CONVERSOR ANALOGO-DIGITAL(ADC), DE UN MODULO JOYSTICK USANDO STM32F411.

Laboratorio #2

Juan Sandoval, Kevin niño

Juandsandoval8@gmail.com, Electrónica Industrial

Kevsalgado122@outlook.coml, Electrónica Industrial

Resumen—Esta guía explica cómo modificar un ejemplo de Mbed Compiler de un ADC, para poder visualizar los 12 bits de cada entrada análoga de un módulo joystick, además mostraremos su comportamiento por pantalla.

Abstract— ***This guide explains how to modify an example of the Mbed compiler of an ADC, to be able to visualize the 12 bits of each analog input of a joystick module, we will also show its behavior on the screen***

1. INTRODUCCIÓN

Un conversor o convertidor de señal analógica a digital es un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador y codificándose en muchos casos en un código binario en particular. Donde un código es la representación unívoca de los elementos, en este caso, cada valor numérico binario hace corresponder a un solo valor de tensión o corriente.

Objetivo. Aprender a usar el conversor ADC de nuestra tarjeta STMF411RE.

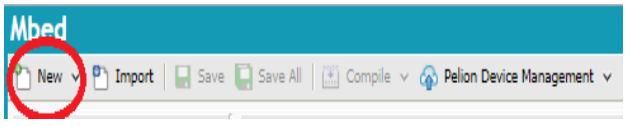
2. METODOLOGÍA

A. El problema

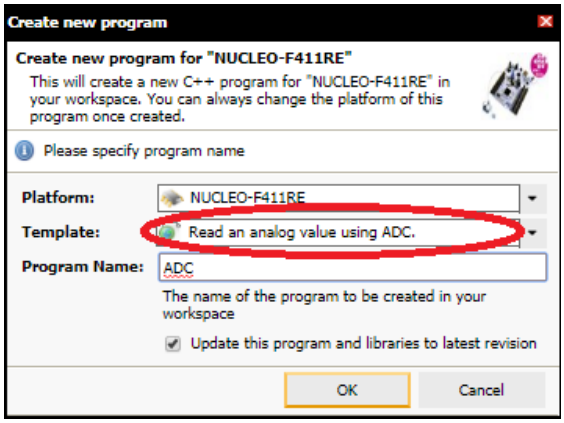
Se ha pedido un conversor ADC a nuestro modulo joystick y mostrar sus 12 bits de conversión en leds.

B. Mbed

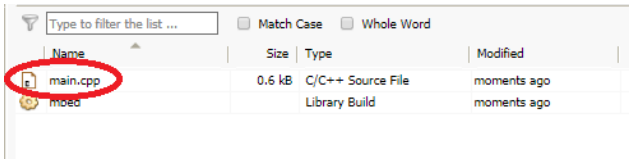
*Como mostramos en la guía anterior, después de crear nuestro usuario y cargar nuestra tarjeta haremos lo siguiente:



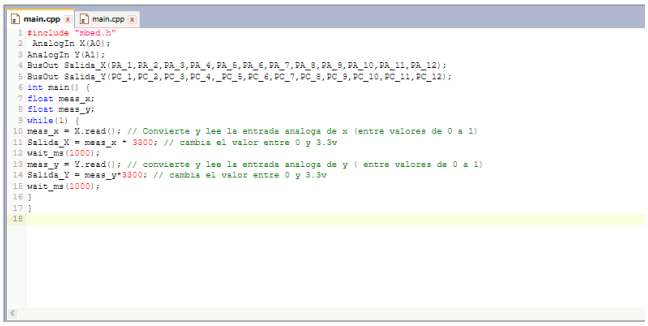
- Iremos a la pestaña que dice New para crear un proyecto nuevo que se ajuste a nuestras necesidades, una vez dado click sobre esta pestaña nos aparecerá una ventana emergente como esta:



- En template seleccionaremos Read an analog value using ADC (Leer un valor análogo usando ADC). Le daremos ok y seguiremos con el siguiente paso.



- Una vez hecho esto iremos a mian.cpp y empezaremos a cargar nuestro primer código.



- Después de borrar el código que por defecto nos arroja el compilador, se escribirá el siguiente código; que explicaremos más adelante línea alinea.

C. Instrucción AnalogIn

- Esta instrucción nos sirve para declarar una entrada análoga en este caso los dos potenciómetros de nuestro joystick. ejes (X, Y).

