*PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA.*

*FACULTADAD DE CIENCIAS DE LAS INGENIERIAS.*

*DEPARTAMENTO DE INGENIERIA ELECTRONICA Y ELECTROMECANICA*

*LABORATORIO DE MICROPROCESADORES I.*

## Entrega Proyecto Final

**Nombres:** Homar López Hawa Fernando Estrella

**Matrícula:** 2011-1339 2011-0732

**Calificación:**

## Objetivo:

- Desarrollo y uso del STM32F100RB;

## Procedimiento:

Se controlará un carro de juguete de manera inalámbrica mediante un “guía”. Se tendrá un modelo de timón o guía, el cual si se gira hacia la izquierda el carro doblará a la izquierda; si se gira el guía a la derecha el carro doblará a la derecha. Para controlar la velocidad dependerá de si se inclina hacia delante o hacia detrás, para acelerar o desacelerar respectivamente.

Diseñe un circuito utilizando el STM32F100RB que sea capaz de establecer la posición en la que se encuentra inclinado el modelo de guía. Para esto debe utilizar un acelerómetro analógico.

Luego de tener esta posición debe ser capaz de transmitir esta información de manera inalámbrica y ser recibida por un sistema que controle el avance, retroceso y giro de un carro.

## Enfoque:

Para la realización de este proyecto fue necesario dividirlo en 3 diferentes partes:

- Determinación de dirección;
- Conexión inalámbrica;
- Control de giro y avance del carro;

### 1. Determinación de dirección:

Para determinar la dirección a la que se desea mover se ha utilizado un acelerómetro. Su descripción de funcionamiento es la siguiente:

El Acelerómetro es un sencillo circuito integrado que mide la aceleración en 3 ejes. La aceleración es el cambio de velocidad respecto al tiempo que sufre un objeto. Esta se mide con unos circuitos internos muy pequeños que tienen unas placas metálicas que hacen una función capacitiva entre ellas. Todas las placas excepto una están inmóviles en el circuito. La que se mueve hace esto debido a la elasticidad del material que la conecta al circuito. Cuando sufre una aceleración, su posición cambia. El circuito transforma el cambio de posición de la placa móvil respecto a las inmóviles en una capacitancia, y a su vez convierte esta en un voltaje de salida. También se ha de considerar que no solo el movimiento produce aceleración. El campo gravitatorio entre los objetos ejerce una aceleración también. La más notable para nosotros es la del planeta Tierra sobre todo lo demás en el. Esto es debido a la atracción entre los objetos del universo. Por lo tanto, aun al mantenerse inmóvil, el acelerómetro sufrirá una aceleración de 1 g (medida de la aceleración de la gravedad de la Tierra). Como el acelerómetro hace mediciones para tres ejes ortogonales entre sí, calculando la componente de la aceleración que afecta cada eje se puede saber la orientación del acelerómetro en el espacio. Sabiendo esto, se puede configurar el circuito de manera que cuando el acelerómetro se encuentre en una orientación dada, el carrito vaya en cierta dirección.

El este caso el acelerómetro utilizado es el ADXL335. Las especificaciones eléctricas de este componente son las siguientes:

| Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| SENSOR INPUT | Each axis | | | | |
| Measurement Range | | ±3 | ±3.6 | | *g* |
| Nonlinearity | % of full scale | | ±0.3 | | % |
| SENSITIVITY (RATIOMETRIC)[2] | Each axis | | | | |
| Sensitivity at $X_{OUT}$, $Y_{OUT}$, $Z_{OUT}$ | $V_S$ = 3 V | 270 | 300 | 330 | mV/*g* |
| ZERO *g* BIAS LEVEL (RATIOMETRIC) | | | | | |
| 0 *g* Voltage at $X_{OUT}$, $Y_{OUT}$ | $V_S$ = 3 V | 1.35 | 1.5 | 1.65 | V |
| 0 *g* Voltage at $Z_{OUT}$ | $V_S$ = 3 V | 1.2 | 1.5 | 1.8 | V |
| POWER SUPPLY | | | | | |
| Operating Voltage Range | | 1.8 | | 3.6 | V |

*Nota: Esta tabla no contiene todos los parámetros, solo los que nos interesan.

