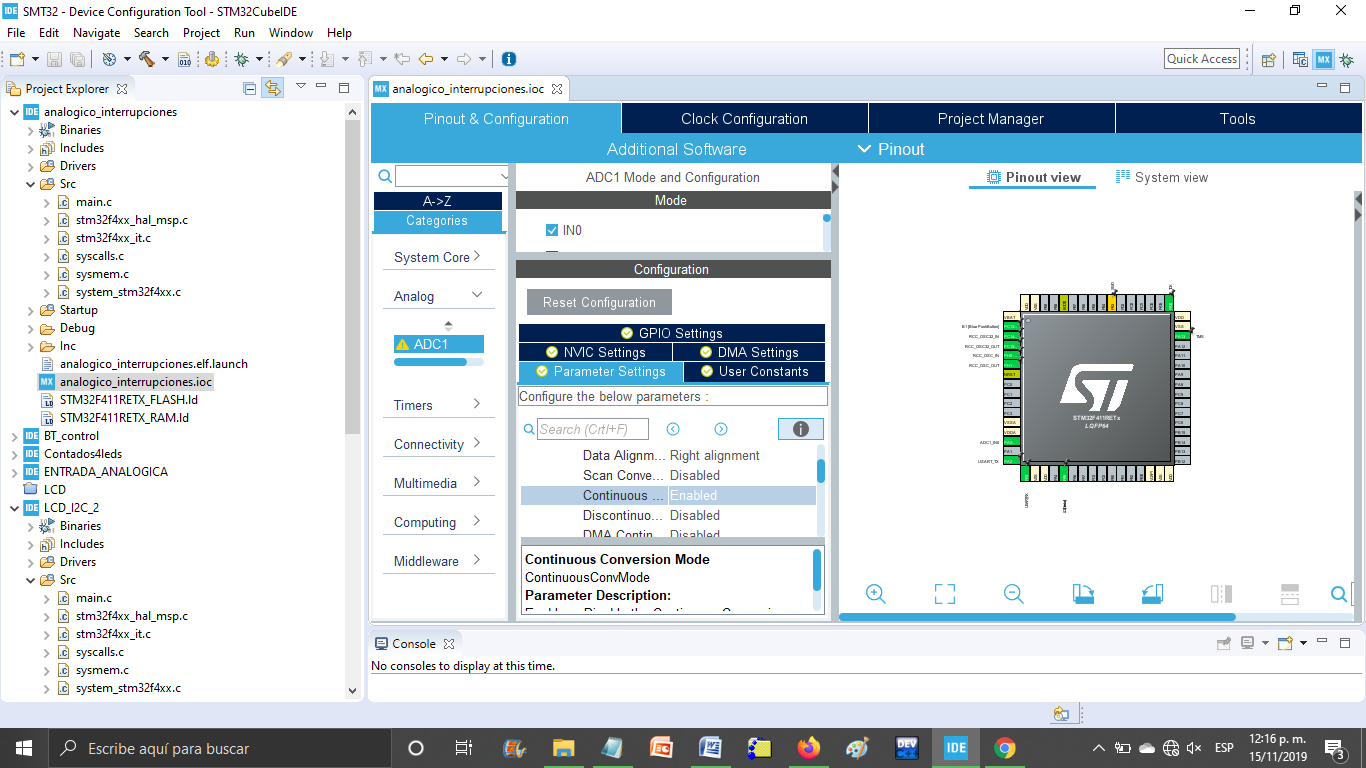


**2.- Configurar la tarjeta**

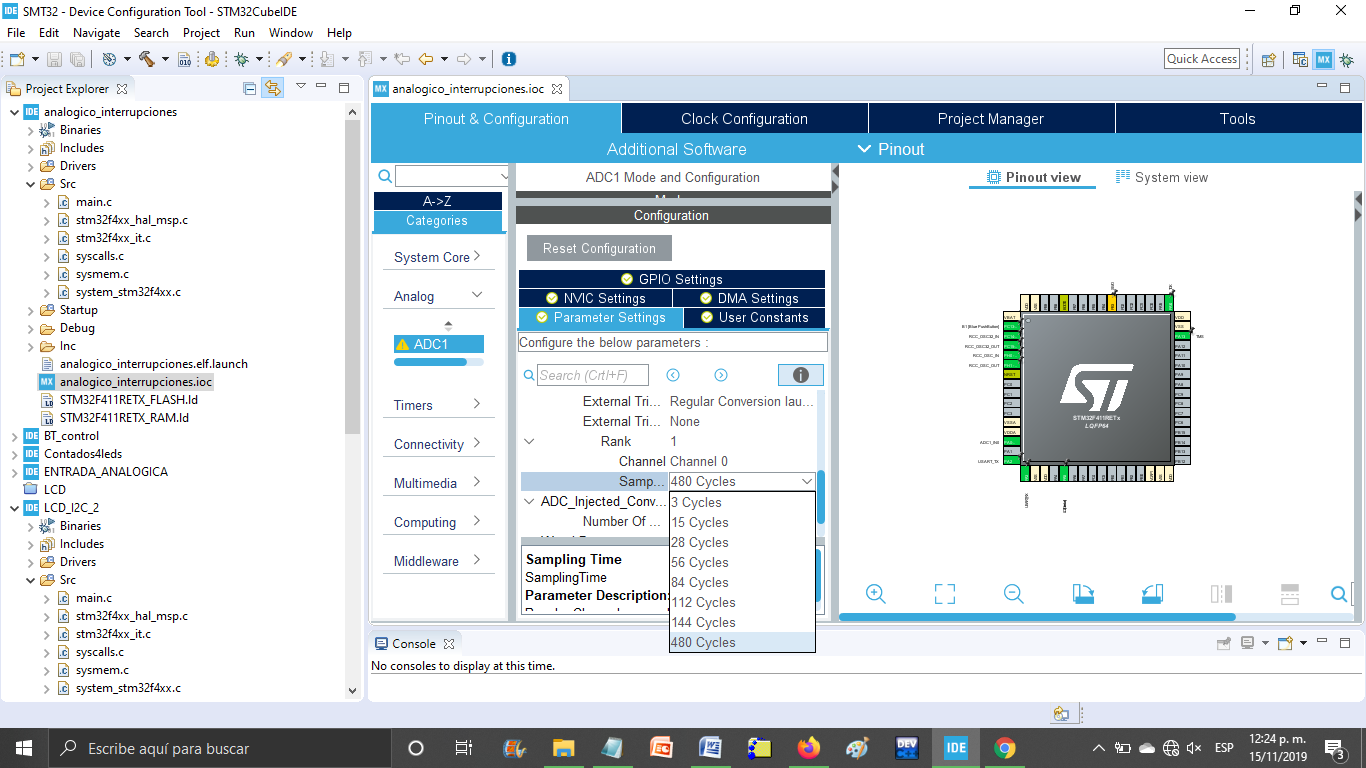
Analog -> ADC1 -> activar IN0



Seleccionar IN0 -> Parameter Settings -> ADC Settings -> Habilitar modo continuo



Seleccionar IN0 -> Parameter Settings -> ADC\_Regular\_ConversionMode->Rank->Aumentar número de ciclos a 480

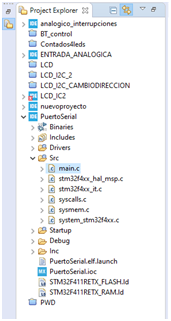


Seleccionar IN0 -> NVIC Settings -> Habilitar ADC1 globlal interruption



**3.- Modificar código**

'Nombre proyecto'-> Src -> main.c



* Verificar que al inicio del código estén declaradas las siguientes variables, si llega a faltar una agregarla:

ADC\_HandleTypeDef hadc1;

UART\_HandleTypeDef huart2;

uint16\_t valor;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_USART2\_UART\_Init**(**void**);

**static** **void** **MX\_ADC1\_Init**(**void**);

**void** **imprimir**(uint16\_t valor2);

* Modificar el contenido del main() con el siguiente código:

/\* 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();

MX\_USART2\_UART\_Init();

MX\_ADC1\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_IT(&hadc1);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

imprimir(valor);

