Laboratorio 06 - CONTROL DE VELOCIDAD DE MOTOR DC CON AJUSTE CONSTANTE MEDIANTE COMUNICACIÓN USB vCOM

Martinez Benavides, Camilo
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
camilo.martinez01@usa.edu.co

Rodriguez Rios, Oscar
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
oscar.rodriguez02@usa.edu.co

Resumen

En este laboratorio se documenta el proceso de medición y control de las revoluciones por minuto (RPM's) de un motor DC mediante la energía suministrada por el módulo PWM que se encuentra a disposición de asignación por los datos emitidos por un Widget del programa Qt creator en el PC, además el modulo PWM tiene un ajuste de carácter proporcional constante determinado por el objetivo de revoluciones planteadas y emitidas desde el Widget, el cual también recibe la transmisión de datos del microcontrolador que contiene las RPM's medidas. Para realizar todo lo anterior primero se crea y programa el proyecto del STM32, incluyendo configuración general de características de funcionamiento de la tarjeta, configuración especifica de un TIM y dos de sus canales para la medición de RPM's y emisión del PWM, explicación de funciones e interrupciones indispensables para obtener información o realizar cálculos, explicación detallada de la lógica de funcionamiento del programa en el ciclo infinito y comunicación bidireccional mediante USB virtual COM (USB vCOM) teniendo en cuenta un protocolo de comunicaciones especifico.

Palabras cláve:

PWM, medición, transmisión, RPM's, Widget, Protocolo de comunicaciones, USB vCOM

1. Marco teórico

PWM: significa Pulse Width Modulation (Modulación por ancho de pulso). Para transmitir una señal, ya sea analógica o digital, se debe modular para que sea transmitida sin perder potencia o sufrir distorsión por interferencias. Esta modulación es muy usada para controlar la cantidad de energía que se envía a una carga, es una técnica utilizada para regular la velocidad de giro de los motores, regulación de intensidad luminosa, controles de elementos termoeléctricos o controlar fuentes conmutadas entre otros usos. [1]

QT creator: Qt Creator es un entorno de desarrollo integrado multiplataforma que puede usar para crear aplicaciones o modificar aplicaciones existentes. [2]

Protocolo de comunicaciones: Los protocolos de red son estándares y políticas formales, conformados por restricciones, procedimientos y formatos que definen el intercambio de paquetes de información para lograr la comunicación entre dos servidores o más dispositivos a través de una red. [3]

STM32F401CCXX: Los dispositivos STM32F401xB/STM32F401xC se basan en el núcleo RISC de 32 bits Arm ® Cortex ® -M4 de alto rendimiento que funciona a una frecuencia de hasta 84 MHz. El núcleo Cortex ® -M4 presenta una unidad de punto flotante (FPU) de precisión simple que admite todas las instrucciones de procesamiento de datos y tipos de datos de precisión simple de Arm. También implementa un conjunto completo de instrucciones DSP y una unidad de protección de memoria (MPU) que mejora la seguridad de la aplicación.

El STM32F401xB/STM32F401xC incorpora memorias integradas de alta velocidad (hasta 256 Kbytes de memoria Flash, hasta 64 Kbytes de SRAM) y una amplia gama de E/S y periféricos mejorados conectados a dos buses APB, dos buses AHB y un 32 Matriz de bus multi-AHB de bits.

Todos los dispositivos ofrecen un ADC de 12 bits, un RTC de bajo consumo, seis temporizadores de 16 bits de uso general, incluido un temporizador PWM para el control del motor, dos temporizadores de 32 bits de uso general. También cuentan con interfaces de comunicación estándar y avanzadas.

El STM32F401xB/STM32F401xC funciona en el rango de temperatura de - 40 a + 125 °C desde una fuente de alimentación de 1,7 (PDR OFF) a 3,6 V. Un conjunto completo de modo de ahorro de energía permite el diseño de aplicaciones de bajo consumo.

