INFORME PRACTICA DE LABORATORIO 4

Santiago Nomesque – Jair Munevar

Universidad Sergio Arboleda

¿QUE USAMOS?

- Microcontrolador STM32F411CEU.
- ST LINK V2.
- Jumpers.
- Diodos.
- TSOP4838.
- Motor DC.
- 2 encoder o sensor IR de herradura.
- Puente H TB6612FNG.
- Pantallamoled-l2C
- Protoboard.
- Programa STMCUBE IDE.
- Control remoto televisor Sony.

PROTOCOLO SONY

En este protocolo se utiliza una señal portadora derivada de 480kHz con un ciclo de cada trama de 45ms. Una trama se compone de un pulso de sincronía (start), un dato de 7 bits (command) y un código de identificación del dispositivo de 5 bits (address), como se ve en la figura 1

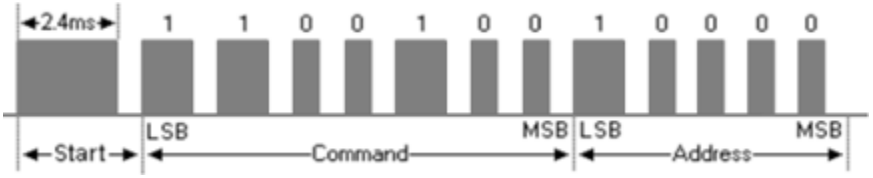

Figura 1. Trama del protocolo SIRC de 12 bits.

En esta trama del protocolo se utiliza una base de tiempo fundamental de 600µs y la señal se envía en múltiplos de esta base de tiempo (T). La codificación utiliza un Bit de inicio que le indica al receptor que le enviarán datos, en este caso es un pulso de tamaño 4T (2.4ms). En seguida se envían los 1's y 0's de la siguiente forma: un bit 0 se manda un espacio (bajo) T y un alto también con una duración T; mientras que un bit 1 se manda un espacio (bajo) T y un alto con una duración de 2T (1.2ms). El protocolo puede mandar datos de 12, 15 y 20 bits, como se muestra en la figura 2

Figura 2. Codificación de datos en protocolo SIRC.

Para estas tres variantes se observa que el protocolo se limita a leer los tamaños de datos de T, 2T y 4T. Después del inicio, las dos partes importantes de datos son el comando (tecla presionada del control remoto) y el dispositivo (el equipo al que se le manda el comando).