HAL\_Delay(100);

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

Agregar las siguientes dos funciones abajo del main():

**void** **HAL\_ADC\_ConvCpltCallback**(ADC\_HandleTypeDef \*hadc){

valor = HAL\_ADC\_GetValue(&hadc1);

}

/\* USER CODE BEGIN 4 \*/

**void** **imprimir**(uint16\_t valor2){

**char** buffer[50]="";

**sprintf**(buffer," %d \n\r",valor2); // @suppress("Float formatting support")

HAL\_UART\_Transmit(&huart2, buffer, **sizeof**(buffer), HAL\_MAX\_DELAY);

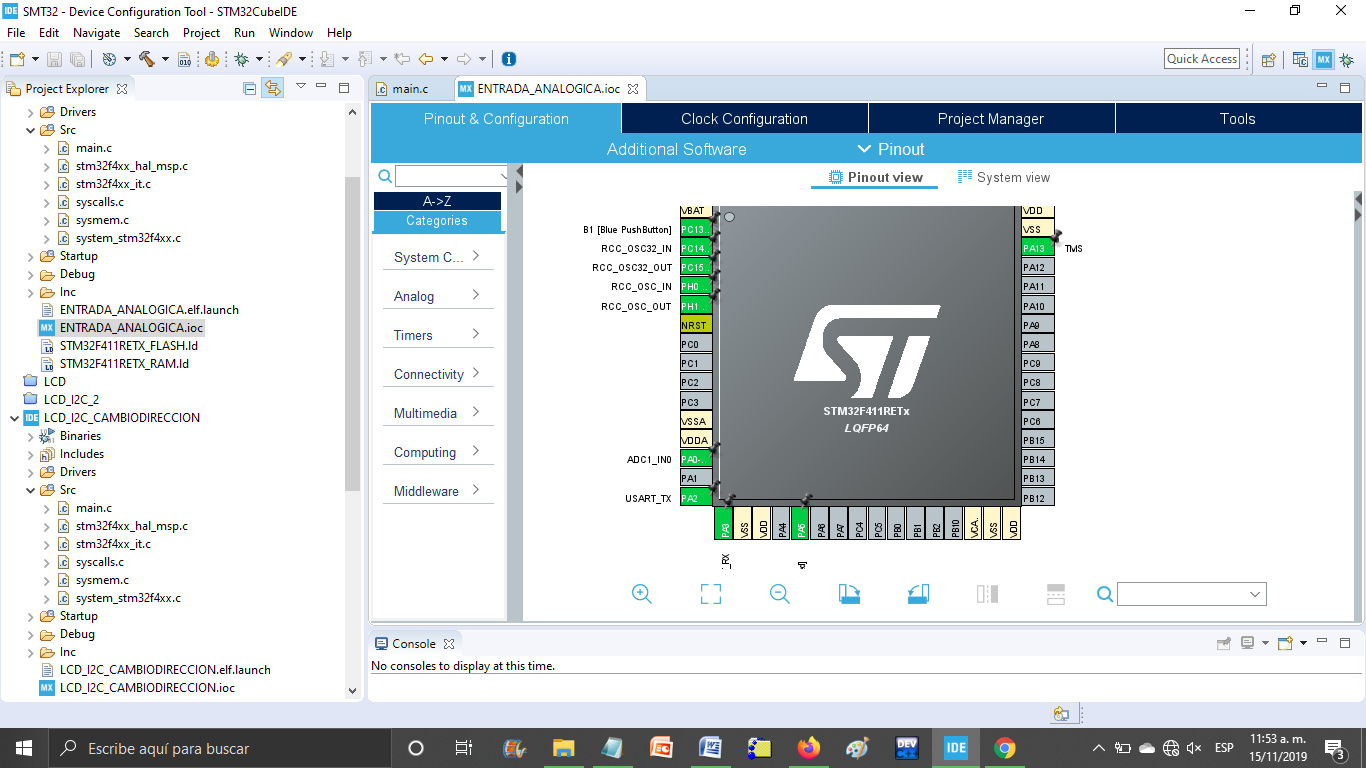
//HAL\_Delay(200);

}

* Guardar los cambios.

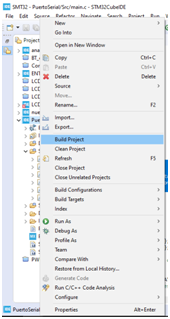
**4. Armar el protoboard**

* Poner el potenciómetro en el protoboard.
* Uno de los extremos del potenciómetro conectarlo a tierra de la tarjeta núcleo.
* Otro de los extremos del protoboard conectarlo a voltaje por medio de una resistencia a la tarjeta núcleo.
* La pata de en medio del potenciómetro conectarla al puerto de la tarjeta núcleo que está indicado en la configuración de la tarjeta del STM32, el puerto debe tener el nombre de ADC1\_IN0:



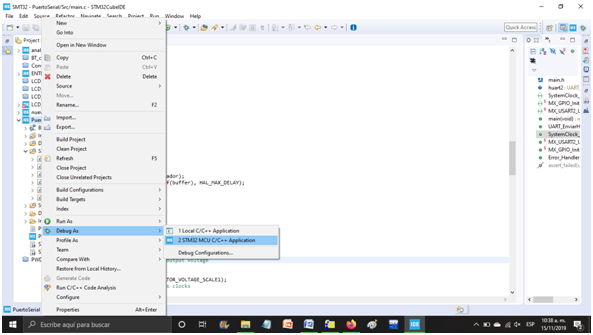
**5.- Compilar programa**

Conectar la núcleo F411RE a la computadora



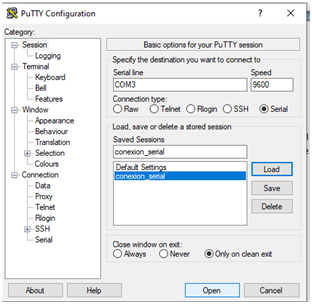
'Nombre proyecto' -> clic derecho -> Build Project

Si no marca error, 'Nombre proyecto' -> clic derecho -> Debug as -> 2 STM32 MCU



**6. Probar el programa**

* Correr el programa ya compilado y sin errores.
* Abrir PuTTY-> seleccionar la opción de 'serial' -> en 'serial line' indicar el puerto usb al que esté conectado la tarjeta núcleo(se puede revisar en el administrador de dispositivos de windows en los puertos COM), en 'Speed' colocar el valor 9600 o 115200 -> Open



En la pantalla que se abre se deberá mostrar un ciclo de números que cambiarán al momento de que se esté moviendo el potenciómetro.

## Práctica 6. Potenciómetro DMA j

**MATERIALES:**

* TARJETA NUCLEO F411RE
* Laptop
* Protoboard
* Potenciómetro
* Jumpers

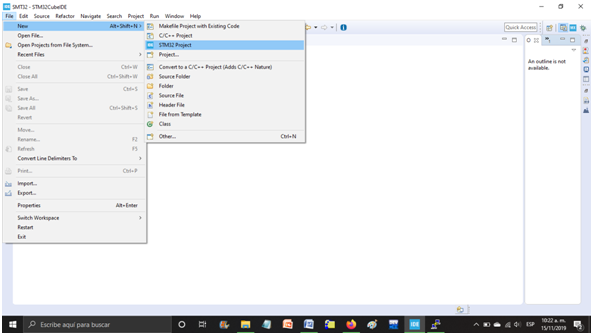
**SOFTWARE:**

* STM32
* PuTTY

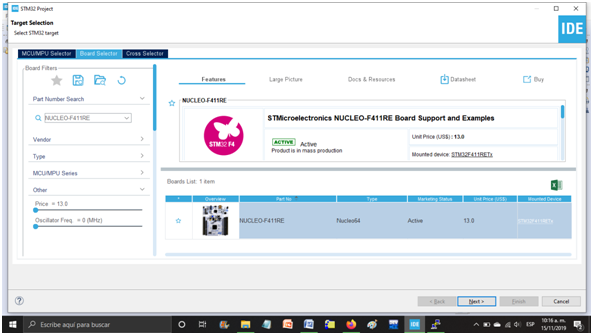
**PROCESO DE LA PRÁCTICA**

**1.- Crear nuevo proyecto**

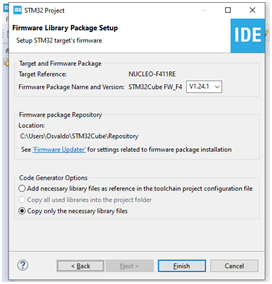
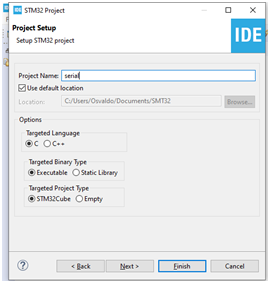
File -> New -> STM32 Project