Estas características hacen que los microcontroladores STM32F401xB/STM32F401xC sean adecuados para una amplia gama de aplicaciones. [4]

2. Procedimiento

En primer lugar se aclara que la tarea principal de esta práctica de laboratorio es medir y controlar las revoluciones por minutos (RPM's) de un motor DC mediante una señal PWM que emite un pulso a una determinada frecuencia, el cual resulta en una proporción de energia aplicada según la duración que tenga, el valor deseado de estas RPM's será seleccionado y enviado a través de una interfaz de usuario en el PC, diseñada y ejecutada en Qt Creator, en la cual es posible seleccionar un valor de RPM's entre 0 y 1000 que son las máximas RPM's que permite el motor utilizado (Figura 1) cuando esta siendo alimentado con 6 V y enviar este valor al microcontrolador STM32F401CCU6 para que este realice el ajuste de manera continua al trabajo de emisión de la señal PWM ejecutándose, por lo que primero se hará la configuración general del programa que debe tener el STM32F401CCU6 para realizar las tareas que se requieren, las cuales son activación del TIM, configuración de su frecuencia, CP, activación del sistema PWM con ajuste del valor de comparación respecto al CP del TIM denominado (CCR2), configuración de un canal del TIM para la medición de las RPM's del motor, configuración de comunicación USB Dual y procesamiento de información.

Figura 1: Motor de 1000 RPM's/6V utilizado como actuador respecto a la señal PWM.

Así que al crear el proyecto de STM32CubeIDE se configura el High Speed Clock (HSE) con el Crystal/Ceramic Resonator, Luego se predetermina el Debug en Serial Wire y el Timebase Source con la opción Serial Wire, para con todo lo anterior colocar la frecuencia de funcionamiento del reloj interno (HCLK) en 84 MHz, con esto se finaliza la configuración general del sistema de la STM32, pasando ahora al desarrollo relacionado con la USB, aun en el archivo .ioc, en la sección de Connectivity-USB_OTG_FS, se configura el modo como dispositivo únicamente ya que no es necesaria otra configuración mas avanzada, y allí mismo se habilita la interrupción global "USB ON The Go FS", para luego en la sección de middleware, que actúa como una capa de traducción para comunicaciones entre dos dispositivos, allí se configura el USB_DEVICE como "Communication Device Class (Virtual Port Com)". Con este aspecto se finaliza el segundo de tres aspectos generales a configurar antes de generar el código de compilación del proyecto. Por ultimo en la sección de Timers, se selecciona el TIM2, donde primero para la fuente de reloj se selecciona el internal clock que ya fue configurado anteriormente, después en el Channel 2 se selecciona la opción de PWM generation CH2, que asocia un pin físico a esta configuración que para este caso puntual fue el Pin A1, después de esto se tienen algunas características del TIM como el pre-escaler y el Counter Period (CP), las cuales se modificaron en varias oportunidades para evidenciar la influencia en la aplicación, por lo que por ahora se dejan los valores predeterminados, por ultimo respecto al TIM en el Channel 1 se selecciona la opción denominada Input Capture Direct Mode, la cual asocia un pin a este modo y se configura con la polaridad de Rising Edge, que implica una acción de recopilación de datos cuando ocurra un cambio de 0 a 1 en el pin asociado. Luego de esto se genera el código y en el main, se agregan llamados a funciones que están bibliotecas anexas para iniciar la configuración de GPIO, TIM2 Y USB_DEVICE, continuamente a esto se encuentra dos funciones para iniciar el PWM y permitir variar el valor del CCR2 para la generación del pulso, además de estas funciones se tiene en la primera posición la declaración de la interrupción del TIM y con se cuente a esto esta la función de interrupción de la modalidad Input Capture que respectivamente, todas las 4 funciones anteriores son:

- HAL_TIM_Base_Start_IT(&htim2);
- HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
- HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_2);
- __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2,0);

En donde la cuarta función, tiene el tercer parámetro de entrada en 0, el cual representa el CCR2, que en este caso indica que no se genera ningún pulso. Luego de esto se procede a constituir las funciones relacionadas con el modo Input Capture del Channel 1, junto la bandera de overflow del TIM, así es que en el código siguiente se encuentran ambas funciones respectivamente, y se destaca primero que la función del Input capture mide la cantidad de ”pasos.º ciclos de reloj que ha realizado el TIM, entre dos eventos específicos teniendo en cuenta la cantidad de overflow que ha hecho el TIM como se evidencia en la segunda función que cada que la bandera se activa se aumenta en una unidad, ya que la flag de overflow solo es un aviso que se ha llegado al limite definido mas no cuenta las veces que se ha realizado por lo tanto se debe configurar manualmente.