## AXES OF ACCELERATION SENSITIVITY

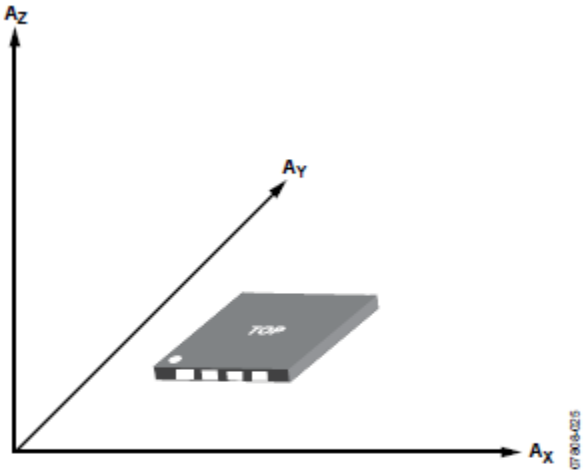



*Figure 23. Axes of Acceleration Sensitivity; Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis.*

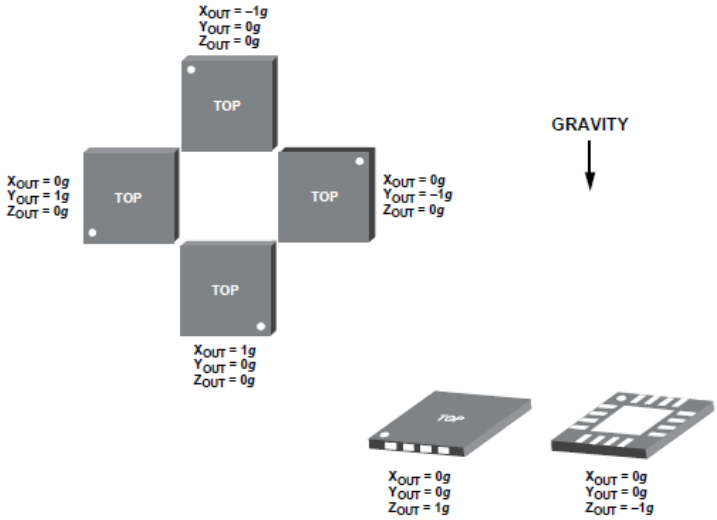



*Figure 24. Output Response vs. Orientation to Gravity*

Imagen que muestra los cambios de g con respecto a la orientación. Si el valor es de 1g, la salida para ese canal será 1.5+0.3 ≈ 1.8. Si el valor es -1g, entonces la salida será 1.5 – 0.3 ≈ 1.2.


Luego de tener la manera de saber la posición en la que se encuentra el dispositivo se procede a procesar y controlar esta señal.

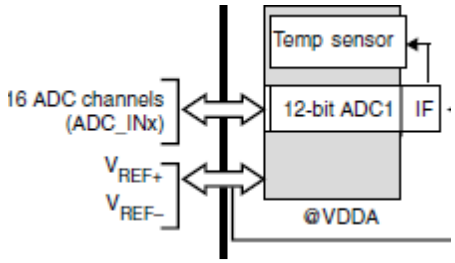
←Esta imagen es una imagen del componente físico utilizado.

## Micro controlador:

Un micro controlador (abreviado µC, UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un micro controlador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida. En este caso se está utilizando un STM32F100RB.

La función principal de este micro controlador para este diseño es de servir de "traductor" entre la señal análoga del acelerómetro y la lógica digital del puente H (necesario para controlar el carro).

Primeramente se necesita convertir la señal analógica del acelerómetro en una señal digital para poder ser procesada. Esto se logra utilizando el convertidor Análogo Digital que contiene el micro controlador en su interior:




Como se puede observar solo se tiene un convertidor, pero este tiene varios canales con los que puede trabajar. En nuestro caso se necesitan dos canales para convertir la posición en X y la posición en Y.