Board Selector -> Buscar el nucleo F411RE y seleccionarlo

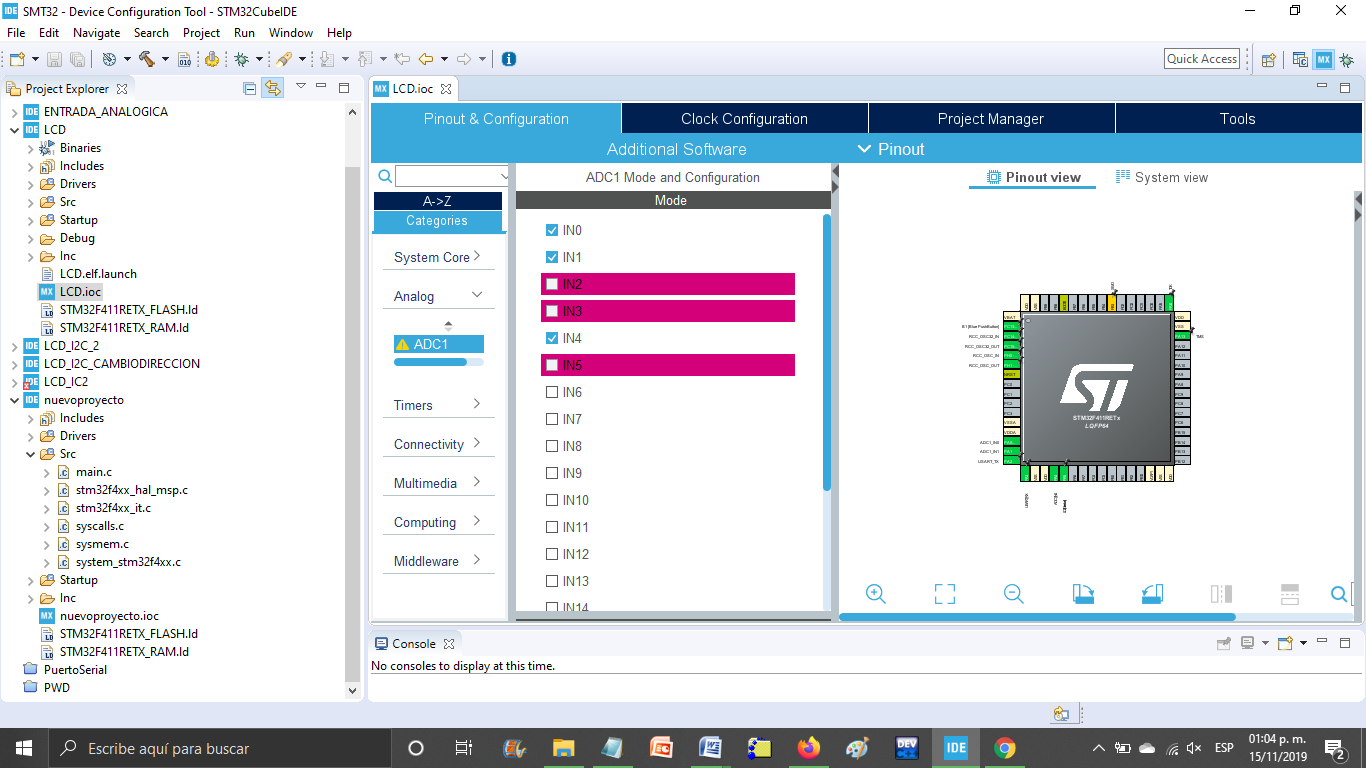


Poner nombre al proyecto -> Next ->Next ->Finish->Dar 'Yes' a todas ventanas emergentes