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef* htim)
{
    if(medida == 0)
    {
        p1=TIM2->CCR1;
        of= 0;
        medida=1;
    }
    else if(medida==1)
    {
        p2 = TIM2->CCR1;
        pasos=(p2 + (of * 1000)) - p1;
        medida=0;
    }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
{
    of++;
    cienms++;
}
```

Antes de continuar con la explicación del eje central de este laboratorio que es la configuración PWM a partir del TIM y la medición de RPM's por ciclos de reloj, se explica a profundidad la sintaxis y procedimiento del sistema USB de comunicación integrado iniciando con el llamado de manera global de la función para transmisión de datos que es:

```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len);
```

Luego de esto se realiza un proceso de intervención para la recepción de datos, primero creando una función de tipo weak en la librería de la USB para llamarla dentro de la función predeterminada de la librería que recibe la información, y poder acceder a la información desde el main, allí se procesan y asignan los datos recibidos según corresponda que para este caso, se basa en la asignación de un valor de funcionamiento para el PWM que consiga unas revoluciones especificas en el motor. Volviendo al PWM del TIM se fijaron los valores de preescaler en 8400, lo que conlleva a una frecuencia de 10 kHz, que puede que sea una frecuencia alta para el motor pero se ve compensada con el CP que tiene un valor de 1000-1 unidades, es decir que la resolución del sistema PWM es 0.1 % y el Delay colocado en while infinito es de 50 ms.

Continuando ahora con la sintaxis de funcionamiento del programa se tiene el código existente en el while(1) que se encuentra a continuación:

```
rpmsmed=150000/pasos;
    diffrpms=rpmsrecep-rpmsmed;
    if(diffrpms!=0)
    {
        if((rpmsrecep+(diffrpms))>1000)
        {
            TIM2->CCR2=1000;
        }else{
            TIM2->CCR2=(rpmsrecep+(diffrpms));
        }
    }
    if((cienms≥3)&&(rpmsrecep!=0))
    {
        cienms=0;
        for(k=0;k<3;k++)
        {
             packtosend[k+2]=((rpmsmed)>>(8-(k*4)))&0x0F;
```

