NFORME PRACTICA DE LABORATORIO 6

Santiago Nomesque – Jair Munevar

Universidad Sergio Arboleda

¿QUE USAMOS?

- Microcontrolador STM32F411CEU.
- ST LINK V2.
- Programa STMCUBE IDE.
- Motor DC.
- Sensores IR de herradura.
- Puente HTB6612FNG.
- Programa QT Creator.

PROTOCOLO PARA ENVIAR RPM Y TIEMPO

Para este laboratorio se implementó un protocolo de comunicación básico creado por nosotros, donde se utilizó un array de siete posiciones, donde la primera indica el inicio del envió de datos con un 0x0F, luego en la segunda se establece el tamaño del array según la información enviada, de siete bytes, para los datos que deseamos enviar se usan las posiciones tres, cuatro y cinco, de modo que en la tres y cuatro se envían las RPM y en la cinco el valor del TIEMPO. La posición seis se usa para el CHK, el cual establecimos una operación XOR entre los datos de la posición tres, cuatro y cinco y el numero decimal 15. Para finalizar nuestra comunicación en la posición siete se envía el valor 0xFF que funciona como indicativo de que el array de información llego a su fin.

QT CREATOR

QT Creator es un entorno de desarrollo integrado (IDE) para desarrollar aplicaciones utilizando el framework Qt. Se usa para crear aplicaciones gráficas de escritorio y móviles, gracias a su capacidad para manejar interfaces de usuario y soporte para múltiples plataformas. Este software se utilizó para crear una interfaz gráfica en la que se visualizará el valor de las RPM de nuestro motor y se tendrá la posibilidad de controlar el sentido de este mediante unos botones.

¿QUE SE REALIZÓ?

Se creo un protocolo para enviar información desde la STM a nuestro PC por medio de USB, y se usó la función CDC_Transmit_FS para enviar la información. Para los datos que se agregaban al arreglo de datos, se dividió el valor de RPM en dos partes ya que esta variable es de tipo UINT16 es decir de dos BYTES, logrando tener el valor de RPM en dos BYTES aparte pero que al unirlos en el orden correcto representa el valor de RPM real. La variable Tiempo también se envió en el array, sin embargo, esta variable al ser UINT8 no se separó, únicamente se envió tal cual se leyó en el programa. Adicionalmente se implementó la interfaz gráfica que se creó usando QT, para ello se entendió el funcionamiento básico de los OBJETOS, las ACCIONES que tienen estos y los SLOTS para poder realizar la comunicación de acuerdo con lo que se desea ejecutar. Primero se programaron las etiquetas que nos permitirían recibir la información desde nuestra STM, para este envió de información se utilizó el protocolo ya mencionado, para la recepción de la información desde el protocolo, se almaceno la información por posición del array en una variable y luego se realizaron las operaciones de desplazamiento de bits y operación lógica OR para conformar el valor completo de las RPM. Una vez logrado lo anterior, procedimos implementando los botones para controlar el encendido, apagado y giro de nuestro motor, para ello se programa que, al hacer clic en alguna de las opciones mencionadas, se enviara por el USB la información del comando asignado para controlar esto, siendo los números 116, 117, 51 y 52 para encendido, apagado, giro a la derecha y a la izquierda, respectivamente. Cabe aclarar que la información se envió en formato Hexadecimal.

¿QUÉ SUCEDIÓ?

Al enviar la información de las RPM a la interfaz de QT, se estaba realizando el correcto envió y recepción de los datos, pero al unir los bytes en QT, en algunos casos no se realizaba de manera correcta la operación y provocaba que el dato no fuera el correcto, esto se solucionó cambiando el tipo de variable donde se almacenaban los bytes, siendo que estos estaban como INT y se cambiaron a UINT8_T y el valor de las RPM total, estaba igualmente como INT y se dejó como UINT32_t. En la parte grafica decidimos unir la etiqueta del texto junto con la del número y esto inicialmente nos generó un problema ya que cuando las RPM superaban las 999, el dato se visualizaba de manera incompleta, esto se arregló estirando el cuadro que contenía estas etiquetas y ya quedo bien. Por último, al pulsar el botón en la interfaz para realizar cualquier acción del motor, el dato enviado, no se estaba recibiendo de la manera correcta en la STM, esto se debía a que en el archivo usbd_cdc_if.c en la función de recibir, se había digitado mal un parámetro para la función de la recepción, este parámetro le estábamos ingresando la dirección de la manera &buf, siendo esto "`CDC_ReceiveCallBack(&Buf,Len[0]);`" y la solución a este problema fue darle el parámetro correctamente, es decir sin el "&" quedando este como buf, de esta manera "`CDC_ReceiveCallBack(Buf,Len[0]);`" y así ya se solucionó el código y se logró el funcionamiento correcto.