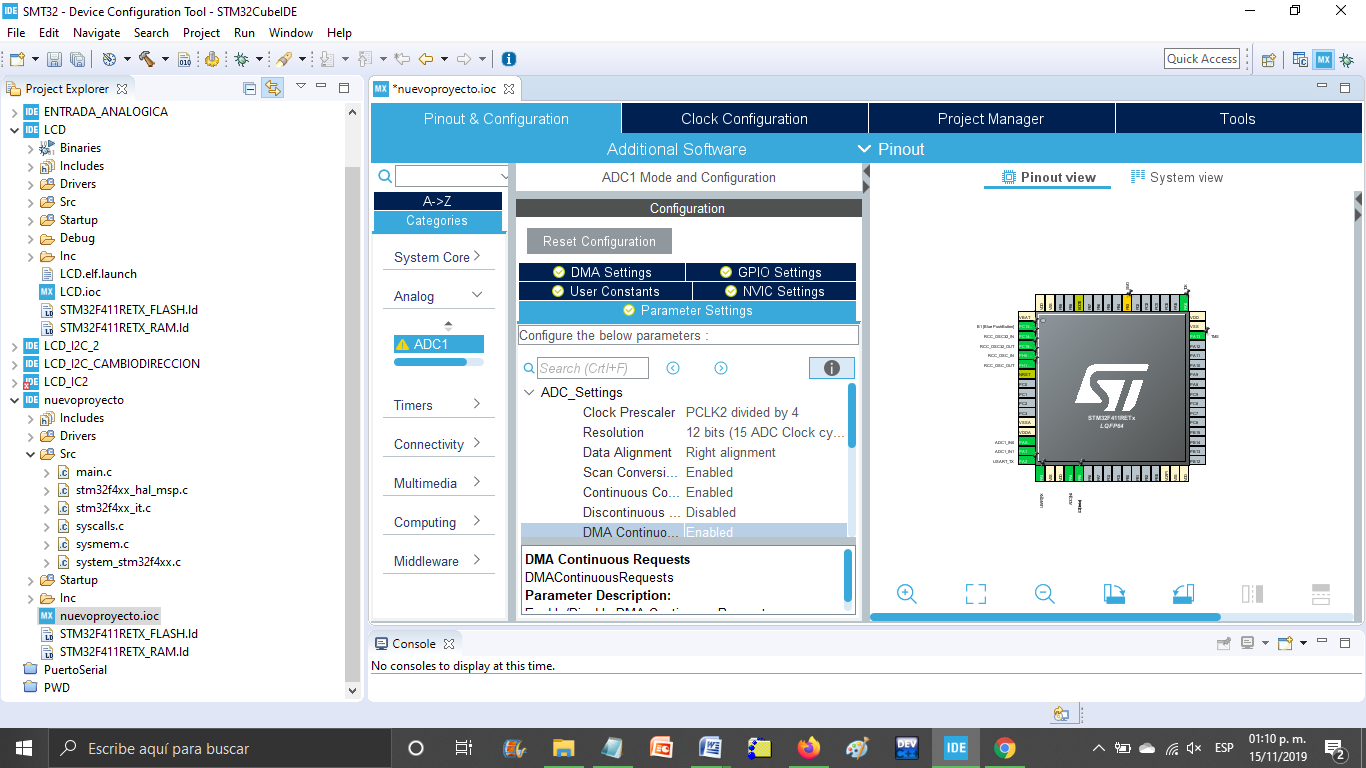


**2.- Configurar la tarjeta**

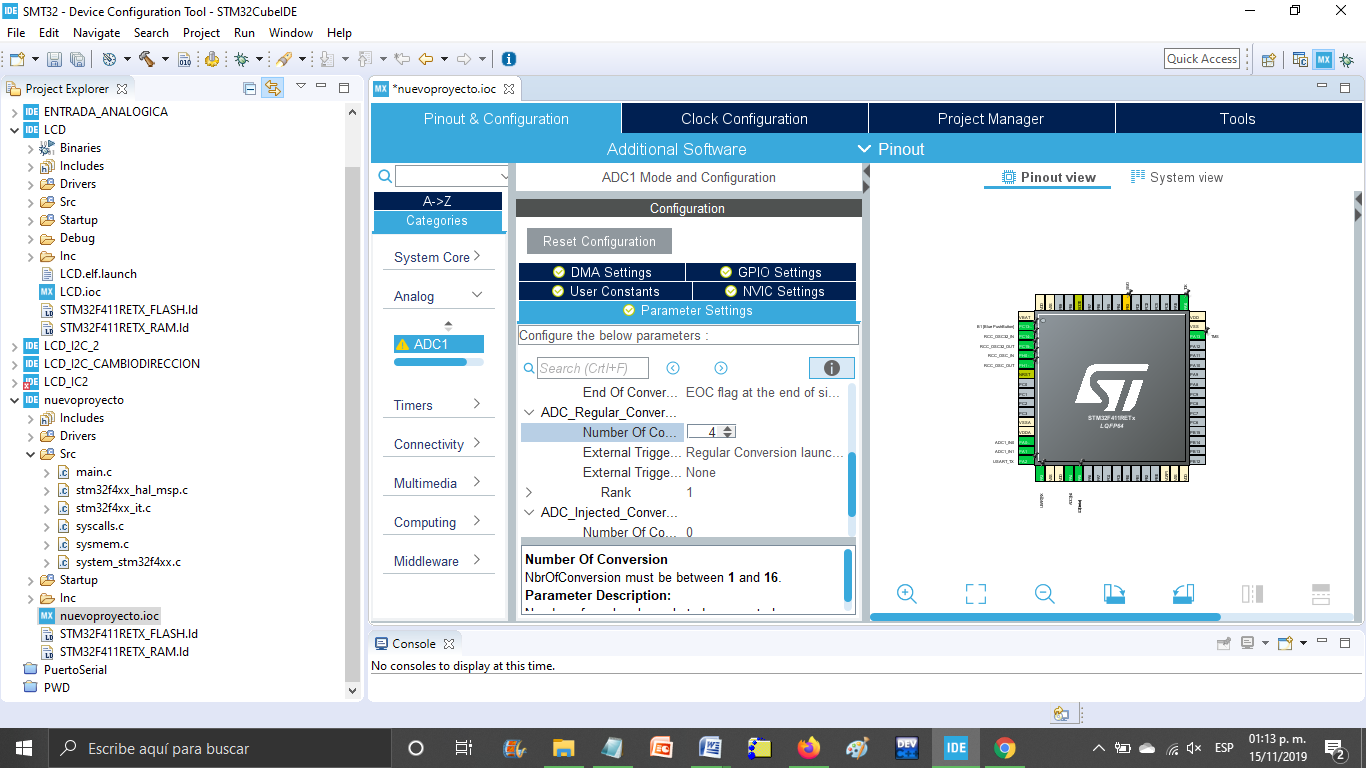
Analog -> ADC1 -> activar IN0,IN1,IN4, Temperature Sensor Channel



Seleccionar IN0,IN1,IN4,temperature -> Parameter Settings -> ADC Settings -> Habilitar modo continuo,modo scan y modo DMA



Seleccionar IN0,IN1,IN4,temperature -> Parameter Settings -> ADC\_Regular\_ConversionMode->Cambiar el número de conversion a 4



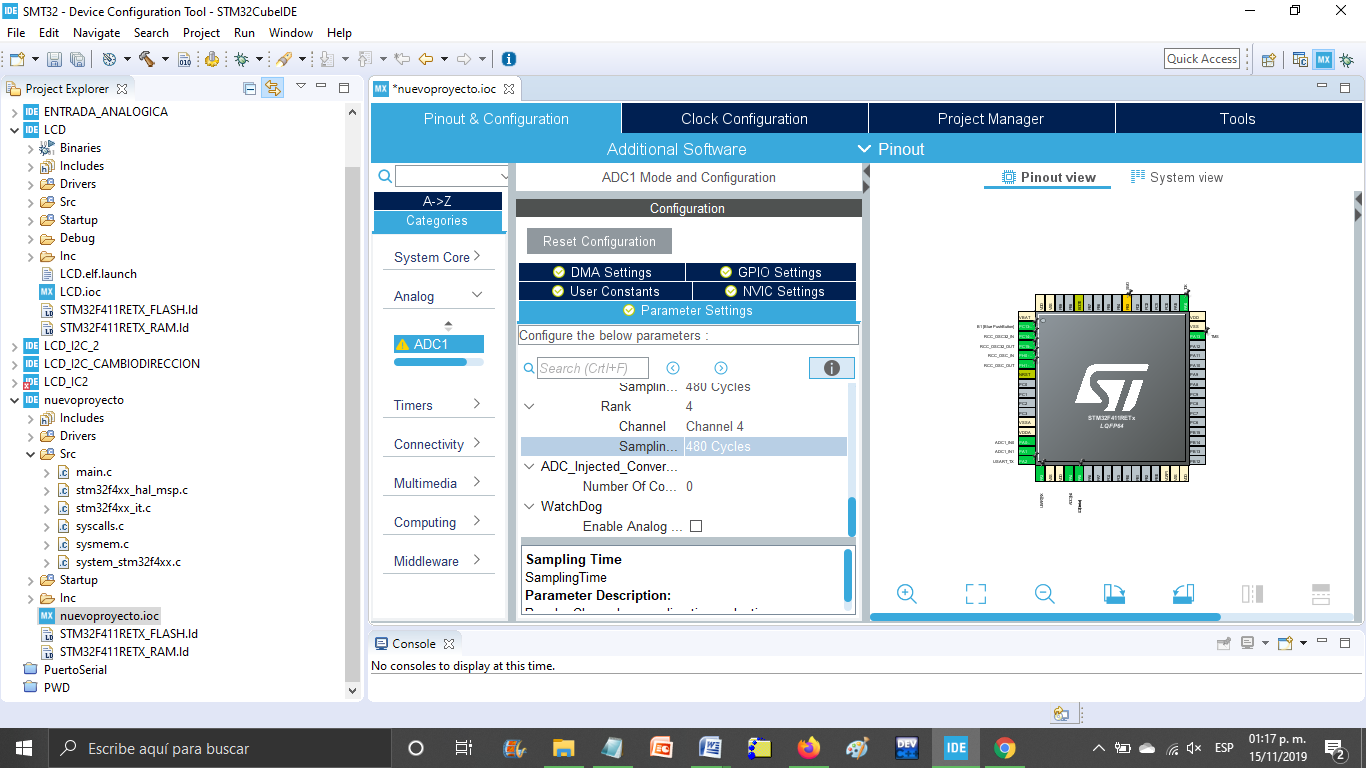
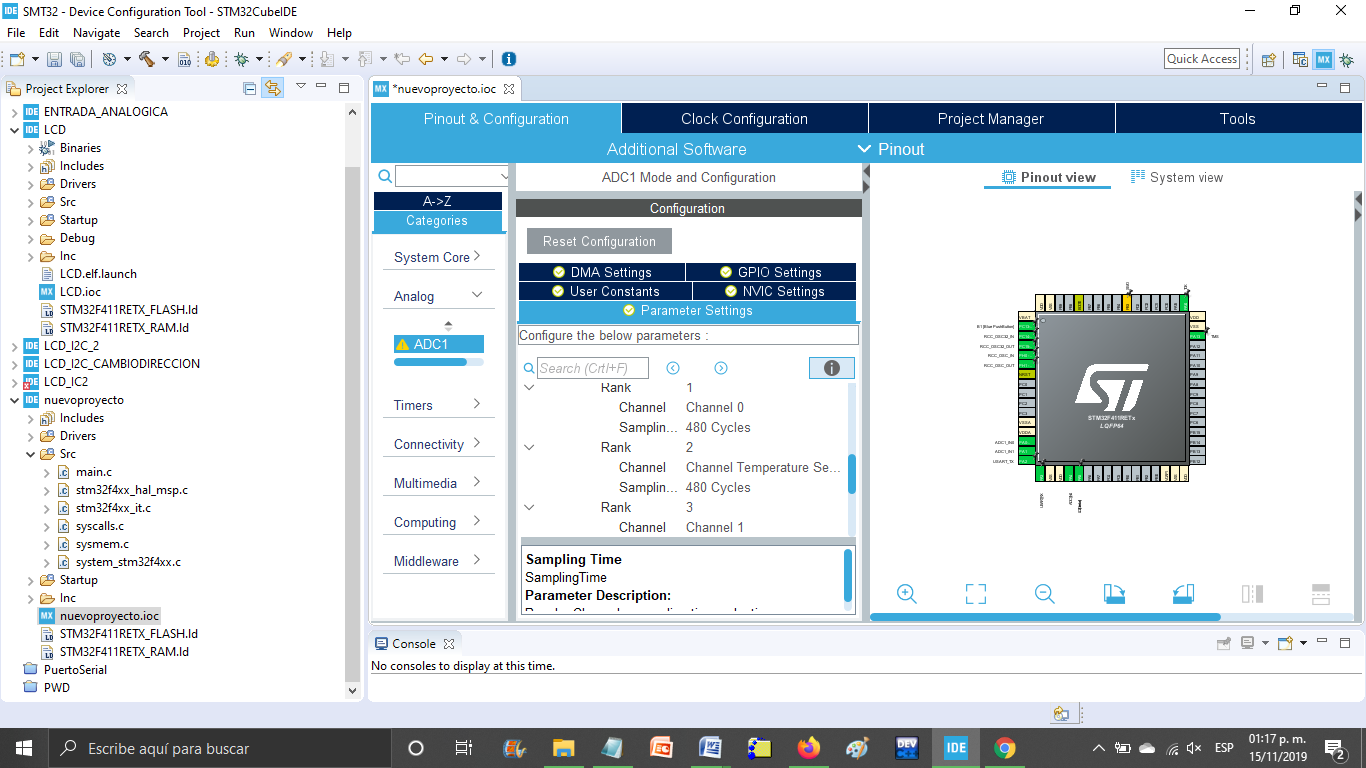
Seleccionar IN0,IN1,IN4,temperature -> Parameter Settings -> ADC\_Regular\_ConversionMode->Cambiar los Ranks a 480 ciclos y asignarlos a los siguientes canales:

Rank1-Channel0

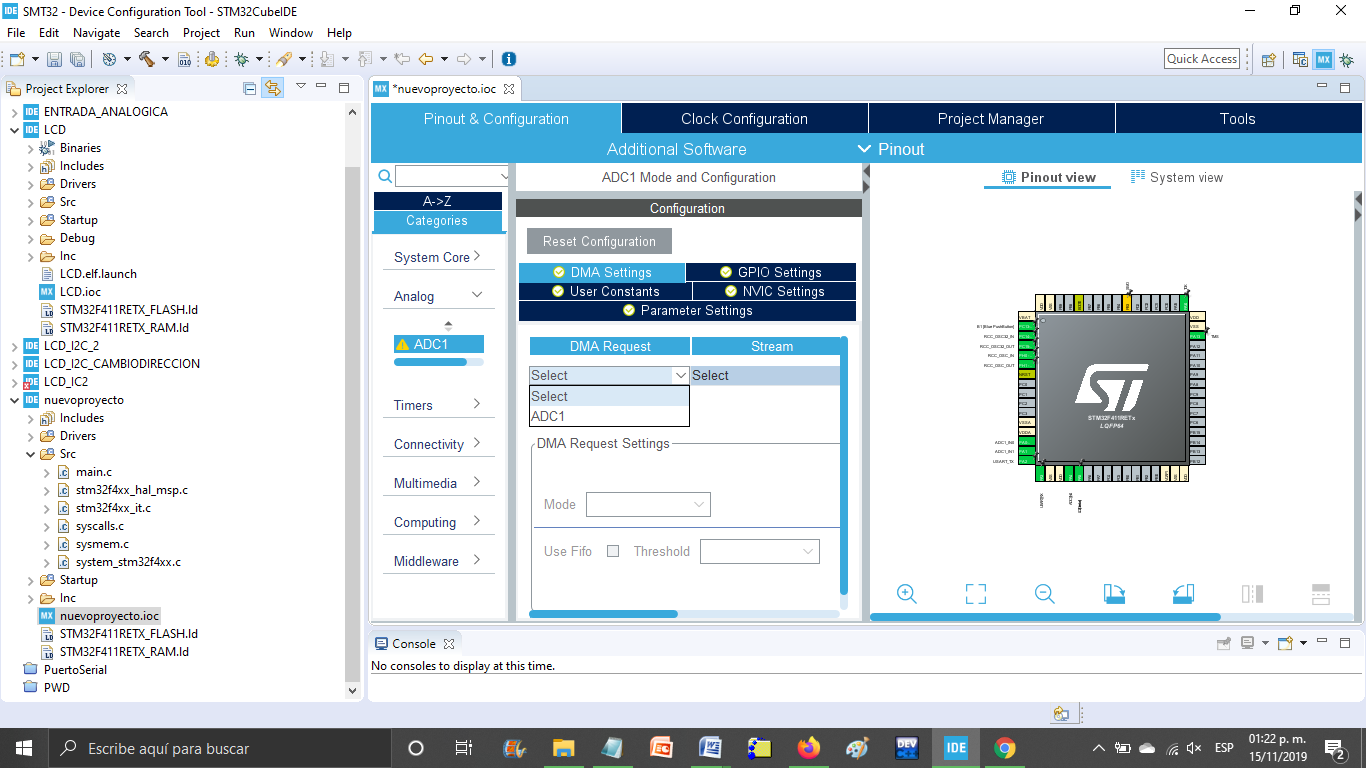
Rank2-temperature

Rank3-Channel1

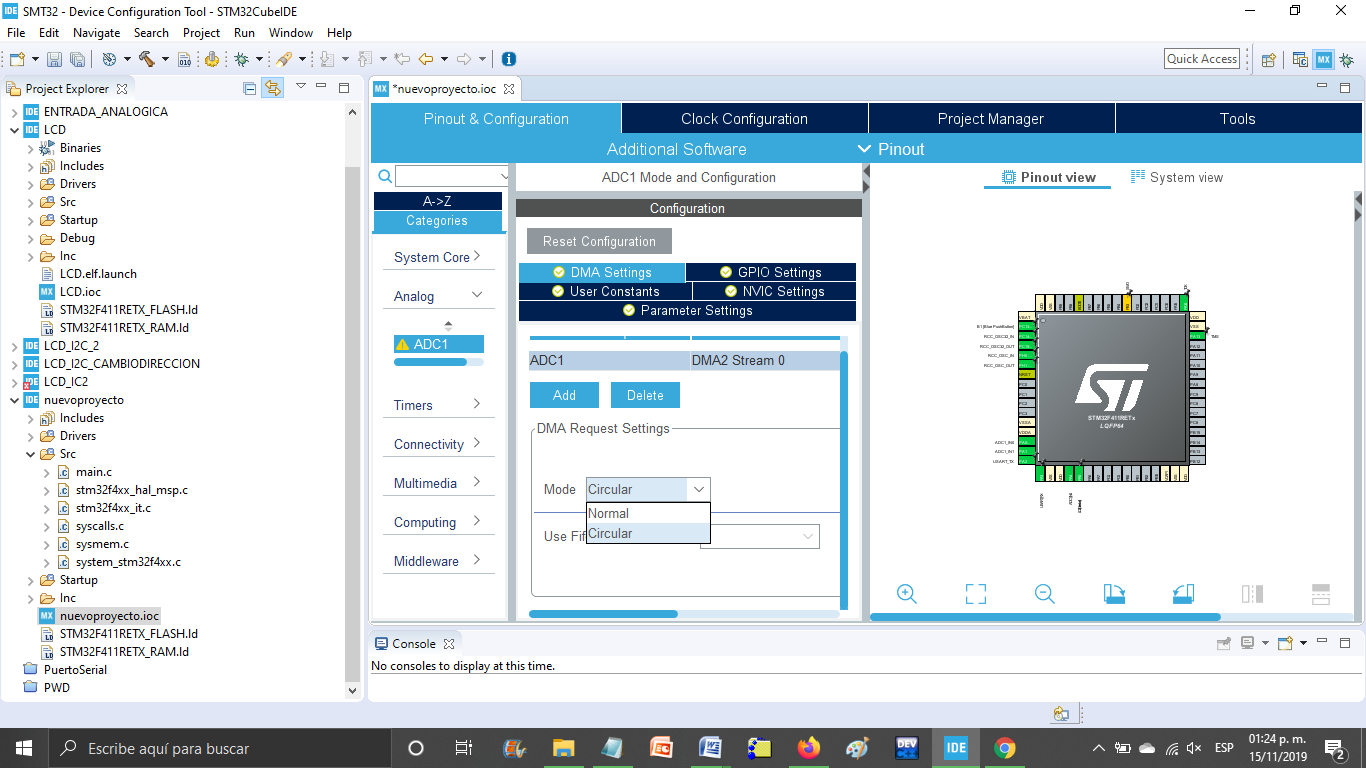
Rank4-Channel2



Seleccionar IN0,IN1,IN4,temperature -> DMA Settings->clic en ‘Add’->Añadir ADC1

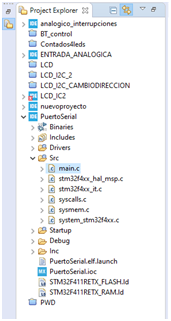


Seleccionar IN0,IN1,IN4,temperature -> DMA Settings->seleccionar ADC1 -> cambiar modo a ‘Circular’



**3.- Modificar código**

'Nombre proyecto'-> Src -> main.c



* Verificar que al inicio del código estén declaradas las siguientes variables, si llega a faltar una agregarla:

ADC\_HandleTypeDef hadc1;

DMA\_HandleTypeDef hdma\_adc1;

uint16\_t adcBuffer1[4];

UART\_HandleTypeDef huart2;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_DMA\_Init**(**void**);

**static** **void** **MX\_USART2\_UART\_Init**(**void**);

**static** **void** **MX\_ADC1\_Init**(**void**);

* Modificar el contenido del main() con el siguiente código:

/\* 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();

MX\_DMA\_Init();

MX\_USART2\_UART\_Init();

MX\_ADC1\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_DMA(&hadc1,(uint16\_t\*)adcBuffer1,4);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

imprimir(adcBuffer1[0]);

imprimir(adcBuffer1[1]);

imprimir(adcBuffer1[2]);

imprimir(adcBuffer1[3]);

HAL\_Delay(1000);

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

* Agregar las siguientes dos funciones abajo del main():

