

Comunicación Serial Matlab-STM

Juan Sandoval, Kevin Niño

Juandsandoval8@gmail.com, Ingeniería Electrónica
kevsalgado122@outlook.com, Ingeniería Electrónica

1

Resumen—La comunicación serie o comunicación secuencial, en telecomunicaciones e informática, es el proceso de envío de datos de un bit a la vez, de forma secuencial, sobre un canal de comunicación o un bus.

La ventaja de la comunicación serie es que necesita un número más pequeño de líneas de transmisión que una comunicación paralela que transmita la misma información. Por otra parte, surgen una serie de problemas en la transmisión de un gran número de bits en paralelo, como los problemas de interferencia o desincronización.

A la misma frecuencia de transmisión, la comunicación paralela tiene un mayor rendimiento.

Abstract—Serial communication or sequential communication, in telecommunications and information technology, is the process of sending data one bit at a time, sequentially, over a communication channel or a bus.

The advantage of serial communication is that you need a smaller number of transmission lines than a parallel communication that transmits the same information. On the other hand, a series of problems arise in the transmission of a large number of bits in parallel, such as interference or desynchronization problems.

At the same transmission frequency, parallel communication has higher performance

1. INTRODUCCIÓN

En este informe se buscará establecer una conexión del tipo serial entre Matlab y Mbed(STM), donde se hará mover dos servomotores, que hará mover una base, que servirá como trípode para ajustar la posición de una cámara de video.

2. METODOLOGÍA

A. PROBLEMA

En clase se planteó una parte del proyecto de grado que consistiría en establecer una comunicación serial entre Matlab y Mbed, donde como anteriormente se hizo en nuestra tarjeta STM, mover dos servos, ahora lo que se buscará es que nuestra tarjeta tenga el menor trabajo posible y que los pulsos los genere nuestro Matlab, que la tarjeta solo sea el medio donde recibamos el dato y mover nuestros servos y que recojamos nuestro ADC de nuestro joystick y enviárselo a Matlab para poder así generar nuestros tiempos a enviar.

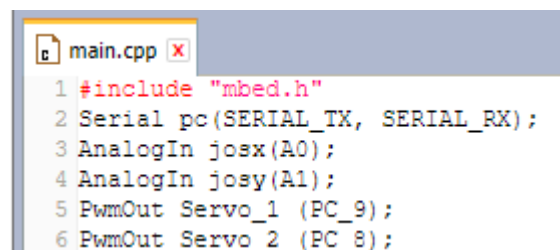
B. MATERIALES

- *STM32F411RE y Matlab
- *Modulo joystick
- *Servos

C. COMO USARLOS

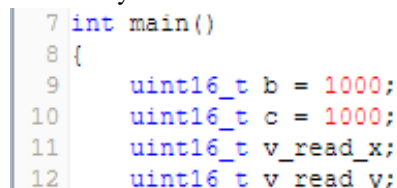
- STM32F411RE: Como ya vimos en los anteriores

informes y guías usaremos nuestro Mbed para programar nuestra tarjeta STM, una vez creado nuestro proyecto trabajaremos nuestro ADC pues será lo único que se enviará a Matlab.



```
1 #include "mbed.h"
2 Serial pc(SERIAL_TX, SERIAL_RX);
3 AnalogIn jox(A0);
4 AnalogIn josy(A1);
5 PwmOut Servo_1 (PC_9);
6 PwmOut Servo_2 (PC_8);
```

- Lo primero que se hace es declarar nuestros puertos seriales que vendrían siendo “SERIAL_TX Y SERIAL_RX”, definimos las entradas analógicas que corresponden a las de los potenciómetros del módulo joystick y por último se definen las salidas PWM que serán las que estén establecidas en la tarjeta para nuestro caso PC8 y PC9.



```
7 int main()
8 {
9     uint16_t b = 1000;
10    uint16_t c = 1000;
11    uint16_t v_read_x;
12    uint16_t v_read_y;
```

- En nuestro código principal se declarará variables del tipo uint16_t que lo único que nos indicara es que serán de 16 bits de tipo entero; para este caso las variables “v_read_x y v_read_y” serán las variables donde se almacenaran las correspondientes conversiones análogas

digitales de cada potenciómetro de nuestro modulo joystick o movimientos en x o en y.