NOTA: Nuestro módulo joystick shield para Arduino encaja perfectamente en nuestra tarjeta STM32F411RE, Los pines (X, Y), encajan con nuestro A0 y A1, es por eso que se declaran A0=X y A1=Y.

D. Instrucción BusOut

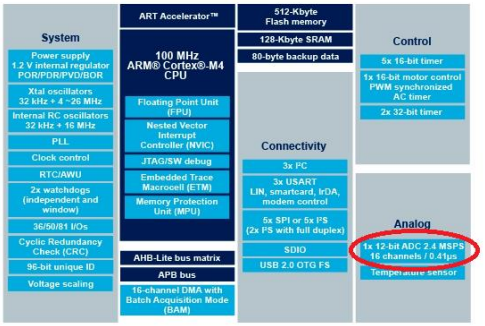
```
BusOut Salida_X(PA_1,PA_2,PA_3,PA_4,PA_5,PA_6,PA_7,PA_8,PA_9,PA_10,PA_11,PA_12);
BusOut Salida_Y(PC_1,PC_2,PC_3,PC_4,_PC_5,PC_6,PC_7,PC_8,PC_9,PC_10,PC_11,PC_12);
```

Esta instrucción nos permite ahorrarnos modificar nuestros GPIO por registros como salidas digitales, tan solo nombramos nuestro BusOut seguido del nombre que le daremos a nuestro Bus de salida en este caso como tenemos dos uno para X y otro para Y, los nombres de la siguiente manera:

BusOut Salida_x (PA_1…PA_12);

BusOut Salida_y (PC_1…PC_12);

NOTA: Se requieren 12 salidas ya que nuestro conversor análogo digital tiene 12 bits de salida y requerimos verlos en leds, estos 12 bits de salida se pueden ver en las especificaciones de nuestra tarjeta STM32F411RE en la página de ST.



-Aquí vemos que nuestra tarjeta tiene un ADC de 12bits de salida.

E. variables meas

```
int main() {
float meas_x;
float meas_y;
```

* estas variables de tipo flotante como lo son meas_x y meas_y, serán donde queden guardadas las mediciones de nuestros valores análogos convertidos digitales.

- Todo esto ya viene dentro del cuerpo del código, pero como solo son variables no van a venir dentro de nuestro código a reproducir.

E. Etapa de conversión de análogo a digital

```
while(1) {
meas_x = X.read(); // Convierte y lee la entrada analoga de x (entre valores de 0 a 1)
```

* La instrucción meas_x = X. read (); lo único que hace es recoger el valor análogo de X ó de A0 y lo lee entre valores de 0 y 1 que es una conversión completa ADC.

- En esta etapa ya tenemos nuestra conversión de un solo valor es decir que debemos repetir este paso para Y ó para A1.

- Estas instrucciones ya van dentro de lo que se va a repetir o lo que realmente queremos para solucionar el problema.

F. Definir y cambiar valores de salidas digitales

```
while(1) {
meas_x = X.read(); // Convierte y lee la entrada analoga de x (entre valores de 0 a 1)
Salida_X = meas_x * 3300; // cambia el valor entre 0 y 3.3v
wait_ms(1000);
```

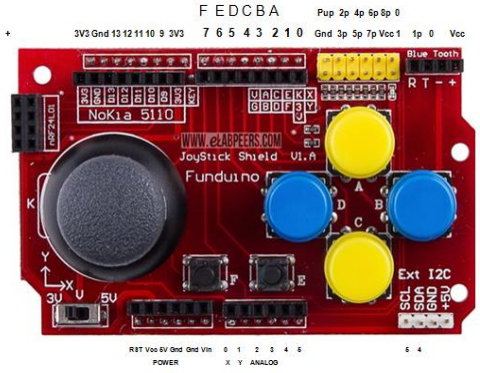
* Una vez nuestra lectura está hecha, lo siguiente será darle las salidas a nuestros 12 leds, que representarán nuestros 12 bits de salida de nuestro ADC, pero para esto tenemos que cambiar nuestra lectura de 1 a 3.3v ya que esa es la alimentación de nuestro módulo joystick; luego le daremos un tiempo de espera para que nos permita visualizar nuestro conversor completo de 12 bits y como ya habíamos dicho antes hay que repetir esta instrucción para Y.

```
while(1) {
meas_x = X.read(); // Convierte y lee la entrada analoga de x (entre valores de 0 a 1)
Salida_X = meas_x * 3300; // cambia el valor entre 0 y 3.3v
wait_ms(1000);
meas_y = Y.read(); // convierte y lee la entrada analoga de y ( entre valores de 0 a 1)
Salida_Y = meas_y*3300; // cambia el valor entre 0 y 3.3v
wait_ms(1000);
}
}
```

- Aquí se puede ver que la instrucción se repite para Y ya que tenemos dos posiciones en nuestro joystick.