```
                packtosend[k+5]=((rpmsrecep)>>(8-(k*4)))&0x0F;
            }
            if(rpmsmed==0)
            {
                packtosend[2]=0x00;
                packtosend[3]=0x00;
                packtosend[4]=0x00;
            }
            chesum=0;
            for(k=0;k<8;k++)
            {
               chesum=chesum+packtosend[k];
            }
            packtosend[8]=chesum;
            CDC_Transmit_FS(packtosend,packtosend[1]);
        }
        HAL_Delay(50);
```

Allí se puede destacar en primer lugar el calculo de las revoluciones por minuto que se están midiendo con el sensor CNY70 (Figura 2) que funciona bajo el principio de un octoacoplador, en donde su salida del emisor va al pin asociado del Input Capture direct mode del microcontrolador, en un configuración de Pull-up, entonces para el calculo de las RPM's primero se debe aclarar que la variable consiste en los ciclos de reloj que tarda en pasar de un aspa a otra es decir un cuarto de giro, que luego de esto se multiplica por 4 obtener los pasos que tendría en un giro completo, después de esto se divide la frecuencia del TIM sobre este valor para hallar la frecuencia con la ocurre o bajo la interpretación puntual de este laboratorio serian vueltas en un segundo y para escalarlo a vueltas por minuto simplemente se multiplica por 60, para hallar el valor de RPM's, así que expresado matemáticamente se ve de esta forma:

$$RPM's = 60 * \frac{FREQ\_TIM}{(pasos * 4)} \tag{1}$$

(2)

Por lo que al reemplazar valores y simplificar se obtiene el siguiente procedimiento, para justificar la primera expresión en el código:

$$RPM's = 60 * \frac{10kHz}{(pasos * 4)} \rightarrow RPM's = \frac{60kHz}{(pasos * 4)} \rightarrow RPM's = \frac{150000}{(pasos)} \tag{3}$$

(4)

Figura 2: Sensor infrarrojo CNY70 de carácter octoacoplador utilizado para la medición de la RPM's del motor.

Cabe aclarar que en la función de interrupción de comunicación USB se encuentra la asignación del valor para la variable "rpmsrecepz su puesta inicial en en CCR2 del TIM para el posterior ajuste del PWM, que es lo que continua en el código, donde primero se calcula la diferencia entre las RPM's recibidas y las medidas para luego entrar en un condicional si esta diferencia no es 0, ya que puede ser negativa o positiva, el control aplicado es que se le asigna al CCR2 el valor de las RPM's recibidas mas la diferencia entre RPM's, aunque si este valor supera el máximo permitido, se colocara la máxima potencia del sistema, y la diferencia al ser negativa resta valor en el CCR2 que conlleva a una disminución de energia entregada.

Luego de esto hay un condicional para la transmisión de datos que cuenta tiene en primer lugar la variable cienms que esta aumentándose en la interrupción por overflow del TIM, y su comparación es mayor o igual a 3 para que aproximadamente cada 300 ms se realice lo que esta dentro y la condición rpmsrecep!=0, es para que no sobrecargue el puerto de información cuando no se ha enviado nada. Entonces en este lugar es donde se ingresa la Data en el paquete de emision que esta conformado de la siguiente manera (Cuadro 1) para la comunicación desde el STM hacia PC.

INICIO	TAMAÑO	D1	D2	D3	D4	D5	D6	Checksum	FINAL
0x16	0x0A	D1	D2	D3	D4	D5	D6	$\sum_{k=0}^{7} P_k$	0x19

Cuadro 1: Protocolo del paquete de transmisión de datos del microcontrolador STM32F401CCU6.

Entonces las RPM's medidas se envían en los espacios de data desde D1 hasta D3 y desde D4 hasta D6 se envían las RPM's recibidas, solo para corroborar la correcta recepción, procesamiento y emisión de los datos, antes de realizar el Checksum se hace una aclaración de valores sobre las RPM's medidas para que no envie un valor anterior o erróneo, ahora respecto al calculo de la operación no convergente se hace mediante la suma de todos los términos anteriores a la casilla donde va este valor en un variable de tipo unsigned char para que sea cíclica la suma y se mantenga entre 0 y 255, al finalizar el calculo se llama la función de transmisión y se envía el paquete especificando el tamaño como el valor que este en la posición 1.