- Por otro lado, las variables “b y c” serán las variables donde se almacenarán lo que se recoge de Matlab y luego se les dará el valor de PC8 y PC9 respectivamente como se verá más adelante.

```
13 while(1) {
14     v_read_x = josx.read_u16();
15     v_read_y = josy.read_u16();
16
17     pc.printf("%d %d\n",v_read_x,v_read_y);
18     wait_ms(20);
19     pc.scanf("%d %d\n",b,c);
20     wait_ms(20);
21     b= Servo_1;
22     c= Servo_2;
23 }
```

- En nuestro bucle pondremos las funciones a realizar por nuestra tarjeta y estas son la conversión analógica digital (ADC), lo único diferente a los anteriores ADC hechos anteriormente es que el valor de la conversión nos dará en una variable entera de 16 bits, una vez hecho esto lo siguiente será enviar los datos de nuestras conversiones a Matlab mediante un printf, donde se enviarán variables del tipo %d que son enteros que corresponden a las variables declaradas anteriormente “v_read_x y v_read_y”, una vez hecho esto procederemos a programar en nuestro Matlab para recibir el dato de nuestra conversión y así mismo enviar un tiempo que será recibido por nuestra tarjeta STM para mover los servos.

```
1 clear all;
2 close all;
3 clc;
4 delete(instrfind({'Port'},{'COM1'}));
5
6 s = serial('COM1','BaudRate',9600,'Terminator','LF');
7 %warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
8
9 fopen(s);
10
11 tmax = 10; % tiempo de captura en s
12 rate = 33; % resultado experimental (comprobar)
13
14 env1 = 0;
15 env2 = 0;
16 t = 0;
17
18 while t<tmax
19
20
```

- En nuestro Matlab lo único raro será abrir nuestro puerto serial que llamaremos “s”, dentro de este código vendrá el COM correspondiente, la velocidad de transmisión y que debe ser la misma para MBED, un terminator y un LF que será nuestro salto de línea. Lo siguiente será un tiempo de muestreo para visualizar los datos.

```
20
21 a = fscanf(s,'%d %d');
22
23 b = a(1)/43.69;
24 b = (b+1000);
25 b = round(b);
26
27 c = a(2)/43.69;
28 c = (c+1000);
29 c = round(c);
30
31 b
32 c
33 % env1 = b / 9.803921569;
34 % env2 = c / 9.803921569;
35 %
36 fprintf(s,"%d %d\n",b);
37 pause(0.1);
38 fprintf(s,"%d",env2);
39 %pause(0.01);
40
41 end
42 clc;
43 fclose(s);
44 delete(s);
45 clear s;
```

- En esta parte del código recibiremos el dato mediante un fscanf y recibiremos todo lo que viene del serial es decir nuestra variable “s” y serán variables del tipo %d (enteros).

- Una vez recibido el dato trabajaremos sobre estos valores y se empezaran a darle valores de tiempos para enviar a nuestra tarjeta STM, lo primero será declarar una variable b que será el valor de lo que recibimos en el fscanf y lo dividiremos entre 43.69 para que nos dé un valor de 1500 que dado en microsegundos(Us) será la mitad de nuestro servo 90°, esto nos servirá para que nuestro servo inicie en la mitad y así poder establecer cualquier valor con la siguiente línea de comando que sumará lo que recibamos de nuestra conversión + 1000; esto podría ser:

* Valor máximo de conversión que podemos recibir en Matlab es 65535

Eso será b; a este valor le dividiremos 43.69 y nos dará 1500 -> $65535/43.69=1500=b$

Ahora a b le sumaremos 1000 y con esto nos dará un valor máximo en tiempo (Us) de 2500. Y así para cualquier valor por ejemplo si la conversión que recibe Matlab fuera 12344, entonces sería $12344/43.69=282.53=b$ y a esto le sumamos 1000 nos dará como resultado 1282.53.