G. El módulo joystick

* Para el laboratorio se usó el módulo joystick para Arduino, que como ya vimos posee dos posiciones el eje (X, Y), además posee alimentaciones a 3.3v y a 5v, salidas digitales conexiones a pantalla LCD y algunas salidas de receptor y emisor.



NOTA: si queremos que nuestro voltaje no varié máximo hasta 3.3v lo único que haremos es cambiar la instrucción, Salida y Salida_Y = meas_x y meas_y * 5000;

- Con esto ya tendremos un rango entre 0 y 5v.

Con esto ya tendremos nuestro respectivo código mostrando en leds los 12 bits de cada conversión para X como para Y, solo daremos compilar y usaremos nuestro archivo.bin y lo importaremos a nuestra tarjeta para cargar nuestro código.

H. Código para mostrar en pantalla

* Crearemos otro archivo fuente para crear nuestro código, hay que constar que este código solo seria de prueba para nuestro módulo joystick, con esto nos daremos cuenta si nuestro conversor si está funcionando antes de hacer cualquier montaje sobre protoboard.

- Para este código requerimos tres cosas nuevas muy importantes:

Tx: transmisor de nuestra tarjeta, la cual usaremos para mostrar en la pantalla de nuestro pc. (transmisión vía USB)

Rx: Nuestro receptor el cual también viene de nuestra tarjeta y la usaremos para recibir datos de nuestro mando joystick. (trasmisión vía USB)

RealTerm: Es un programa para poder visualizar en pantalla todo nuestro programa. (Tenemos que bajarlo desde nuestro navegador).

* Código fuente completo:

```
#include "mbed.h"

Serial pc (USBTX, USBRX); // tx, rx

AnalogIn analog_value_x(A0);

AnalogIn analog_value_y(A1);

int main () {
    float meas_x, meas_y;
    while (1) {
    pc.printf("X Value = %f\n\r", analog_value_x.read() * 3.3);
    pc.printf("Y Value = %f\n\r", analog_value_y.read() * 3.3);
    wait_ms (1000);


    }
}
```

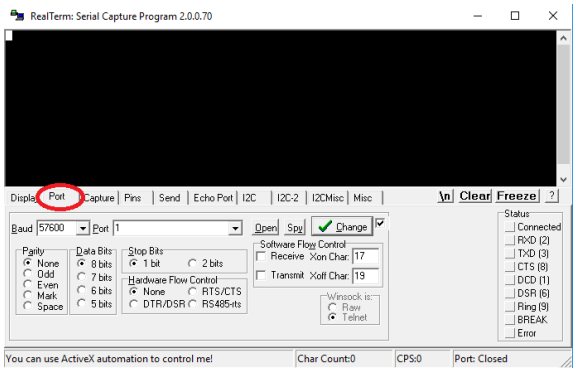
- Tanto Tx como Rx son comunicaciones seriales por eso la instrucción Serial pc, como cable USB usaremos el que tiene nuestra tarjeta STM32F411RE.

* Para el uso de nuestro programa RealTerm antes de compilar nuestro programa abriremos nuestro programa y haremos los siguiente:

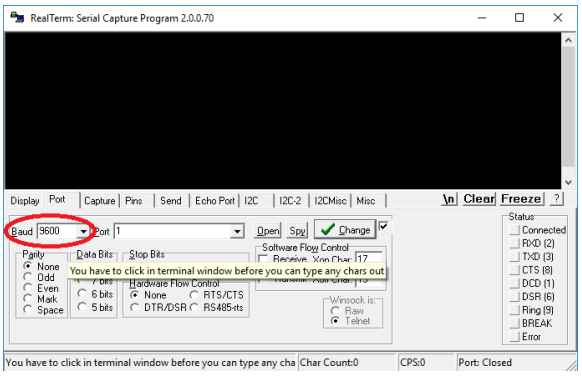
1. Compilaremos nuestro programa e importaremos nuestro archivo.bin a nuestra tarjeta STMF411RE.