**void** **imprimir**(uint16\_t valor2){

**char** buffer[50]="";

**sprintf**(buffer," %d \n\r",valor2); // @suppress("Float formatting support")

HAL\_UART\_Transmit(&huart2, buffer, **sizeof**(buffer), HAL\_MAX\_DELAY);

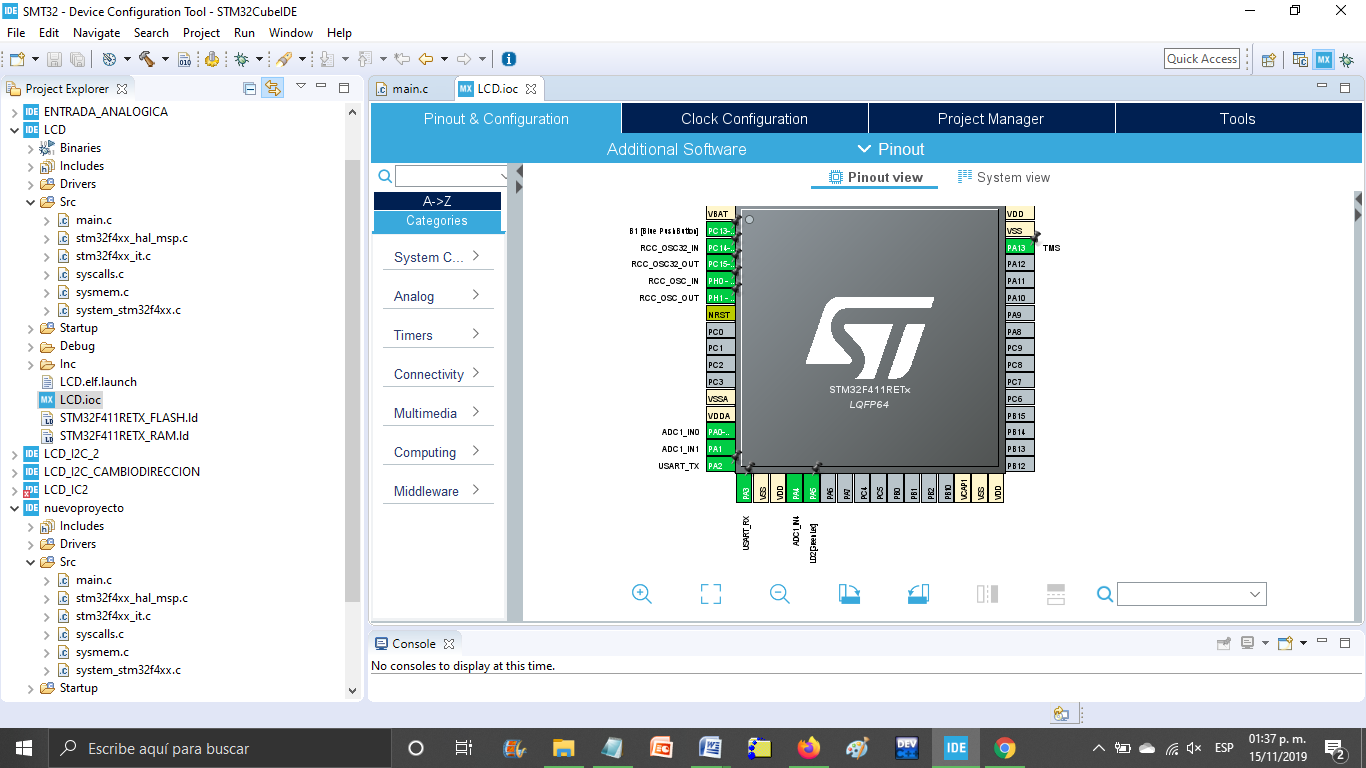
//HAL\_Delay(200);

}

* Guardar los cambios.

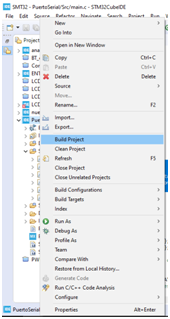
**4. Armar el protoboard**

* Poner el potenciómetro en el protoboard.
* Uno de los extremos del potenciómetro conectarlo a tierra de la tarjeta núcleo.
* Otro de los extremos del protoboard conectarlo a voltaje por medio de una resistencia a la tarjeta núcleo.
* La pata de en medio del potenciómetro conectarla a los dos puertos de la tarjeta núcleo que está indicado en la configuración de la tarjeta del STM32, los puertos deben tener el nombre de ADC1\_IN0 y ADC1\_IN1:



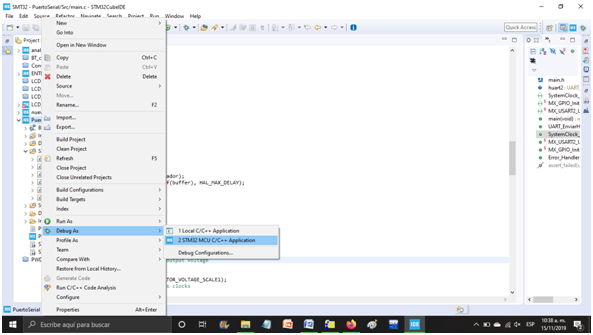
**5.- Compilar programa**

Conectar la núcleo F411RE a la computadora



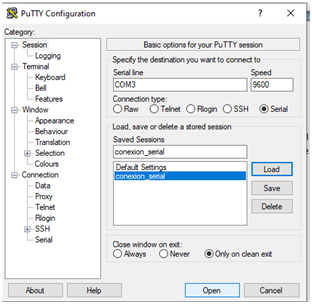
'Nombre proyecto' -> clic derecho -> Build Project

Si no marca error, 'Nombre proyecto' -> clic derecho -> Debug as -> 2 STM32 MCU



**6. Probar el programa**

* Correr el programa ya compilado y sin errores.
* Abrir PuTTY-> seleccionar la opción de 'serial' -> en 'serial line' indicar el puerto usb al que esté conectado la tarjeta núcleo(se puede revisar en el administrador de dispositivos de windows en los puertos COM), en 'Speed' colocar el valor 9600 o 115200 -> Open



En la pantalla que se abre se deberá mostrar un ciclo de números que cambiarán al momento de que se esté moviendo el potenciómetro.