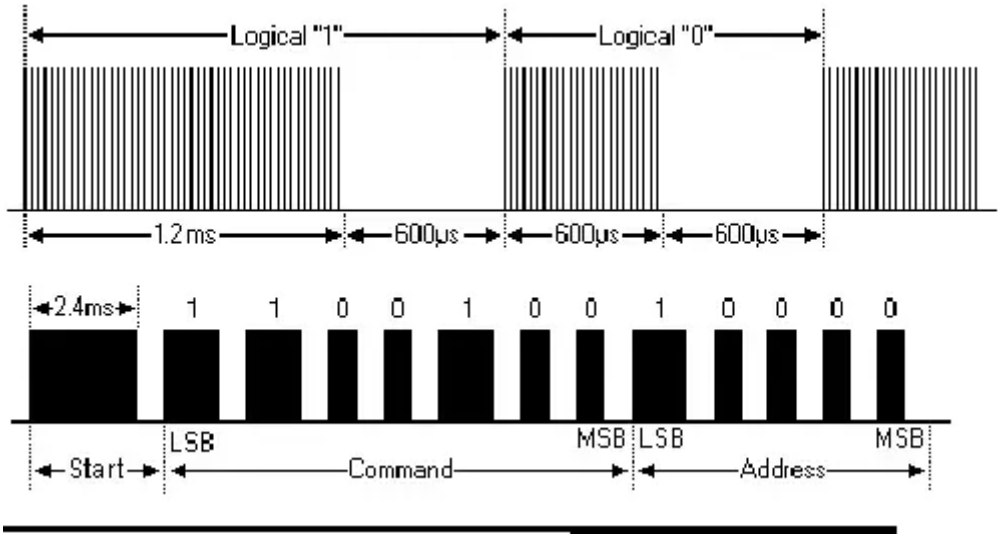

Figura 3. Tren de pulsos y muestra grafica de un bit 1 y bit 0

La figura 3 muestra un **Tren de pulsos** del protocolo SIRC. El bit de inicio (Start) tiene una duración de 2.4ms y después vendrá su respectiva separación de 0.6ms como en cualquier bit. Entonces los 7 bits de comando son transmitidos, seguidos de los 5 bits de dirección del dispositivo, en el ejemplo de la figura se transmite la dirección 1 y el comando 19

Comando (Hex,D)	Botón presionado	Comando (Hex,D)	Botón presionado
0x0=0d	1	0x10=16d	Chan +
0x1=1d	2	0x11=17d	Chan –
0x2=2d	3	0x12=18d	Vol +
0x3=3d	4	0x13=19d	Vol –
0x4=4d	5	0x14=20d	Mute
0x5=5d	6	0x15=21d	Power
0x6=6d	7	Dispositivo	
0x7=7d	8	1	TV
0x8=8d	9	2	VCR 1
0x9=9d	0	3	VCR 2

Tabla 1. Botones con sus respectivos comandos.

LIBRERIAS PARA EL USO DEL TIM

Para este laboratorio se hizo uso de las librerías HAL_TIM, las cuales nos permitieron contar mientras el código se seguía ejecutando, de esta manera se obtuvo una manera de contar tiempo real. Para inicializar el TIM se usó la función ***HAL_TIM_Base_Start*** seguido de la dirección del TIM que se utilizó**,** luego durante la ejecución del código se usó la función ***__HAL_TIM_SET_COUNTER*** la cual nos permitió establecer la variable de conteo en el valor deseado, en nuestro caso en 0, posteriormente se usó la función ***__HAL_TIM_GET_COUNTER*** para leer la variable encargada de contar del TIM y de esta manera se obtuvo la cantidad de veces que el TIM contó, cabe aclarar que cada vez que cuenta (cada cuanto tiempo se realiza un conteo) se establece en la configuración previa de nuestro microcontrolador en el periférico de los TIMS, en el apartado del preescaler, esto teniendo en cuenta a que valor te tiempo queremos que sea equivalente un TICK o conteo del TIM. Una vez se implementó lo anterior se procedió con el código teniendo en cuenta la equivalencia para así determinar los tiempos de nuestra señal infrarroja.

ITERRRUPCIONES ISR

Las Interrupciones de Servicio de Rutina (ISR) son fundamentales en sistemas embebidos y microcontroladores debido a la necesidad de responder a eventos externos de manera rápida y eficiente.

En un microcontrolador, las interrupciones son señales que informan al procesador sobre eventos que requieren atención inmediata, como la llegada de datos en un puerto de comunicación o un cambio en una entrada de un sensor. Una ISR es un bloque de código diseñado para manejar estos eventos. Cuando ocurre una interrupción, el microcontrolador detiene temporalmente su ejecución actual y ejecuta la ISR asociada para atender el evento. Por ejemplo, si un temporizador alcanza un valor específico, la ISR podría actualizar un contador o realizar una acción periódica.

Los sistemas embebidos, que son sistemas dedicados a realizar tareas específicas (como en electrodomésticos, automóviles, etc.), a menudo dependen de microcontroladores para gestionar múltiples tareas simultáneamente. Las ISRs permiten que estos sistemas respondan de manera oportuna a eventos del mundo real, como interrupciones de hardware o señales de sensores, sin tener que esperar a que el microcontrolador termine otras tareas. Son chips que integran un procesador, memoria y periféricos en un solo paquete. Los microcontroladores utilizan interrupciones y ISRs para manejar eventos de manera eficiente. La capacidad de gestionar interrupciones permite a los microcontroladores ejecutar tareas críticas en tiempo real, algo esencial para la operación de sistemas embebidos.

