USAR PWM PARA MOVER DOS SERVOS CON MODULO JOYSTICK USANDO TRAJETA STM32F411RE.

Laboratorio #3

Juan Sandoval, Kevin niño

Juandsandoval8@gmail.com, Electrónica Industrial
Kevsalgado122@outlook.coml, Electrónica Industrial

Resumen—Esta guía explica cómo mover dos servomotores, utilizando las salidas Pwm de la tarjeta STM32F411RE y generándolas con un módulo de joystick.

.

Abstract— ***This guide explains how to move two servomotors, using the Pwm outputs of the STM32F411RE card and generating them with a joystick module.***

1. INTRODUCCIÓN

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

Objetivo. Aprender a usar las salidas Pwm de nuestra tarjeta STMF411RE.

2. METODOLOGÍA

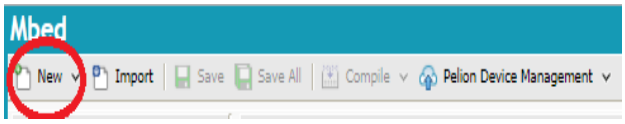
A. El problema

En los anteriores laboratorios empleamos la conversión ADC y el mapeo de los GPIO para saber cuáles pines usar.

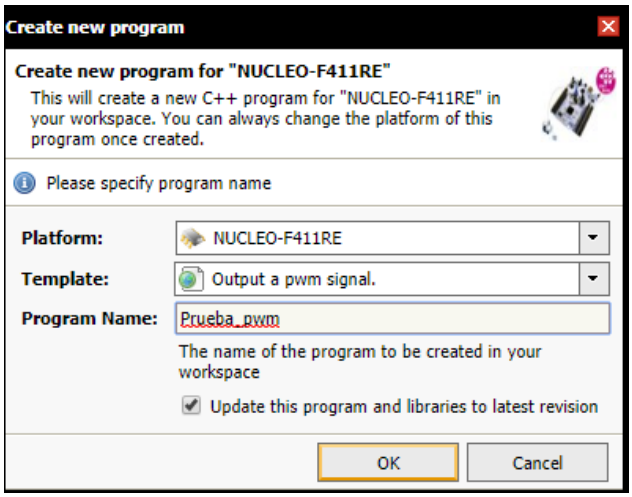
- Lo que se busca es mover un par de servos; uno para el eje X y el otro para el eje Y, generando los pulsos con un módulo joystick.

B. Mbed

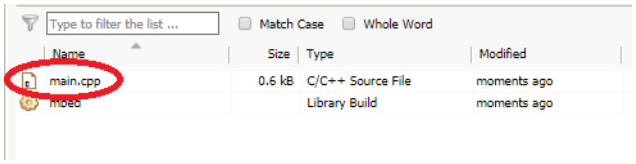
*Como mostramos en la guía anterior, después de crear nuestro usuario y cargar nuestra tarjeta haremos lo siguiente:



- 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 Output a Pwm signal (Señal Pwm como una salida). Le daremos ok y seguiremos con el siguiente paso.

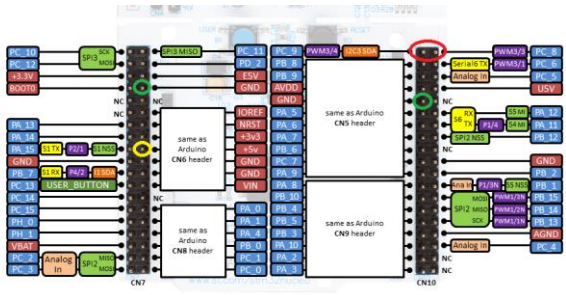


- Una vez hecho esto iremos a mian.cpp y empezaremos a cargar nuestro primer código.

```cpp
#include "mbed.h"

PwmOut mypwm(PWM_OUT);

DigitalOut myled(LED1);

int main() {

    mypwm.period_ms(10);
    mypwm.pulsewidth_ms(1);

    printf("pwm set to %.2f %%\n", mypwm.read() * 100);

    while(1) {
        myled = !myled;
        wait(1);
    }
}
```

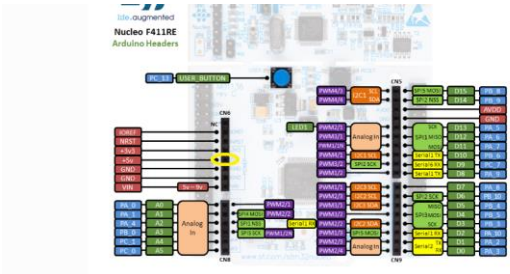
- Después de borrar el código que por defecto nos arroja el compilador, se escribirá el siguiente código; que explicaremos más adelante línea alinea.