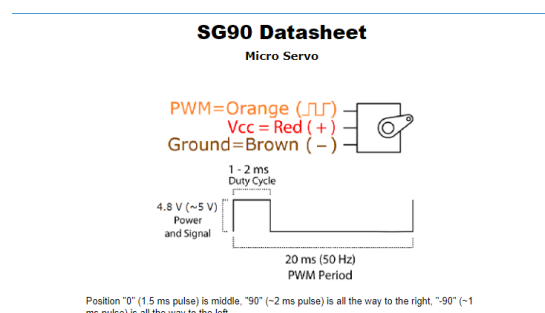
- Pero como nuestro servo solo recibe valores enteros lo que hacemos será redondearlo en Matlab con la función round

- Se hace este proceso para las dos conversiones y las enviaremos mediante un fprintf por medio de comunicación serial nuestra tarjeta STM.

```
pc.scanf("%d %d\n",b,c);
wait_ms(20);
b= Servo_1;
c= Servo_2;
}
```

- Una vez enviado el dato desde Matlab, lo que haremos en Mbed es que nos reciba el dato mediante un scanf que será lo que recibiremos del serial, es decir el tiempo en alto que hará funcionar nuestros servos, luego de recibirlas les daremos el valor de cada variable recibida a nuestras salidas PWM para mover nuestros servos y así concluir con nuestro trabajo.

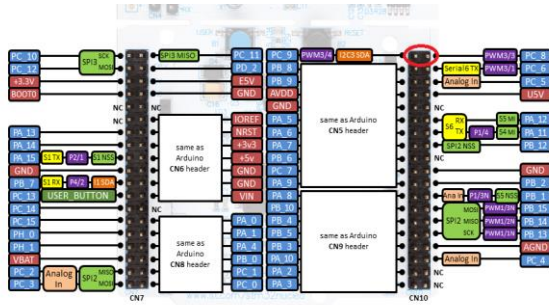
E. SERVOS Y CONEXIONES



- Bien podemos conectar nuestros servos a las alimentaciones de nuestra STM o alimentarlos vía externa con 5v.

- Pero el cable de entrada de PWM si serán conectados en nuestros pines que especifican salidas PWM en nuestra tarjeta en nuestro caso serán los siguientes pines.

Para la tarjeta STM32F411RE:



- La zona marcada con rojo son las salidas PWM (PC8 y PC9), que especificamos al principio del código, ahí irán conectados las entradas PWM de nuestros servos.

3. RESULTADOS Y DISCUSIÓN

Aunque al principio fue complicado y bastante tedioso el tratar de lograr el enlace entre Matlab y Mbed los resultados al final fueron los esperados nos tardamos más de lo que teníamos presupuestado pues a veces es complicado programar en Mbed y mucho más en Matlab.

4. CONCLUSIONES

Podemos concluir que muchas veces tenemos un déficit grande en cuanto a la programación con esta tarjeta de desarrollo aun así cuantas con muchas facilidades que esperamos tener la facilidad de usar después con la experiencia adquirida en esta clase, se logró el objetivo que es lo importante pero falta mejorar.

5. BIBLIOGRAFIA

- *Docs.zephyrproject.org. (2019). ST Nucleo F411RE — Zephyr Project Documentation. [online] Available at: https://docs.zephyrproject.org/latest/boards/arm/nucleo_f411re/doc/index.html [Accessed 29 Nov. 2019].*
- *Arm Mbed. (2019). Home | Mbed. [online] Available at: <https://www.mbed.com/en/> [Accessed 29 Nov. 2019].*
- *La.mathworks.com. (2019). MATLAB - El lenguaje del cálculo técnico. [online] Available at: <https://la.mathworks.com/products/matlab.html> [Accessed 29 Nov. 2019].*