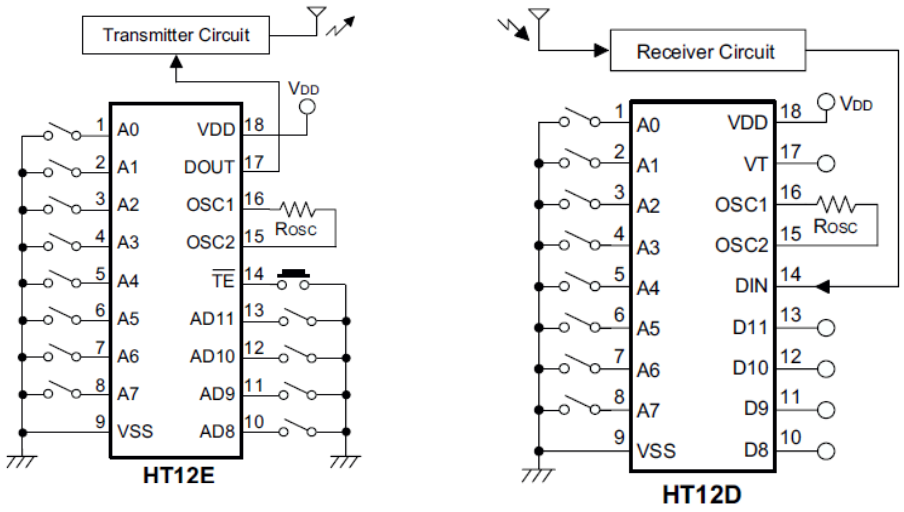
Luego de tener el dato digital equivalente a la posición se procede a transformar ese dato en el dato deseado (ver lógica puente H). Este también ha de generar un PWM para la variación de velocidad en el carro. Para ver como se realiza esto, revisar sección de CÓDIGO.

## 2. Conexión inalámbrica:

Para la conexión inalámbrica se utiliza un par de integrados, un codificador y un decodificador, los cuales pueden enviar 4 bits de información de manera paralela.

El HT12E (codificador, imagen de la izquierda) funciona así:

Las patas 1 a 8 eligen la dirección (que tendría que ser igual en el transmisor y receptor para lograr comunicación), las patas 10 a 13 son los datos que se desean enviar, la pata 14 controla el envío (al dejarla en estado bajo transmite, es posible dejarla en 0 para que transmita continuamente), la patas 15 y 16 necesita ser conectada a una resistencia de 1 MΩ para generar la señal de clock interna, la pata 17 es la salida de datos, que debe ser conectada al módulo de transmisión.




El HT12D (decodificador, imagen de la derecha) funciona así:

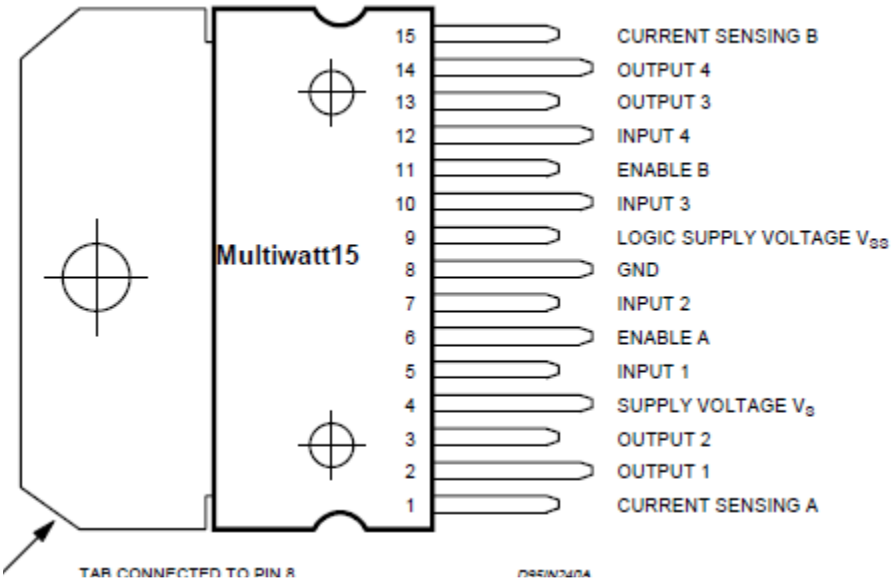
Nuevamente, las patas 1 a 8 eligen la dirección (que tendría que ser igual en el transmisor y receptor para lograr comunicación), las patas 10 a 13 son los datos recibidos (pueden ser conectados a LEDs o cargas siempre que la corriente no supere 5mA), la pata 14 deberá ser conectada a la salida del módulo de recepción de RF, las patas 15 y 16 necesitan ser conectadas a una resistencia de 47 kΩ para generar la señal de clock interna, la pata 17 dice si la recepción fue correcta o no.