C. Mapeo de los pines de salida Pwm



- En Amarillo son las salidas de 5v
- En verde las salidas de tierras
- En rojo las salidas Pwm que definimos para nuestros dos servos (PC9) y (PC8).

NOTA: La otra salida de 5v para nuestro otro servomotor la sacaremos de las entradas de Arduino de la placa STM.



- Se saca justo encima de donde salen los pines de nuestro modulo Joystick ya que son los mismos.

D. Instrucción Serial pc

```cpp
#include "mbed.h"

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

- Esta instrucción nos permitirá ver por pantalla lo que sucede con nuestro joystick será muy útil para determinar los valores a usar.

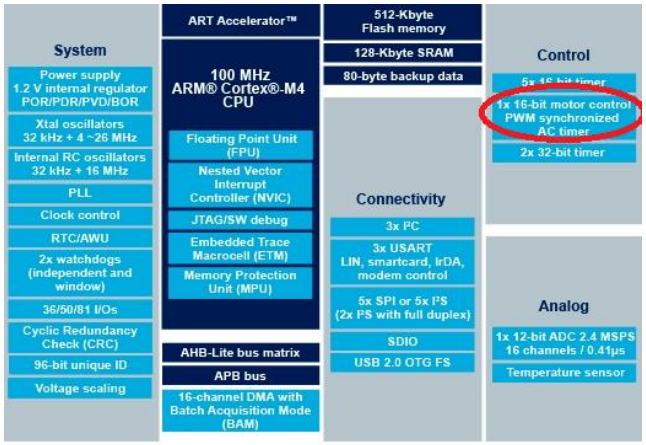
Nota: usaremos el cable de conexión de nuestra STM a nuestro PC para poder visualizar en nuestra consola RealTerm; programa que explicamos en el informe anterior.

E. Instrucción AnalogIn y PwmOut

```cpp
AnalogIn X(A0);
AnalogIn Y (A1);
PwmOut Servo_1 (PC_9);
PwmOut Servo_2 (PC_8);
```

- AnalogIn: Esta instrucción nos sirve para declarar una entrada análoga en este caso los dos potenciómetros de nuestro joystick. ejes (X, Y).
- PwmOut: Esta instrucción nos permite configurar una salida Pwm Desde los Gpio que tienen esa función predefinida en el mapeo.

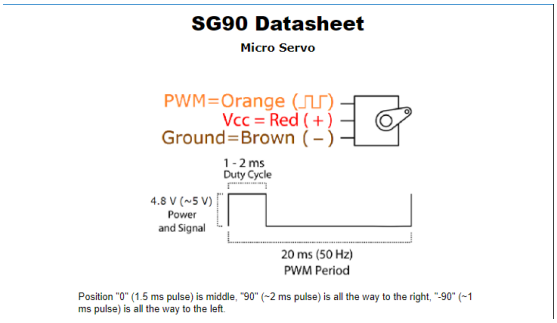
NOTA: Nuestro módulo joystick shield para Arduino encaja perfectamente en nuestra tarjeta STM32F411RE, Los pines (X, Y), encajan con nuestro A0 y A1, es por eso que se declaran A0=X y A1=Y.



-Aquí vemos que nuestra tarjeta tiene un PWM de 16 bits de salida

```cpp
int main ()
{
    Servo_1.period_ms(20);
    Servo_1.pulsewidth_ms(1);
    Servo_2.period_ms(20);
    Servo_2.pulsewidth_ms(1);

    float Temp_1;
    float Temp_2;
    int grados=750;
    int grados2=750;
```

* Servo_1 y Servo_2.period_ms: Como ya vimos en la instrucción anterior Servo_1 y Servo_2 son las variables donde guardamos las salidas de Pwm, ahora lo que se define es el periodo de nuestros servos, como se puede ver en el datasheet de estos.



- Se puede ver que estos servos tienen un periodo de 20ms, por eso definimos el periodo en las variables como el datasheet nos muestra y esto para ambos servos.

* Servo_1 y Servo_2.pulsewidht_ms: Esta variable nos indicara un valor inicial del servo para moverse y como vemos en nuestro datasheet nos indica que esta entre 1 y 2 ms.

* Float Temp_1,Temp_2: Es la variable donde vamos a guardar nuestra conversión ADC de cada EJE (A0) y (A0).

* Int grados, grados2=750: es valor inicial de nuestros servos al momento de cargar el código y esta exactamente en 90 grados.

Nota: Este valor después se explicará en el código.

F. Procesos matemáticos para usar el Pwm y conversión ADC