2. Iremos a nuestro programa a configurar la conexión.

2.1. Lo primero es ir a el apartado PORT o entrada.

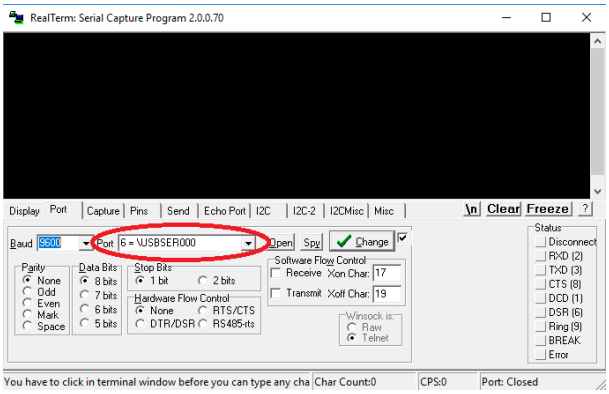


2.2. Iremos a configurar nuestros baudios de manera correcta a 9600 Baud.

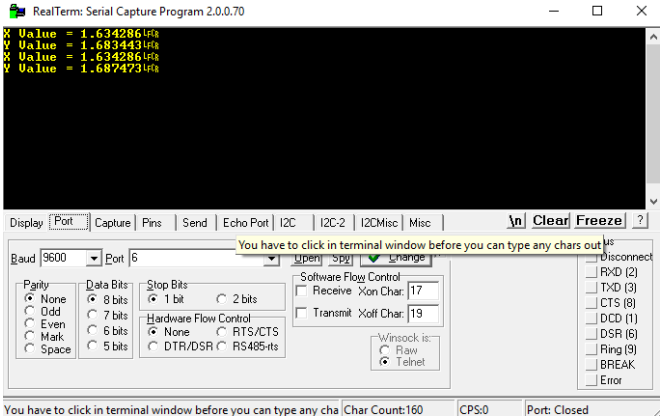


- Esto nos permitirá una conexión estable entre la tarjeta STM y nuestro pc.

2.3. Ahora buscaremos el apartado PORT y escogeremos nuestra conexión USB.

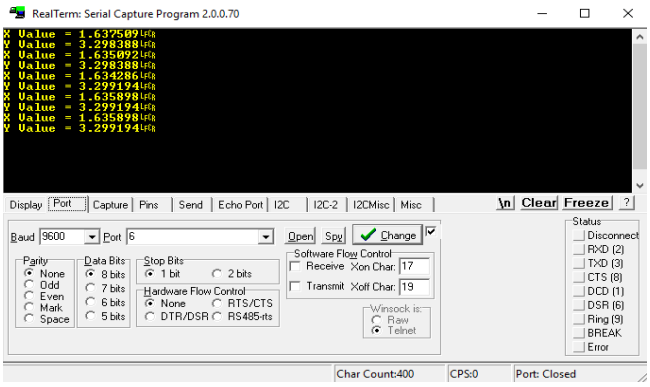


2.4. Le daremos click a OPEN para que nos abra todo el puerto de comunicación y nos empiece a mostrar los valores estándar de nuestro joystick.



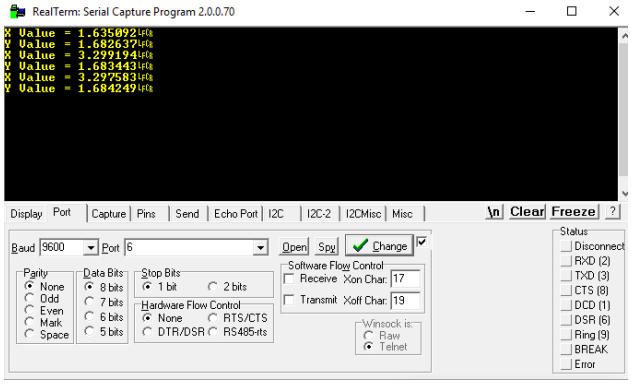
- Como podemos ver en pantalla, nos empieza a mostrar valores de 1.5v a 1.7v y esto se debe a que el potenciómetro no es perfecto.