En nuestro caso, los pines A1-A7 están al aire en ambos integrados. En el codificador los pines AD11-AD8 están conectados con el micro, el cual le facilita el dato de 4bits necesario para el puente H. Con relación al decodificador los pines D11-D8 están conectados a las entradas $I_1$, $I_2$, $I_3$ y $I_4$ del puente H.

## 3. Control de giro y Avance del Carro:

### Puente H:

Para control de giro y avance del carro se utilizó un L298. Este integrado contiene en su interior dos puente H capaces de suministrar hasta 4 A. El diagrama de este integrado es el siguiente




En el circuito se conectaron juntos los pines 15, 1 y 8. Esto se realizó ya que no era necesario hacer censado de corriente y el fabricante especifica que en caso de no utilizar los pines 15 y 1 es mejor llevarlos a tierra. Para el control de giro se han conectado los pines 2 y 3 de las terminales del motor delantero. El giro es controlado dependiendo de la señal digital que reciban las entradas 1 y 2, es decir, los pines 5 y 7. Para el control de avance o retroceso se conectó al motor trasero los pines 13 y 14, los cuales son controlados por las entradas de control Input 3 y 4, que corresponden a los pines 10 y 12 respectivamente.

Este circuito necesita algunas conexiones externas, como un capacitor de 100nF entre el Pin 4 y 8. Al igual que entre los pines 9 y 8. Más debajo se muestran los parámetros máximos de este integrado.

Este circuito se ha probado con tres baterías de 1.5v y ha funcionado perfectamente.

| Symbol | Parameter | Value | Unit |
|---|---|---|---|
| $V_S$ | Power Supply | 50 | V |
| $V_{SS}$ | Logic Supply Voltage | 7 | V |
| $V_I$,$V_{en}$ | Input and Enable Voltage | –0.3 to 7 | V |
| $I_O$ | Peak Output Current (each Channel)<br>– Non Repetitive (t = 100µs)<br>–Repetitive (80% on –20% off; $t_{on}$ = 10ms)<br>–DC Operation | <br>3<br>2.5<br>2 | <br>A<br>A<br>A |
| $V_{sens}$ | Sensing Voltage | –1 to 2.3 | V |
| $P_{tot}$ | Total Power Dissipation ($T_{case}$ = 75°C) | 25 | W |
| $T_{op}$ | Junction Operating Temperature | –25 to 130 | °C |
| $T_{stg}$, $T_j$ | Storage and Junction Temperature | –40 to 150 | °C |

**Lógica Puente H1**

| **$I_1$** | $I_2$ | Out 1 | Out 2 | Dirección |
|---|---|---|---|---|
| **0** | 0 | 0 | 0 | NADA |
| **0** | 1 | 0 | 1 | Adelante |
| **1** | 0 | 1 | 0 | Atrás |

**Lógica Puente H2**

| **$I_3$** | $I_4$ | Out 3 | Out 4 | Dirección |
|---|---|---|---|---|
| **0** | 0 | 0 | 0 | NADA |
| **0** | 1 | 0 | 1 | IZQUIERDA |
| **1** | 0 | 1 | 0 | DERECHA |