```
while(1) {
    Temp_1=X.read();
    pc.printf("El valor de la conversion es %.3f \n\r",Temp_1);
    Temp_1=(Temp_1*1000)-500;
    grados+=(int)(Temp_1*1);
    if (grados>1500) {
       grados=1500;
    }
    if (grados<0) {
       grados=0;
    }
    Servo_1.pulsewidth_us(1000+grados);

     Temp_2=Y.read();
    pc.printf("El valor de la conversion es %.3f \n\r",Temp_2);
    Temp_2=(Temp_2*1000)-500;
    grados2+=(int)(Temp_2*1);
    if (grados2>1500) {
       grados2=1500;
    }
    if (grados2<0) {
       grados2=0;
    }
    Servo_2.pulsewidth_us(1000+grados2);

 }
}
```

* La instrucción Temp_1= X.read(): En esta línea de código se realiza la conversión ADC de X (A0), estos valores de conversión están entre 0 y 1; esta misma lectura y conversión es para Temp_2= Y.read(); solo que para esta Y (A1).

* Pc.printf("El valor de la conversión es %.3f\n\r";temp_1): Esta instrucción nos permite ver la conversión en tiempo real en nuestra consola RealTerm, que ya aprendimos a usar en el laboratorio 2; lo mismo usaremos para Temp_2, como se muestra a continuación: Pc.printf("El valor de la conversión es %.3f\n\r";temp_2).

* Temp_1 =(Temp_2*1000)-500: En esta instrucción se logra mover el servo conforme al movimiento de joystick; explicaremos un paso a paso a continuación:

- Como ya sabemos la conversión de X y Y están guardas en sus respectivas variables Temp_1 y Temp_2, y esto tiene unos valores definidos entre 0 y 1, se hacen algunos cálculos:

Para tener en cuenta:

1. El valor 0 de nuestro joystick tiene un valor en conversión de 0,5 ya que se encuentra entre las dos mitades de la conversión ADC 1 y 0.

2. Los valores por los cuales multiplicamos nuestras conversiones están en uS.

Cálculos:

* Temp_1 = 0.5 (Valor 0 de nuestro joystick o punto medio)

Temp_1= (0.5*1000) = 500; Temp_1 (0.5*1000)-500 = 0

- Con esto logramos que nuestro servo y nuestro joystick se queden en posición inicial, y así para cualquier valor entre 0 y 1 el servo nos moverá la cantidad sugerida. Por ejemplo:

Tem_1 = 1 (valor de nuestro joystick hacia la derecha)

Temp_1 = (1*1000) = 1000; Temp_1 (0.5*1000)-500 = 500;

Esto nos moverá el servo, pero hasta el momento la relación no es correcta pues para un valor de 1 de nuestro joystick deberían haber 180 o 2500 ms.

* Ahora usaremos una variable llamada grados+ indicando que siempre son positivos:

- Grados+ = (Temp_1 * 1); pero hay un problema nuestros grados no pueden ser números flotantes, entonces usaremos una función de truncamiento de valores y ente caso truncaremos un valor flotante.

* Como se logra esto:

1. debemos volverlo un valor entero, lo que se hará es multiplicar una variable del tipo INT a nuestra operación entonces todo lo que tomemos en esta operación nos saldrá un numero entero.

-Grados+=(Int)(Temp_1*sensibilidad).

Pero esto aún no nos da la relación de movimiento de joystick con movimiento de servos.

- Colocamos entonces nuestras condiciones para regular el movimiento de los servos:

```
if (grados>1500) {
       grados=1500;
    }
    if (grados<0) {
       grados=0;
    }
```

* Para solucionar el problema de relación le daremos una última instrucción:

1. Servo_1 = Que es nuestra salida Pwm

2. pulsewidth = ciclo en alto

- Servo_1. pulsewidth_us(1000+grados);

Ahora esta última línea de código si nos soluciona el problema de relación. Por ejemplo:

- Sí Temp_1 = 500
- Grados+= 150
- Servo_1. pulsewidht_us (1000+500)= 1500
- Tenemos que 1500-> 180; esto fue definido en nuestra condición del If de un valor máximo de 1500.

Ahora si tenemos la relación correcta; replicaremos estos pasos para Y(A1). Y tendremos listo nuestros dos servos moviéndose independientemente de lo que generemos en X o en Y.

G. CONCLUSIONES

Podemos concluir que con la ayuda de los anteriores laboratorios fue un poco más fácil el desarrollo de este laboratorio, el problema radico mucho en la solución de la resolución matemática.

H. RESULTADOS Y DISCUSIÓN

El resultado fue satisfactorio ya que pudimos dar por concluido este tema y solucionado el problema estipulado en clase con la ayuda de los anteriores temas vistos como lo fueron Mapeo de pines y Conversión ADC de los dos potenciómetros del joystick.

I. 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].