’

## Práctica 7. Conexión con LCD por medio de bus i2c

Introducción

En esta práctica conectaremos una pantalla LCD al núcleo y ocuparemos una conexión i2c para enviarle un mensaje y que se imprima.

Materiales

Protoboard

Núcleo

Jumpers

LCD

I2C

Software

STM32

Procedimiento

Para la conexión I2C, se debe ir al apartado de conectividad y activar la conexión I2C, y activar los pines en PB8 y PB9, también se necesitarán 4 funciones extras para poder hacer el manejo de la conexión con el LCD

uint8\_t data\_xx[2];

#define SLAVE\_ADDRESS\_LCD 0X4E

Las anteriores son declaraciones de variables necesarias para el funcionamiento de este programa

void lcd\_send\_cmd (char cmd)

{

char data\_u, data\_l;

uint8\_t data\_t[4];

data\_u = (cmd&0xf0);

data\_l = ((cmd<<4)&0xf0);

data\_t[0] = data\_u|0x0C; //en=1, rs=0

data\_t[1] = data\_u|0x08; //en=0, rs=0

data\_t[2] = data\_l|0x0C; //en=1, rs=0

data\_t[3] = data\_l|0x08; //en=0, rs=0

HAL\_I2C\_Master\_Transmit (&hi2c1, SLAVE\_ADDRESS\_LCD,(uint8\_t \*)data\_t, 4, 100);

}

void lcd\_send\_data (char data)

{

char data\_u, data\_l;

uint8\_t data\_t[4];

data\_u = (data&0xf0);

data\_l = ((data<<4)&0xf0);

data\_t[0] = data\_u|0x0D; //en=1, rs=0

data\_t[1] = data\_u|0x09; //en=0, rs=0

data\_t[2] = data\_l|0x0D; //en=1, rs=0

data\_t[3] = data\_l|0x09; //en=0, rs=0

HAL\_I2C\_Master\_Transmit (&hi2c1, SLAVE\_ADDRESS\_LCD, (uint8\_t \*) data\_t, 4, 100);

}

void lcd\_init (void)

{

lcd\_send\_cmd (0x02);

lcd\_send\_cmd (0x28);

lcd\_send\_cmd (0x0c);

lcd\_send\_cmd (0x80);

}

void lcd\_init2 (void)

{

uint8\_t i=0;

HAL\_Delay(100);

for (i=0; i<3; i++)

{

lcd\_send\_cmd(0x03);

HAL\_Delay(45);

}

lcd\_send\_cmd (0x02);

HAL\_Delay(100);

lcd\_send\_cmd (0x28);

HAL\_Delay(1);

lcd\_send\_cmd (0x0C);

HAL\_Delay(1);

lcd\_send\_cmd (0x80);

HAL\_Delay(1);

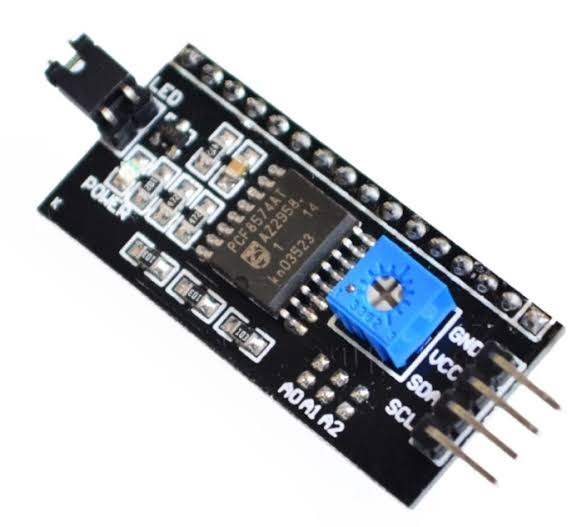
}

void lcd\_send\_string (char \*str)

{

while (\*str) lcd\_send\_data(\*str++);

}



Se recomienda que uno de los i2c se suelde al LCD.

Como se puede apreciar el I2C cuenta con SDA, SCL, VCC y GND

para conectar los i2c lo único que se debe hacer es, desde el núcleo con el uso de jumpers conectar los pines de 5v,GND,SDA(PB8),SCL(PB9).

Una nota importante es que el I2C cuenta con un regulador que es de color azul, como se puede apreciar en la imagen, con ella se puede modificar el contraste del LCD

Código

En el código lo único que se debe hacer es.

En main

lcd\_init();

lcd\_send\_string(“Hola Buenos días”);

## Práctica 8. Carro a control remoto por bluetooth con lcd

Introducción

El objetivo de esta práctica es poner en funcionamiento un carrito que cuenta con 2 motores de corriente directa, junto con el integrado L293D para administrar el movimiento de las ruedas, y una conexión por medio de bluetooth para controlar el movimiento. Hasta aquí esta práctica es similar a la número 2. Sin embargo, también tiene que contar con una pantalla LCD que esté conectada al Núcleo por medio de I2C, por lo tanto en esta práctica necesitaremos la pantalla y 2 I2C.

Materiales

Protoboard

Núcleo

Integrado l293d

Powerbank

Jumpers

Bluetooth HC06

Estructura de carro

LCD

2 I2C

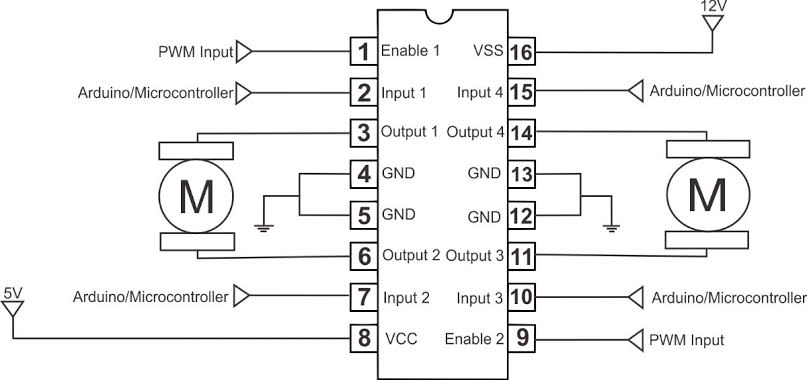
Software

STM32

Procedimiento

Comenzaremos por conectar el integrado a los motores, corriente y neutro.Es importante teneer en cuenta que la alimentación de los motores y el integrado debe ser completamente paralela al núcleo y HC06, por lo tanto se necesita la powerbank siguiendo el siguiente diagrama.

* Los pines 1, 8, 16, 9. Van conectados a corriente de 5v
* Los pines 2,7,10,15 Se conectan al I2C con dirección 0X4C
* Pines 4,5,12,13 se conectan a GND
* Pines 3,6,11,14 corresponden a los outputs y estos estarán conectados a los motores en pares
  + 3,6
  + 11,14

****

Una vez conectado el integrado con los motores podemos proseguir con la preparación al Núcleo. existen 6 pines que debemos modificar para proseguir con el proyecto

I2C

SLA

SDA

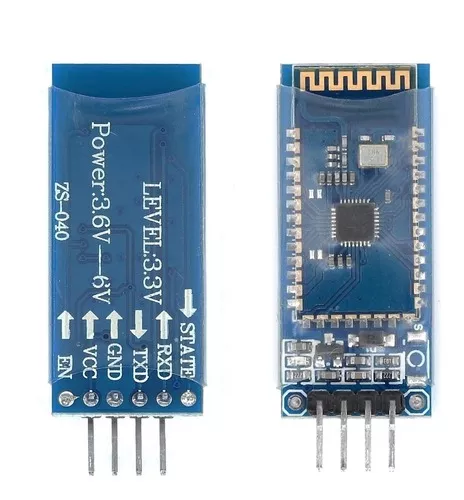
Usart

RX

TX

La necesidad de activar la conectividad por USART es para poder conectar el bluetooth, en el nuevo proyecto se debe ir a la pestaña de conectividad y activar USART2, seguido de seleccionar los pines en donde se requiere este tipo de protocolo, en este caso se seleccionó los pines PA9, PA10.