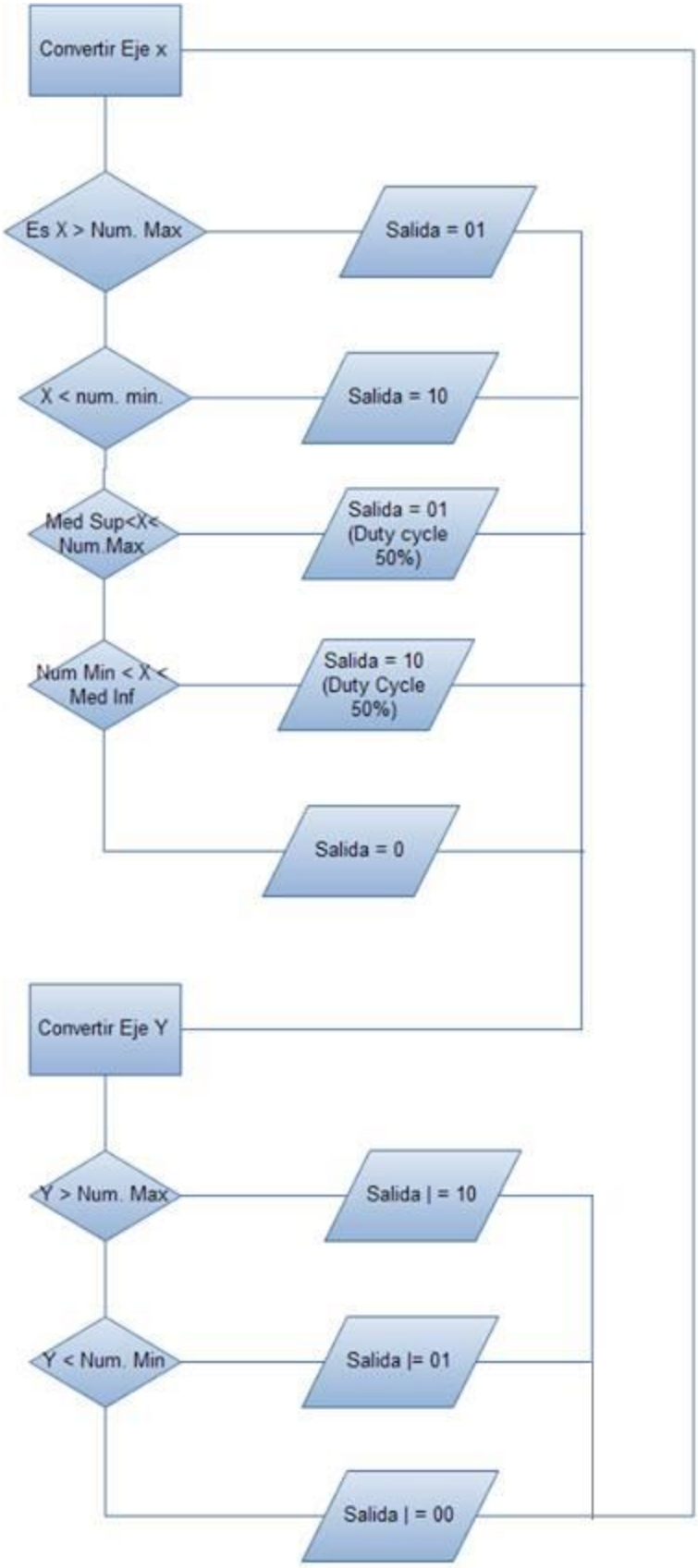
El puente H1 es para controlar el avance y el retroceso del carro. Para el cambio de velocidad en vez de enviar un 1, se envía un PWM con un duty cycle a 50%. El puente H2 se encarga de controlar el giro del carro.

Las salidas van conectadas a los motores del carro, cuyo funcionamiento se describe a continuación:

### Carro:

El carrito a control remoto es un carrito de juguete convencional, alimentado por tres pilas AA de 1.5 V cada una, al que se le retiro el circuito de control que traía originalmente. El principio de funcionamiento del mismo como vehículo consta de que dos motores DC, uno delante y otro detrás, controlan la desviación lateral y el desplazamiento frontal/trasero, respectivamente. Un motor DC es uno de dos terminales cuya velocidad de giro está determinada por el voltaje DC equivalente de la señal diferencial (o sea, entre dos terminales) de entrada. La polaridad de esta señal indica la dirección de giro del eje del motor. Aplicando un voltaje en una polaridad dada, el motor trasero hace avanzar el carrito, y aplicando la polaridad inversa, el carrito va en reversa (en este caso, una diferencia de potencial de Voltios). Así mismo, al aplicar un voltaje en una polaridad dada en el motor frontal, este gira las ruedas aproximadamente 40 grados hacia un lado dado y las deja así. Inversamente, al aplicar un voltaje en la otra polaridad, las ruedas frontales giran en la dirección opuesta.