2.5. Si subimos nuestro joystick hacia Y+ veremos valores de 3.3 o lo mas cercanos 3.3v

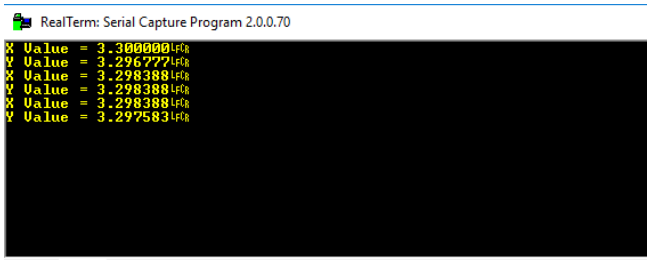


- Vemos como cuando movemos nuestro joystick hacia Y+ nos arroja valores para Y= 3.29v

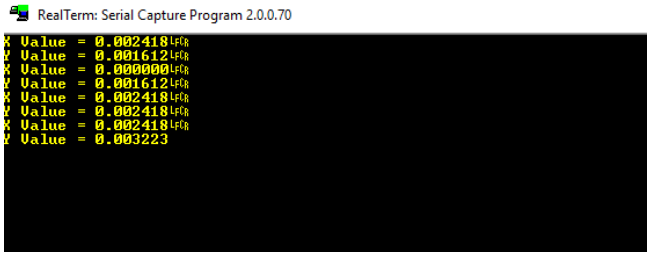
Lo mismo nos pasará si movemos hacia X+ nos arrojará valores de X= 3.29v



NOTA: Como solo podemos movernos en dos ejes nos mostrara uno en alto y el otro en estado de reposo, pero si tiramos el joystick en diagonal digamos el eje Z+ nos mostrara valores altos para X y para Y.



2.6. Si ponemos valores para X-, Y- o Z- nos arrojara valores de 0v ya que nuestro rango de valores va desde 0v hasta 3.3v.



I. conexión de pines y mapeo de los puertos GPIOA y GPIOC

* Las conexiones deben venir en orden para PA:

PA1-PA2-PA3-PA4-PA5-PA6-PA7-PA8-PA9-PA10-PA11-PA12.

- Recordemos que los PA son los 12 bits de las salidas del ADC de X o A0.

* Las conexiones para PC deben venir en orden también:

PC1-PC2-PC3-PC4-PC5-PC6-PC7-PC8-PC9-PC10-PC11-PC12.

- Recordemos que los PC son los 12 bits de las salidas del ADC de Y o A1.

Siendo esta la configuración correcta sobre la protoboard para un funcionamiento a la perfección de nuestro montaje.

NOTA: En algunos momentos se verán oscilar los leds sobre todo los de menos peso, pero esto se debe a que el potenciómetro no es perfecto y jamás nos dará un valor de 0 en estado de reposo, esto se puede observar cuando rectificamos por pantalla y al no mover nuestro joystick nos dan valores de 0v a 1.6v.

3. RESULTADOS Y DISCUSIÓN

El resultado de este laboratorio fue satisfactorio ya que se cumplió con lo estipulado en clase y se le dio una solución concreta y rápida.

Con el plus de haber rectificado el modulo joystick por pantalla para verificar si realmente está funcionando de manera correcta y ayudar a rectificar algunas dudas sobre el laboratorio.

4. CONCLUSIONES

Podemos concluir que los ejemplos que nos provee nuestro compilador de Mbed son bastante útiles, aunque debería venir una especie de ayuda para declarar algunas instrucciones, ya que Mbed es relativamente nuevo, aun así, es bastante útil estas ayudas ya que algún ejemplo solo basta con cambiar algunos valores.

5. REFERENCIAS

[1] *Docs.zephyrproject.org*, 2019. [Online]. Available: https://docs.zephyrproject.org/latest/_images/nucleo_f411re_morpho.png. [Accessed: 10- Oct- 2019].

[2] *Docs.zephyrproject.org*, 2019. [Online]. Available: https://docs.zephyrproject.org/latest/_images/nucleo_f411re_morpho.png. [Accessed: 10- Oct- 2019].

[3] *Docs.zephyrproject.org*, 2019. [Online]. Available: https://docs.zephyrproject.org/latest/_images/nucleo_f411re_morpho.png. [Accessed: 10- Oct- 2019].