INFORME PRACTICA DE LABORATORIO 3

Santiago Nomesque – Jair Munevar

Universidad Sergio Arboleda

¿QUE USAMOS?

- Microcontrolador STM32F411CEU.
- ST LINK V2.
- Jumpers.
- Diodos.
- TSOP4838.
- 8 resistencias 220 OHM - 1 Resistencia 200 OHM.
- 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

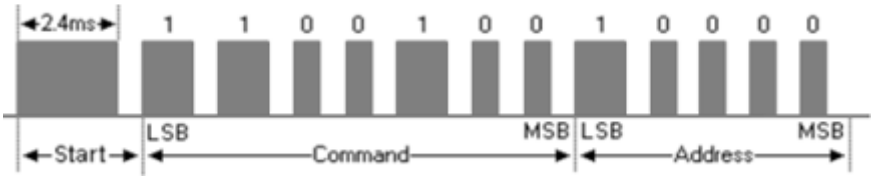

Figura 1. Trama del protocolo SIRC de 12 bits.

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

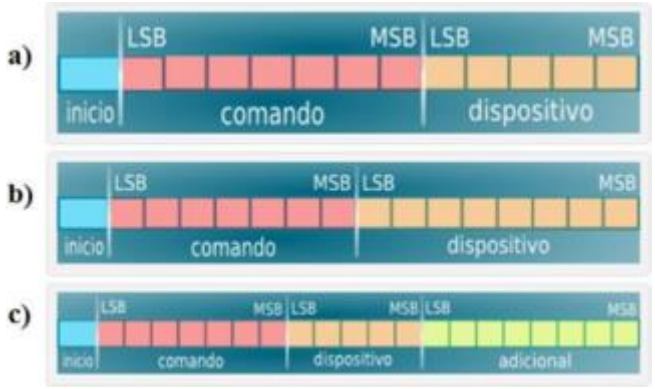

Figura 2. Codificación de datos en protocolo SIRC.

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).

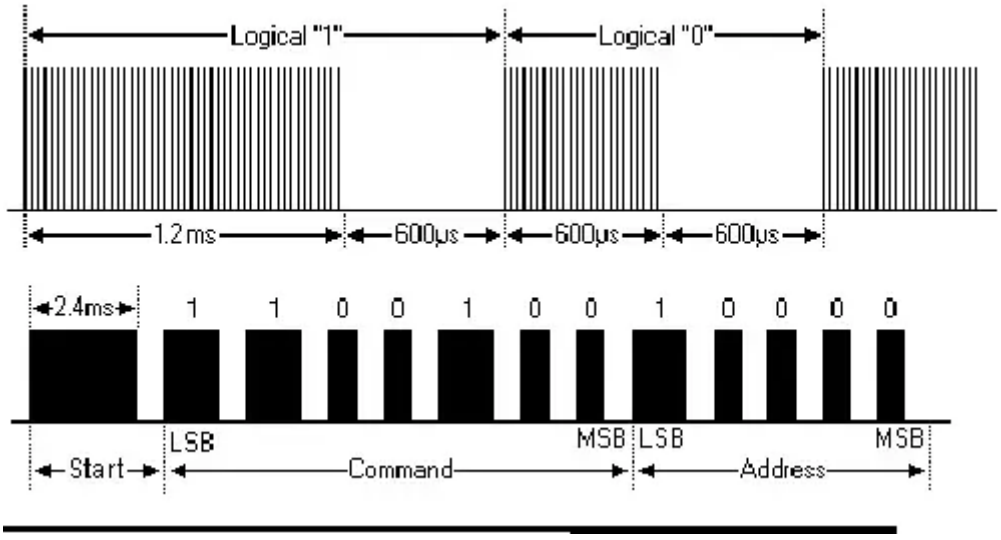

Figura 3. Tren de pulsos y muestra grafica de un bit 1 y bit 0

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.

¿QUE SE REALIZÓ?

Se implemento un código donde al presionar cualquier botón del control elegido (Modelo Sony) en los leds se debe visualizar la señal en binario según correspondan los 0 y 1 del protocolo de Sony. Se debían elegir 8 botones (a elección de cada uno) y al presionarlos se encendieran los leds asignados. Durante la clase se vieron las librerías necesarias para realizar e implementar el código, primero se realizó un boceto del diagrama de flujo acerca de cómo se debía proceder en el código, luego se procedió a implementar y crear el código corrigiendo cualquier error visto, para así realizar las respectivas pruebas.

¿QUÉ SUCEDIÓ?