¿QUE SE REALIZÓ?

Para este laboratorio se debe realizar que, al usar un control remoto, usaremos el control y al presionar un botón el motor se encenderá y así mismo al presionar otro se apagará, por otro lado, se hará de forma similar para que al presionar un botón gire hacia la derecha y otro para que lo haga hacia la izquierda.

Para la siguiente parte se van a contar las RPM (Revoluciones por minuto) haciendo uso del sensor de IR de cruce siendo el valor de este visualizado en la pantalla (mínimo 0.5s de refresco), de manera similar se debe visualizar en la pantalla hacía que sentido está girando sin necesidad del comando, puesto que se moverá con la mano y en la pantalla se visualizara el sentido y RPM, así mismo se visualizara el valor máximo de RPM y se actualizara en tiempo real.

¿QUÉ SUCEDIÓ?

En este Laboratorio se decidió usar el lector del infrarrojo del código anterior para que al momento de ocurrir la ISR esta no tuviera problemas, pero al usarlo se quedaba dentro de la interrupción siempre, para arreglarlo se usó una variable para romper el while que ayudara a tomar la señal una única vez, y si entraba al while pero de manera incorrecta este se interrumpiera.

Luego se implementaron los comandos de girar hacia la derecha, girar hacia la izquierda, prender y apagar el motor, pero no hubo problema, pero al implementar l contador por alguna razón el microcontrolador no funcionaba y luego se arregló al cargar de nuevo el código, pero contando las revoluciones implementando de más un TIM haciendo que sobre.

Luego usamos un TIM que estaba antes para contas así indirectamente las vueltas y así poder sacar las revoluciones por minuto de este, luego nos percatamos que los comandos se imprimían, pero poníamos la pantalla en negro, entonces se quito es parte del código para imprimir los comandos de manera correcta.