DIAGRAMA DE FLUJO

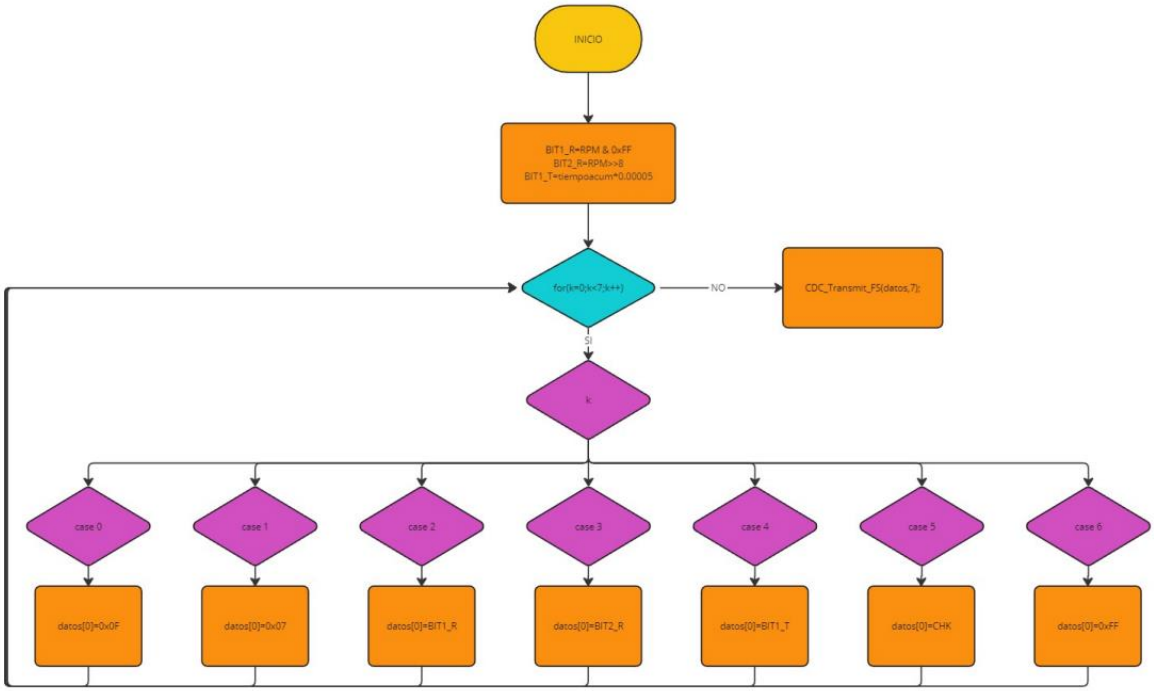

Imagen 1: Diagrama de flujo del código de la STM32 para él envió de datos

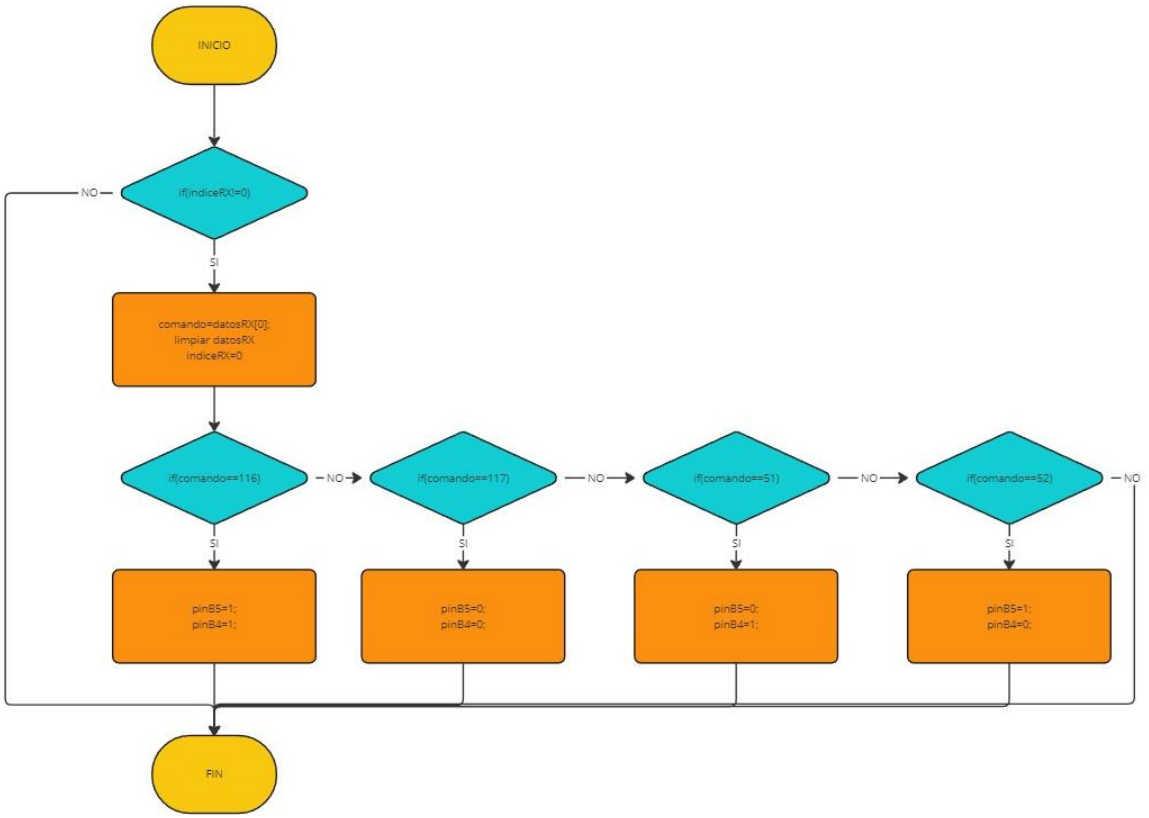

Imagen 2: Diagrama de flujo del código de la STM32 para la recepción de datos

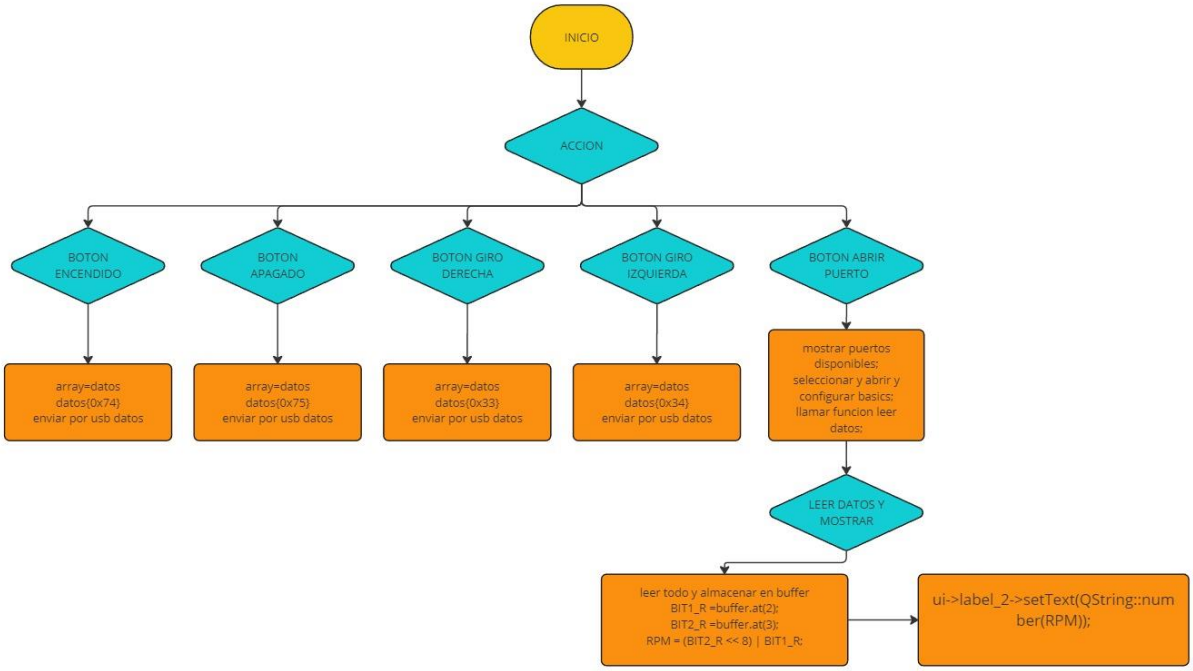

Imagen 3: Diagrama de flujo del código de QT para la recepción de datos

INTERFAZ DE QT

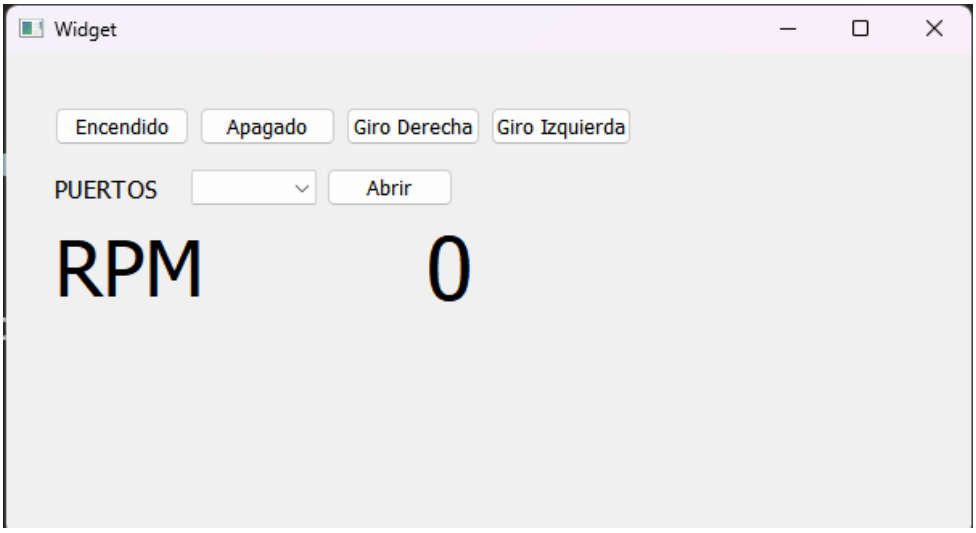

Imagen 4: Imagen de la interfaz creada con QT

CONCLUSIONES

- La implementación de un protocolo de comunicación básico permitió la transmisión efectiva de datos entre el microcontrolador STM32F411CEU y la interfaz gráfica en QT, logrando el control de las RPM del motor y el tiempo de operación de forma confiable.

- El manejo incorrecto de tipos de variables durante la transmisión y recepción de datos en la interfaz gráfica ocasionó errores de visualización y cálculo de las RPM. Esto se resolvió cambiando las variables a tipos más adecuados (UINT8_T para los bytes y UINT32_T para el total de RPM), asegurando la precisión en la operación y la visualización.

- La implementación del código de recepción en la función CDC_ReceiveCallBack del archivo usbd_cdc_if.c contenía un error en la asignación de parámetros, lo que impedía la correcta recepción de comandos para controlar el motor. Este problema se solucionó al eliminar el uso incorrecto del operador & en los parámetros, lo que restableció el funcionamiento esperado del sistema.

- La interfaz gráfica creada en QT permitió no solo visualizar las RPM del motor, sino también controlarlo de manera interactiva. La solución a problemas de diseño,

como la visualización incorrecta de valores superiores a 999 RPM, se logró ajustando los elementos de la interfaz, mejorando la experiencia de usuario.

- La programación modular y el uso de objetos en QT permitieron crear una interfaz funcional y escalable. Al comprender el funcionamiento de las acciones y los slots en QT, se logró una interacción fluida entre la interfaz y el microcontrolador, facilitando el control y monitoreo del motor.