Durante la creación del código surgieron varios problemas, el primero de ellos es que al oprimir el botón se enviaban muchas señales lo que ocasionaba que no se guardaran bien los datos, esto se solucionó creando una variable que “contara” cuantos estados en 0 transcurrieron y una vez se contaran los 13 estados, no recibiera más información hasta iniciar de nuevo, el segundo error fue que al enviar la señal desde el control las variables solo estaban guardando los 0 en los 1 debido a una mala configuración en los tiempos, luego se vio que al usar muchos ciclos if se veían muchas inconsistencias y problemas por lo que se nos aconsejó usar un ciclo while para que se optimizara el proceso, al cambiar dicho código y usar el ciclo se encontraron muchas incongruencias, entonces se decidió guardar la señal completa para ello se creó un arreglo con el fin de almacenar los tiempos de la señal asignando a cada posición un tiempo, que luego se analizaría si es un uno lógico o cero lógico según corresponda.

Luego de interpretar cada posición del arreglo como un bit y luego en comando y dirección, las direcciones no se estaban guardando, luego de arreglar se vio en el código que cuando guardaba un cero si se desplazaba un bit mientras que con los 1 no se desplazaba, lo que ocasionaba que no se guardaran algunos datos. Al realizar varias pruebas creímos que la señal que enviaba nuestro control era muy vieja ya que los comandos y direcciones enviadas no coincidían con la Tabla 1, pero nos dimos cuenta de que estábamos asignando los bits en orden del más significativo al menos significativo, cuando debía ser del menos significativo al mayor. Para el siguiente punto se tomaron los comandos de los 8 botones elegidos (en nuestro caso los números del 2 al 9 del control) y se pusieron 8 condicionales if cada uno con la condición del comando para encender el led correspondiente.