DIAGRAMA DE FLUJO

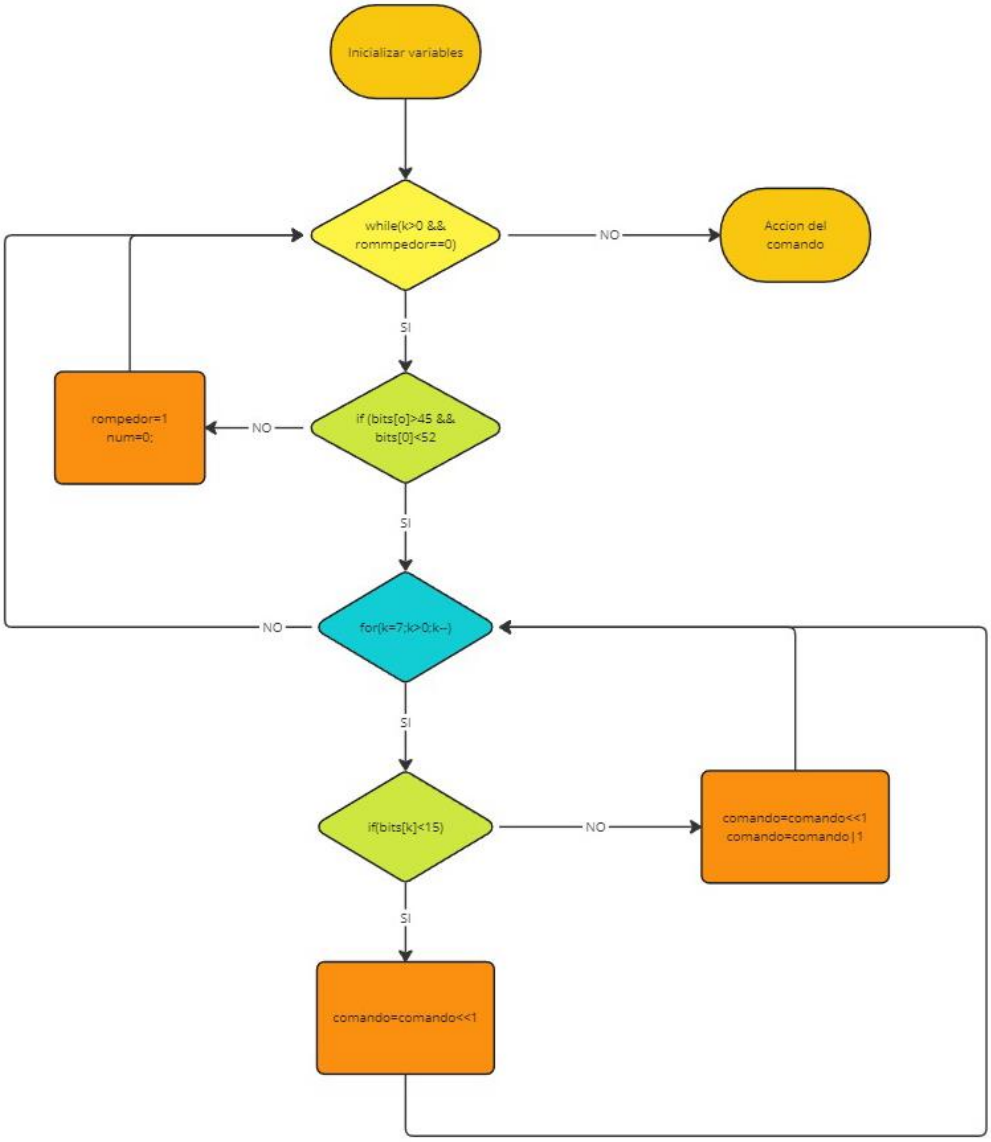

Figura 4: Diagrama de bloques para interpretar la señal

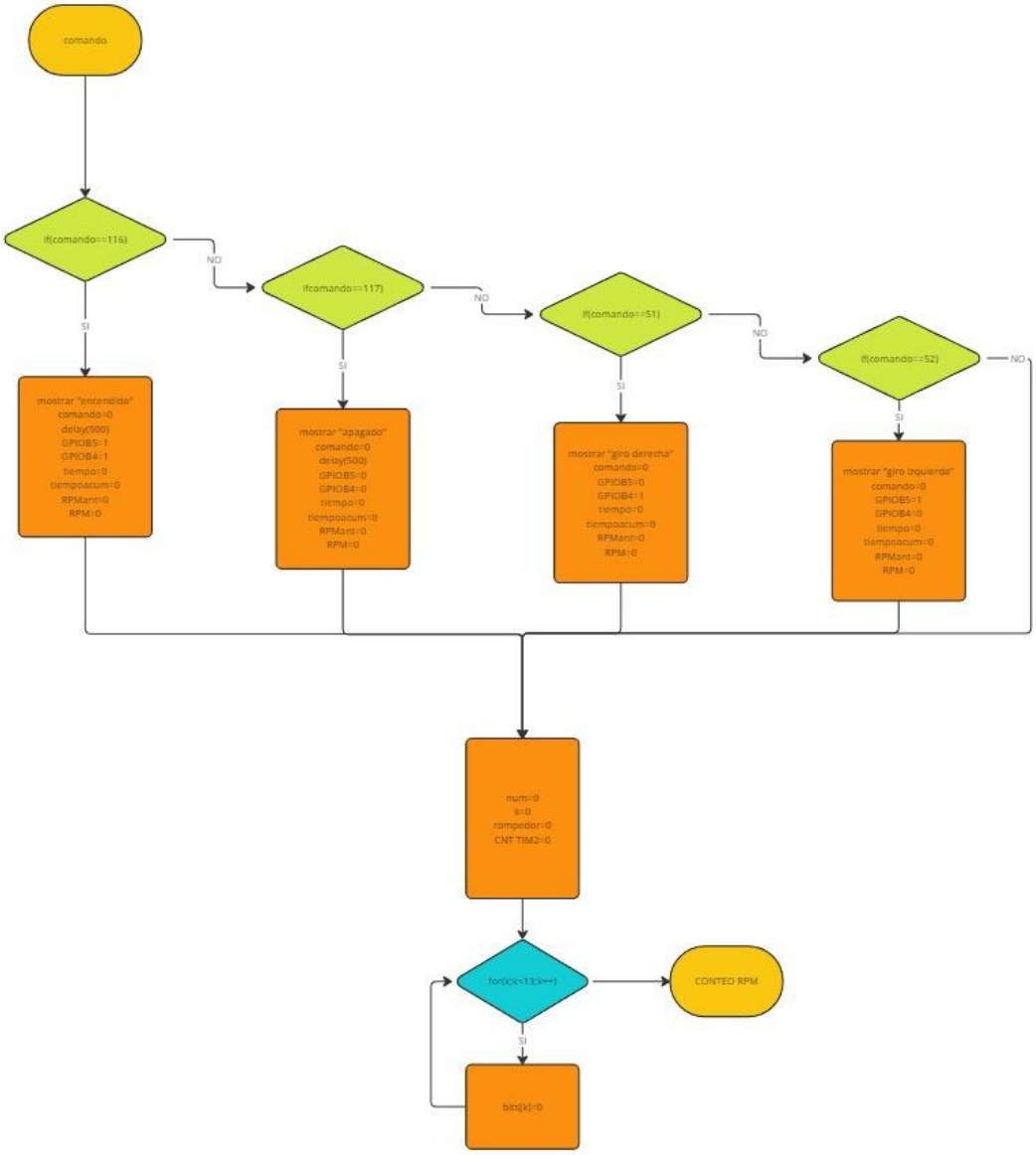

Figura 5: Diagrama de flujo para controlar el motor según el comando

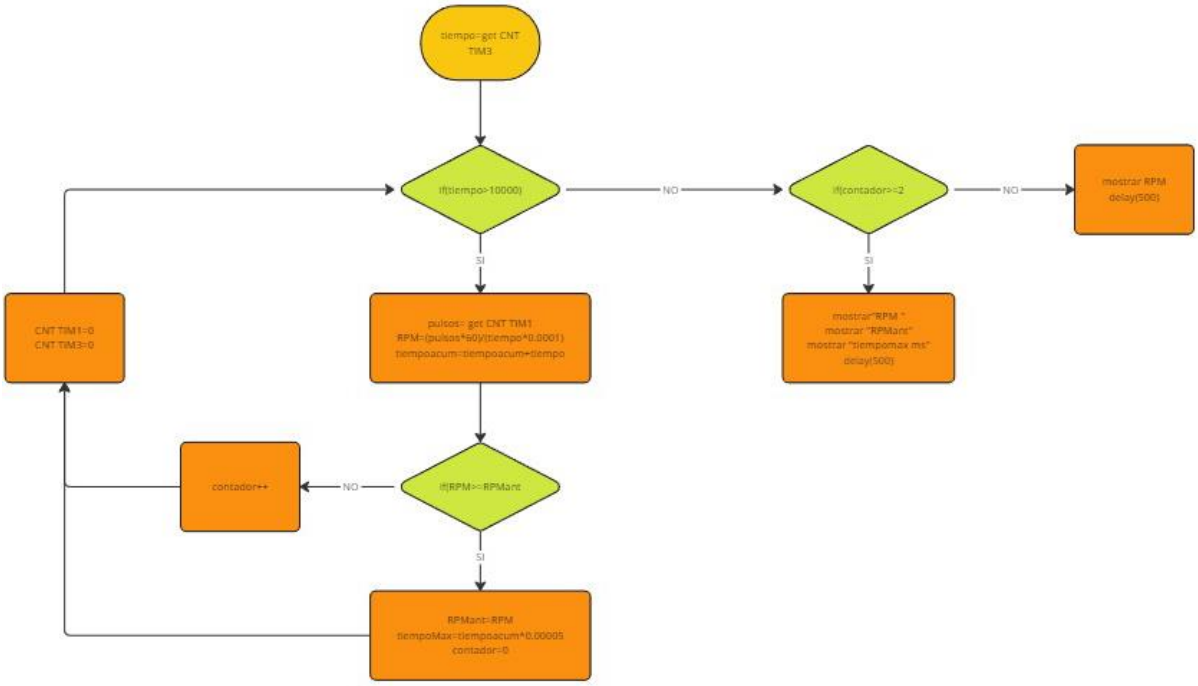

Figura 6: Diagrama de flujo de conteo y muestreo de RPMs

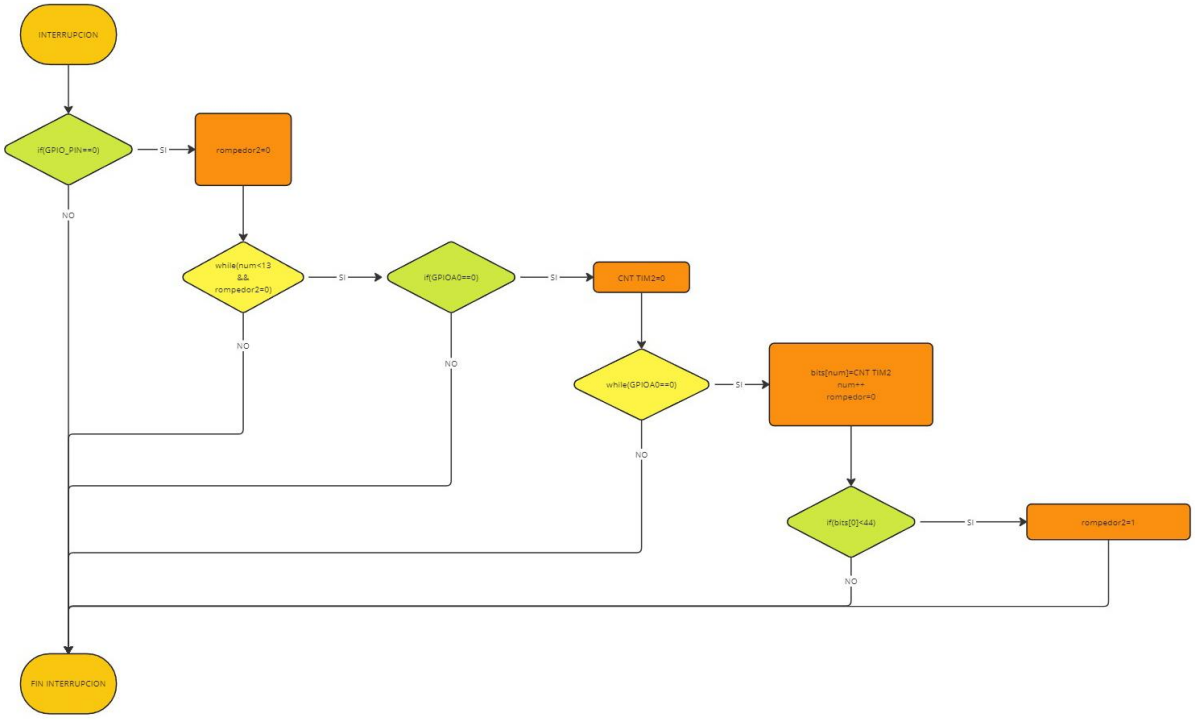

Figura 7: Diagrama de flujo de la ISR

EQUEMATICO

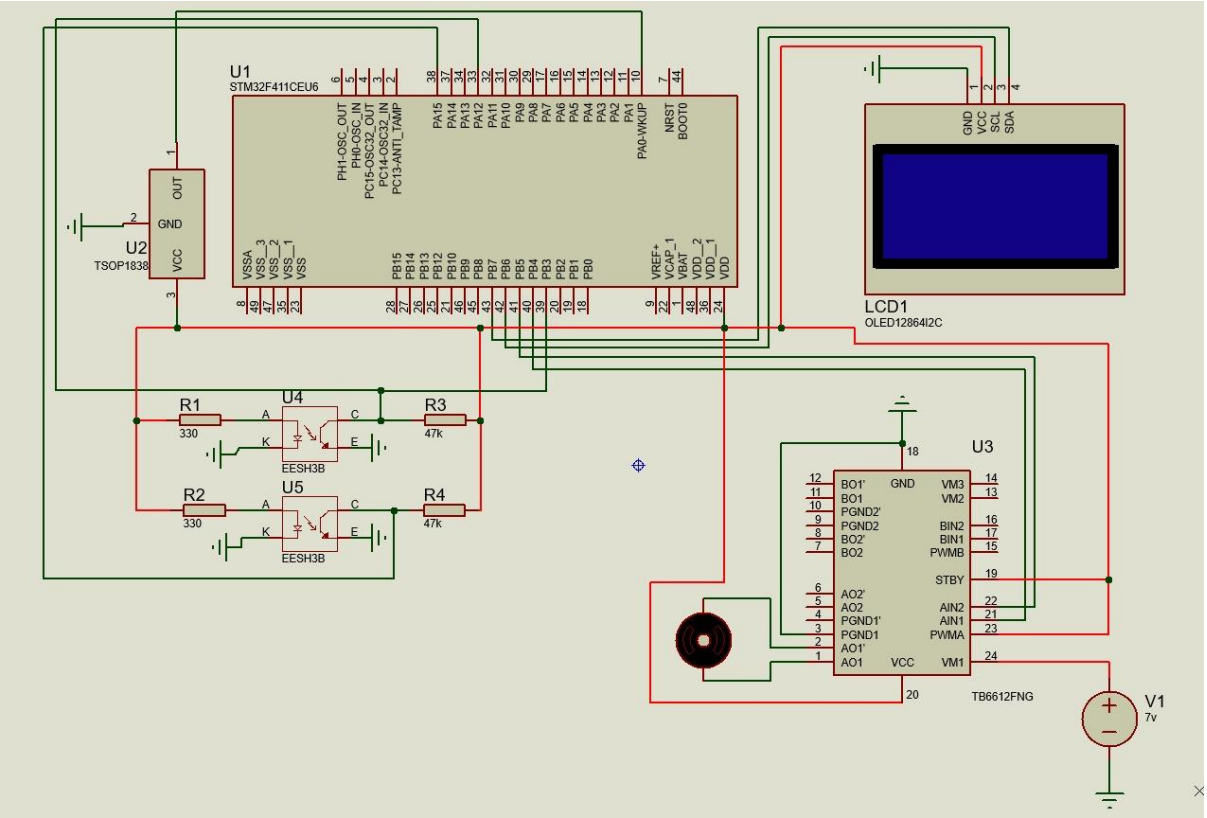