Cambiando de enfoque a la parte de Qt Creator, donde se creo y diseño un Widget para la interfaz de usuario que se evidencia en la Figura 4, ventana que tiene características especiales intrínsecas, primero hay un Combo Box que contiene los puertos disponibles en el momento de la ejecución y con el botón del lado es posible cerrar y abrir el puerto conectándolo con el programa de Qt, en la parte izquierda hay un botón "SET RPM's y un line edit debajo que en conjunto pueden colocan un valor especifico en el deslizador, el cual es el que dictamina el valor de RPM's que se quieren enviar y este valor aparece en la parte derecha debajo de un botón enviar, el cual al ser oprimido envía toma el valor que este debajo lo ingresa en el paquete de emisión (Cuadro 2) desde Qt hacia el microcontrolador que se evidencia debajo de la Figura 4, por ultimo al lado de la frase "$VALORDERPMRECIBIDO:$". se encuentra un cuadro de texto que coloca el valor de RPM's medidas y transmitidas por el STM, es decir el valor que se distribuye en los 3 primeros espacios de DATA en el protocolo del STM (Cuadro 1). Alguna información adicional se evidencia y se reconstruye en la consola del Qt creator.

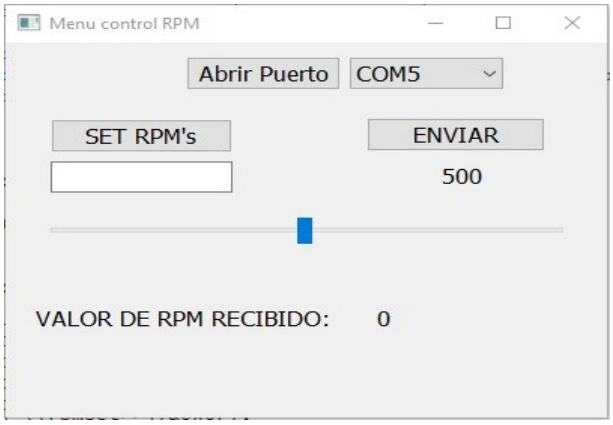

Figura 3: Widget de Qt utilizado para el control.

INICIO	TAMAÑO	D1	D2	D3	Checksum	FINAL
0x16	0x0A	D1	D2	D3	$\sum_{k=0}^{4} P_k$	0x19

Cuadro 2: Protocolo del paquete de transmisión de datos de la interfaz gráfica de usuario de Qt Creator

Hablando de manera general, al cargar el programa en el STM32 el motor esta totalmente quieto ya que el CCR2 se encuentra en 0, y no hay transmisión de información, luego de esto se conecta el cable USB al PC y se ejecuta el programa de Qt, en donde escoge el puerto relacionado al STM que es el COM8, al hacer el primer envío de RPM's el microcontrolador realiza la asignación de valor en el CCR2 e inicia el ajuste de PWM para llegar a las revoluciones indicadas mientras cada 300 ms envía el valor de las RPM's medidas cuyo valor se imprime en el cuadro de texto al inferior de la ventana.

Ahora hablando para el ámbito de graficación de los datos, no se realizo en excel, sino se implemento directamente en Qt creator para que los datos se vayan mostrando en tiempo real y así ahorrarse tiempo del traspaso de datos y funcionalidades de visualización en excel. Así que de manera gráfica se tiene un cuadro adicionado en la ventana de Widget que tiene divisiones de magnitud y escala, ademas de avance segun el valor que reciba por la comunicación USB. Un grafica de una sección representativa de la grafica se evidencia en la Figura

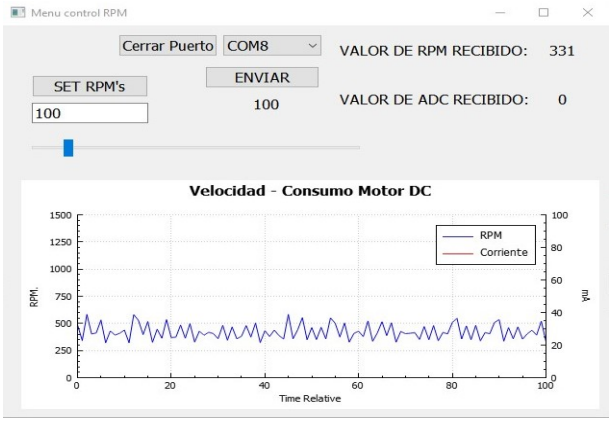

Figura 4: Gráfica resultante de una muestra representativa.

3. Conclusiones

- Se puede concluir en este laboratorio que gracias a la STM32 la energía que se le entregue a un motor se controla fácilmente con el módulo PWM donde se comprendió y entendió su funcionamiento para en la práctica acertar en una frecuencia óptima para un motor DC de 12V para generar una velocidad deseada
- En conclusión al experimentar con varias valores de CP y preescaler se determino que la mejor opción de configuración es un preescaler de 42000 para que el TIM tenga una frecuencia resultante de 2 kHz y completado a esto un CP de 200, para obtener una resolución de 0,5
- En conclusión, se puede afirmar que gracias a la practica de prueba y error se aproximó de la mejor manera, al método mas óptimo de aplicación para este problema planteado, además que el PWM es una herramienta útil con un amplia gama de aplicación en el ámbito de la electrónica enfocado en el control de motores específicamente para su precisión, en arranque o funcionamiento a largo plazo.

4. Referencias

[1] ¿Qué son y para qué sirven los protocolos de comunicación de redes? (s. f.). Recuperado 29 de septiembre de 2022, de https://www.kionetworks.com/blog/data-center/protocolos-de-comunicaci %C3 %B3n-de redes#: %7E:text=Un %20protocolo %20es %20un %20conjunto,a %20trav %C3 %A9s %20de %20una %20red.

[2] Qt Company. (s. f.). Embedded Software Development Tools — Cross Platform IDE — Qt Creator. Recuperado 29 de septiembre de 2022, de https://www.qt.io/product/development-tools

[3] ¿Qué es PWM y cómo usarlo? (s. f.). Recuperado 29 de septiembre de 2022, de https://solectroshop.com/es/blog/que-es-pwm-y-como-usarlo–n38