DIAGRAMA DE FLUJO

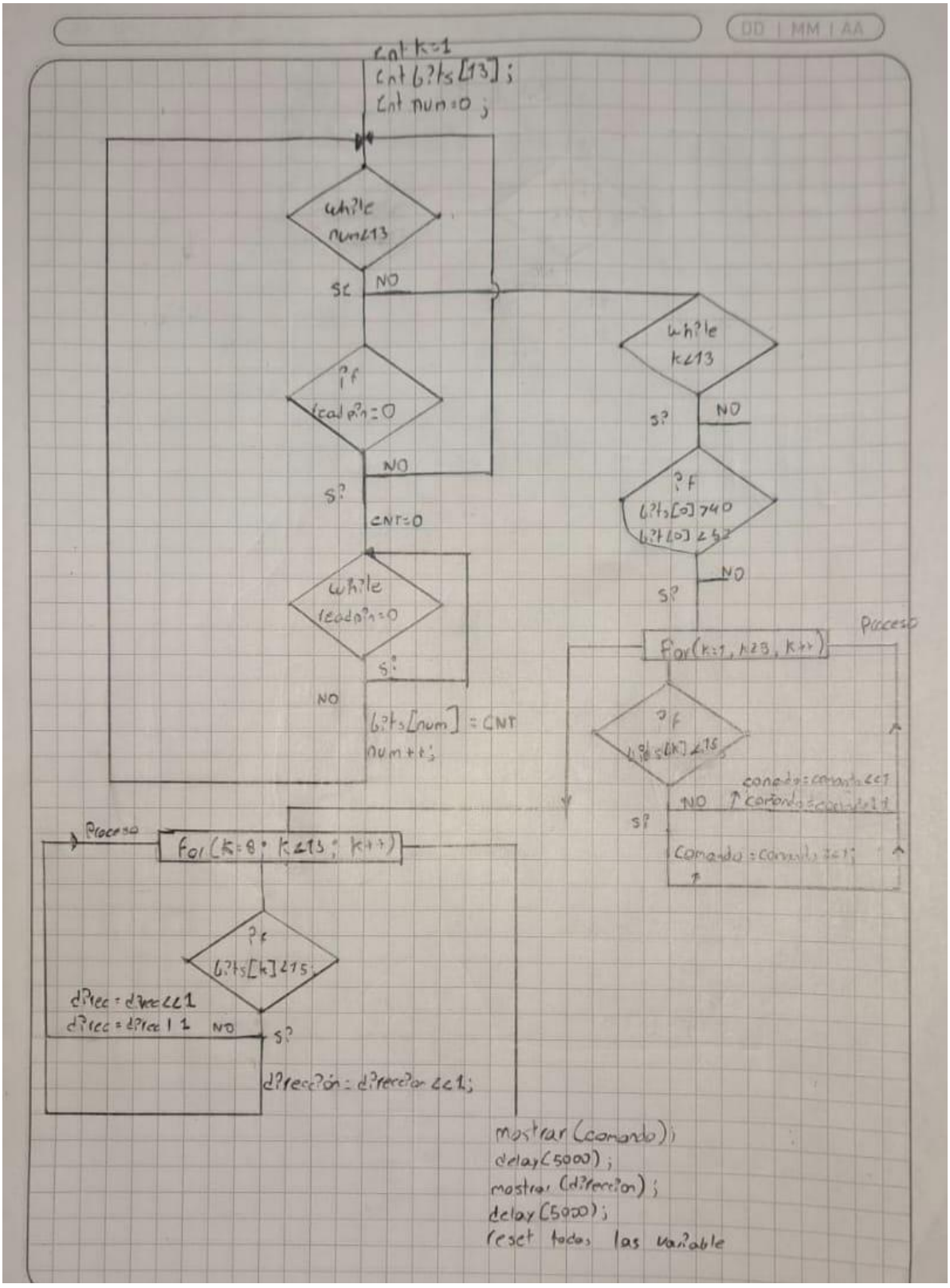

Figura 4. Diagrama de flujo lectura señal IR

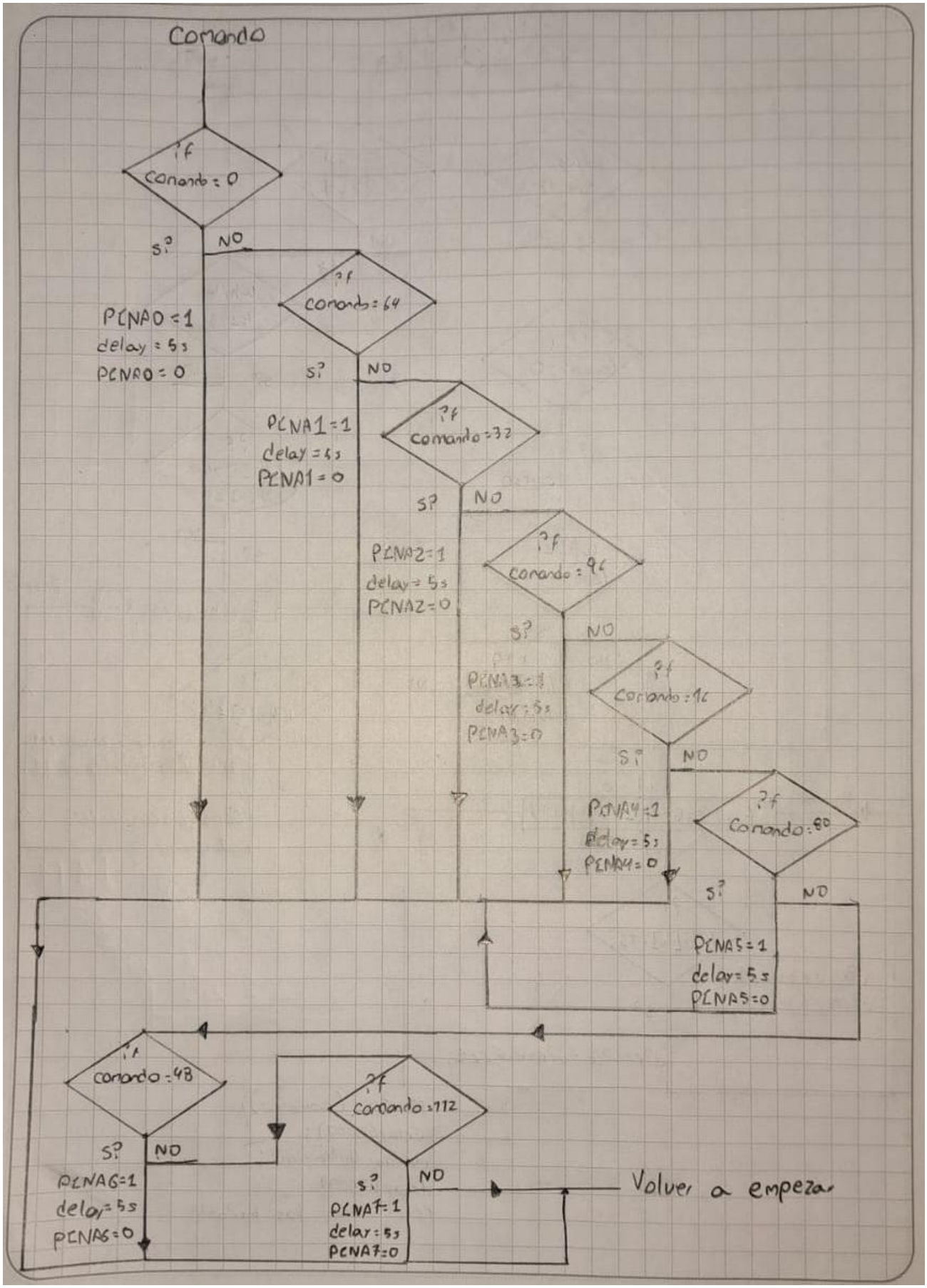

Figura 5. Encendido leds de acuerdo con los botones seleccionados.