Figura 8. Esquemático del circuito implementado.

CONCLUSIONES

- El uso de un microcontrolador STM32 con sensores de IR y un control remoto permitió implementar un sistema eficiente para controlar un motor DC, ajustando su velocidad y dirección de rotación precisamente en casi todas situaciones donde solo hubo inconvenientes con la implementación del código.
- El sensor de cruce de herradura IR demostró ser adecuado para medir con precisión las revoluciones por minuto (RPM) del motor en tiempo real, permitiendo al sistema actualizar y visualizar este valor continuamente. ESTA

- Detectar y visualizar correctamente el sentido de giro del motor fue crucial para evitar comportamientos inesperados. Esto aseguró que el motor operara correctamente en ambas direcciones, según lo comandado por el control remoto. ESTA
- Al medir el tiempo de respuesta del motor desde el comando de arranque hasta alcanzar su velocidad máxima, se evalúa el rendimiento del sistema y se hace los ajustes necesarios para optimizarlo de forma correcta y eficaz el sensor de la respuesta de control. ESTA
- El proyecto requirió una buena integración entre los componentes de hardware (motor, sensores, puente H) y el software que controla el sistema. Esto permitió que el control remoto pudiera interactuar fácilmente con el sistema para realizar las funciones esperadas sin que hubiera alguna interrupción o interpretación no deseada.