Nota importante, al conectar al dispositivo bluetooth, se deben de invertir RX-TX, TX-RX, y conectar a corriente y tierra normalmente. Estas conexiones pueden ir unidas al núcleo, así que se pueden aprovechar los pines de 5v y GND.



Para la conexión I2C al igual que con USART, se debe ir al apartado de conectividad y activar la conexión I2C, y activar los pines en PB8 y PB9, también se necesitarán 4 funciones extras para poder hacer el manejo de la conexión con el LCD

uint8\_t data\_xx[2];

char lectura[1];

#define SLAVE\_ADDRESS\_LCD 0X4E

#define CAR\_ADDRESS 0X4C

Las anteriores son declaraciones de variables necesarias para el funcionamiento de este programa

void lcd\_send\_cmd (char cmd)

{

char data\_u, data\_l;

uint8\_t data\_t[4];

data\_u = (cmd&0xf0);

data\_l = ((cmd<<4)&0xf0);

data\_t[0] = data\_u|0x0C; //en=1, rs=0

data\_t[1] = data\_u|0x08; //en=0, rs=0

data\_t[2] = data\_l|0x0C; //en=1, rs=0

data\_t[3] = data\_l|0x08; //en=0, rs=0

HAL\_I2C\_Master\_Transmit (&hi2c1, SLAVE\_ADDRESS\_LCD,(uint8\_t \*)data\_t, 4, 100);

}

void lcd\_send\_data (char data)

{

char data\_u, data\_l;

uint8\_t data\_t[4];

data\_u = (data&0xf0);

data\_l = ((data<<4)&0xf0);

data\_t[0] = data\_u|0x0D; //en=1, rs=0

data\_t[1] = data\_u|0x09; //en=0, rs=0

data\_t[2] = data\_l|0x0D; //en=1, rs=0

data\_t[3] = data\_l|0x09; //en=0, rs=0

HAL\_I2C\_Master\_Transmit (&hi2c1, SLAVE\_ADDRESS\_LCD, (uint8\_t \*) data\_t, 4, 100);

}

void lcd\_init (void)

{

lcd\_send\_cmd (0x02);

lcd\_send\_cmd (0x28);

lcd\_send\_cmd (0x0c);

lcd\_send\_cmd (0x80);

}

void lcd\_init2 (void)

{

uint8\_t i=0;

HAL\_Delay(100);

for (i=0; i<3; i++)

{

lcd\_send\_cmd(0x03);

HAL\_Delay(45);

}

lcd\_send\_cmd (0x02);

HAL\_Delay(100);

lcd\_send\_cmd (0x28);

HAL\_Delay(1);

lcd\_send\_cmd (0x0C);

HAL\_Delay(1);

lcd\_send\_cmd (0x80);

HAL\_Delay(1);

}

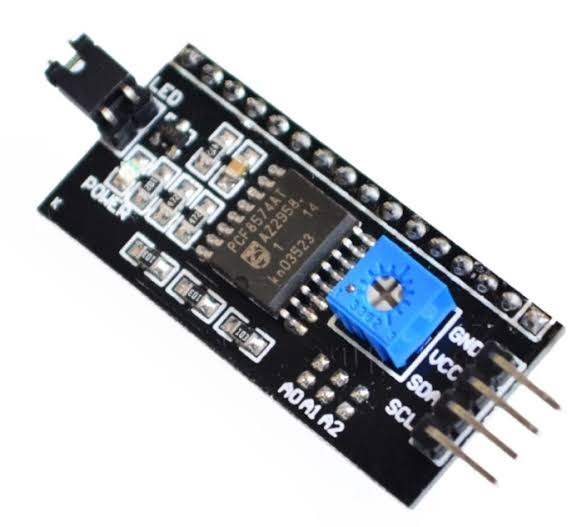
void lcd\_send\_string (char \*str)

{

while (\*str) lcd\_send\_data(\*str++);

}

Una vez declaradas las funciones es importante aclarar algunos puntos. Ya que se utilizaran 2 I2C, es necesario modificar la dirección de uno por lo tanto se deben de soldar los “pines” que corresponden al valor 0, esto provoca que la dirección de uno sea 0x4E (no soldado) y 0x4C (soldado), de esta manera en nuestro código es fácil enviar información a uno o al otro con el uso de estas dos direcciones



Se recomienda que uno de los i2c se suelde al LCD, de preferencia el que no haya sido modificado.

Como se puede apreciar el I2C cuenta con SDA, SCL, VCC y GND

para conectar los i2c lo único que se debe hacer es, desde el núcleo con el uso de jumpers conectar los pines de 5v,GND,SDA(PB8),SCL(PB9).

y conectar los dos I2C, ambos a los mismos pines.

Código

Para hacer la lectura de nuestra conexión bluetooth necesitamos declarar una función

char Leer\_Serial(){

char entrada[1];

HAL\_UART\_Receive(&huart1, entrada, sizeof(entrada),HAL\_MAX\_DELAY);

HAL\_Delay(10);

HAL\_UART\_Transmit(&huart2, entrada,sizeof(entrada),HAL\_MAX\_DELAY);

HAL\_Delay(10);

return entrada[0];

}

y dentro de main() se necesita insertar el siguiente código

lcd\_init2 ();

lcd\_send\_string ("Hola");

HAL\_Delay(5000);

lcd\_send\_cmd(0x01);

HAL\_I2C\_Master\_Transmit (&hi2c1, 0x4C, data\_xx, 2, 100);

while (1)

{

lectura[0]=Leer\_Serial();

lcd\_send\_cmd(0x01);

if(lectura[0]=='a'){ //izq

data\_xx[0]=0x01;

data\_xx[1]=0x01;

lcd\_send\_string ("\*\*\* Izquierda");

HAL\_I2C\_Master\_Transmit (&hi2c1, 0x4C, data\_xx, 2, 100);

}

if(lectura[0]=='b'){ //der

data\_xx[0]=0x10;

data\_xx[1]=0x10;

lcd\_send\_string ("\*\*\* Derecha");

HAL\_I2C\_Master\_Transmit (&hi2c1, 0x4C, data\_xx, 2, 100);

}

if(lectura[0]=='c'){ //adelante

data\_xx[0]=0x11;

data\_xx[1]=0x11;

lcd\_send\_string ("\*\*\* Adelante");

HAL\_I2C\_Master\_Transmit (&hi2c1, 0x4C, data\_xx,2, 100);

}

if(lectura[0]=='d'){ //atras

data\_xx[0]=0x22;

data\_xx[1]=0x22;

lcd\_send\_string ("\*\*\* Atras");

HAL\_I2C\_Master\_Transmit (&hi2c1, 0x4C,data\_xx, 2, 100);

}

HAL\_Delay(1000);

lcd\_send\_cmd(0x01);

HAL\_Delay(1000);

//data\_xx[0]=0xF0;

//HAL\_I2C\_Master\_Transmit (&hi2c1, 0x4C,data\_xx, 1, 100);

}

}

Conexiones finales

Se debe conectar el bluetooth, al núcleo teniendo en cuenta la observación anterior y conectar los pines del núcleo

* I2C pin 12 =2 (L293D)
* I2C pin 13 =7 (L293D)
* I2C pin 5 =15 (L293D)
* I2C pin 6=10 (L293D)

Al igual que los output del integrado l293d, alimentar la protoboard con la power bank e iniciar el programa en STM32