EQUEMATICO

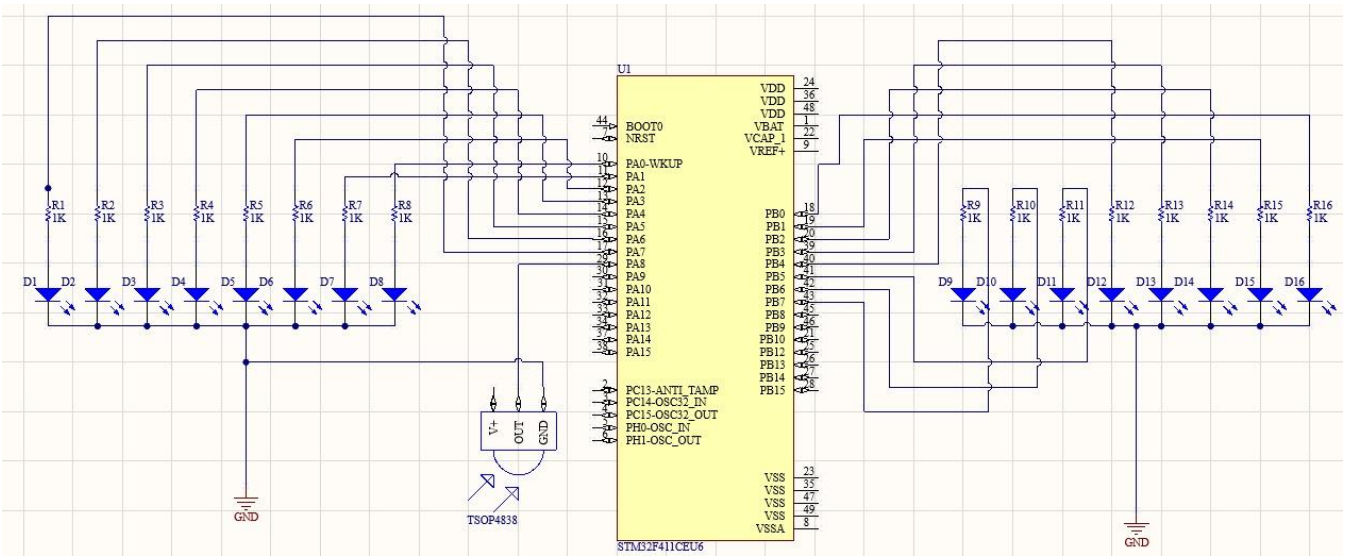

Figura 6. Esquemático del circuito implementado.

CONCLUSIONES

- La capacidad del microcontrolador STM32F411CEU para capturar y decodificar señales IR dependió de la precisión en la configuración del TIMER y de la capacidad del receptor IR para captar señales confiablemente. La eficiencia del protocolo IR utilizado se verificó mediante la correcta visualización de los comandos en los Leds.
- El receptor IR TSOP4838 demostró ser efectivo en la captura de señales del control remoto. Sin embargo, la correcta interpretación de las señales IR requirió un análisis cuidadoso de la señal y un ajuste fino en la configuración del TIMER del microcontrolador para una correcta interpretación.
- La configuración del TIMER fue crucial para medir con precisión los tiempos de bit y los comandos IR. Se observó que un ajuste incorrecto en el TIMER podía llevar a errores en la decodificación de los datos, afectando la visualización en los Leds en cuanto a la visualización del comando y dirección del botón presionado.
- La implementación de la visualización en Leds permitió verificar los datos recibidos del control remoto. La representación en binario y la asignación de teclas a Leds específicos permitieron una verificación clara del funcionamiento del sistema al momento de guardar dicho comando y dirección.
- Un desafío importante fue la sincronización precisa entre el receptor IR y el microcontrolador. La solución involucró ajustes en la configuración del TIMER y la calibración del receptor IR dando como resultado su correcto funcionamiento.