## Diagrama de Flujo:



## Código:

```
#include <stm32f10x.h>//definición de los nombres

int n;
int out;
int led;

//THERE SHOULD BE A DELAY BETWEEN TIMER ON AND START OF CONVERSION
//resolution (12 bits) (Vref + aprox 3 Volts) (Vref- = 0 V)

int main(void)
{
	led = 0;
	n = 0;
	out = 0;


	RCC->APB2ENR = 0x214;//Reset and clock control: Del registro RCC se accede a la parte APB2ENR
	GPIOA->CRL = 0; //PUERTO A, que tiene el ADC, configurado como INPUT, ANALOGO
	GPIOC->CRL = 0x22222222;//PUERTO C LOW, config en general purpose output, max speed 2MHz
	GPIOC->CRH = 0x22222222;//PUERTO C HIGH, config en general purpose output, max speed 2MHz

	//Configuración del timer
    RCC->APB1ENR |= 0x1;		//Habilita el reloj del timer 2.
    //TIM2->PSC = 23999;			// divisor de la frecuencia de reloj del timer
    TIM2->PSC=100;			// divisor de la frecuencia de reloj del timer
    TIM2->ARR = 2;
	TIM2->CR1    |= 0x1;			//habilita el timer y lo arranca!.


	NVIC_EnableIRQ(ADC1_IRQn);

	ADC1->CR2 = 0x1; //ADC ON , single mode(cont=0),right alignment (bit 11)
	ADC1->CR1 = (1<<5); //habilita EOC, que nos permite saber que acabo la conversión
	ADC1->SMPR2 = 0x000; //sample time for each channel set to 1.5 cycles
	ADC1->SQR1 = 0;		// total number of conversions (one conversion is a 0 in the bits #20-23), other bits are order of the conversion of channel, from last to first going down
	ADC1->SQR2 = 0; // no channels converted in this range (from 12th to 7th)

	GPIOC->ODR = 0x0; //valor inicial
```

```
while(1) // repite todo esto
{

    ADC1->SQR3 = 0x1; // in the first 5 bits you put the channel # of the channel you want to be converted first, in this case channel 2 (second in position)
        ADC1->CR2 |= ADC_CR2_ADON; //comenzar la conversión


            while (n<30) // crea un retardo para que le dé tiempo a convertir
            {
            n    = n + 1;
            }
            n = 0;

                    if (ADC1->DR > 2100)     // límite superior de la conversión que controla el movimiento frontal/trasero
                    {
                            out = 0x102;
                    }
                    else if ((ADC1->DR < 2120) & (ADC1->DR > 2060)) //rango intermedio para el avance
                    {
                            if(TIM2->SR & TIM_SR_UIF )       //Verifica el flag de actualización del timer 2
                            {
                                    TIM2->SR &= ~(1<<0); // Resetea el flag UIF
                                    led ^= 0x102; //led on y para delante con velocidad intermedia
                                    out |= led;
                            }
                    }
                    else if ((ADC1->DR > 1950) & (ADC1->DR < 1990))
                    {
                            if(TIM2->SR & TIM_SR_UIF )       //Verifica el flag de actualización del timer 2
                            {
                                    TIM2->SR &= ~(1<<0); // Resetea el flag UIF
                                    led ^= 0x101; // led on y para atrás con velocidad intermedia
                                    out |= led;
                            }
                    }
                    else if (ADC1->DR < 1950) // para atrás con velocidad máxima
                    {
                            out = 0x101;
                    }
```

```
                        else
                        {
                                out = 0x0; // posición inmóvil
                        }


        ADC1->SQR3 = 0x2; // in the first 5 bits you put the channel # of the channel you want to be converted first, in this case channel 2 (second in position)
        ADC1->CR2 |= ADC_CR2_ADON; //start conversion del otro eje



                        while (n<30 ) // crea un retardo para que le dé tiempo convertir
                        {
                        n    = n + 1;
                        }
                              n = 0;



                        if (ADC1->DR > 2080) //se mueve a la derecha si se da esa condición
                        {
                                out |= 0x4;//hace una OR con todo el dato del eje anterior para que no afecte el dato del otro eje
                        }
                        else if (ADC1->DR < 1990) // se mueve a la izquierda si se da esta condición
                        {
                                out |= 0x8;
                        }
                        else
                        {
                                out |= 0x0; // no dobla
                        }

                    GPIOC->ODR = out;// pone el dato resultante en la salida del puerto C
      }
}

void ADC1_IRQHandler(void)
{
                ADC1->SR = 0; // si se da la interrupción, resetea el pin de estatus
}
```