Laboratorio # 3 Comunicación Serial – Protocolos IR

D. Jiménez, S. Herran.

Universidad Sergio Arboleda
Escuela de ciencias exactas e ingeniería
Sistemas Embebidos
27 de Agosto del 2024

1. Problema

1. Realizar un montaje (circuito) utilizando los materiales antes mencionados y realizando el procedimiento sugerido que tenga las siguientes funciones:

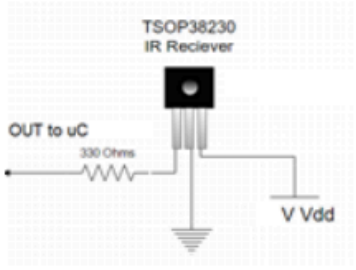
   a. Capturar las tramas enviadas por el control remoto (todas las teclas) y mostrar el código de la funcionalidad correspondiente (vol+ ch+ 1,2, etc.) en los LEDs.
   b. Capturar las tramas enviadas por el control remoto (todas) y mostrar el código capturado en binario en los LEDs (comando y dirección) correspondientes.
   c. Escoger 8 botones y asignar una tecla del control a cada uno de los LEDs para que cuando sea oprimida el LED asignado cambie de estado.
   d. Entregar un informe con toda la **documentación del protocolo analizado**, diagramas de flujo de cómo funciona el programa y 5 conclusiones de haber desarrollado el laboratorio.

2. Materiales

- ☐ **Microcontrolador STM32F411**
- ☐ **LEDs (8 mínimo)**
- ☐ **Componentes pasivos (resistores, condensadores, etc)**
- ☐ **Control remoto IR de televisor LG**
- ☐ **Receptor IR TSOP38230**

3. Resolución del problema

- Conectar el receptor IR revisando la hoja de datos del módulo que cada grupo tenga para realizar la conexión correcta, si no se tiene, realizar las pruebas utilizando en el VCC una resistencia de 330 Ohms.



- Conectar la terminal OUT en una entrada del microcontrolador.
- Conectar el Osciloscopio en la terminal OUT para visualizar las tramas y el protocolo del control remoto utilizado y así poder calcular tiempos de bit, etc.
- Realizar varias capturas de las tramas enviadas por el control remoto operando varias veces la misma tecla y probar con el osciloscopio como con el TIMER para determinar la estructura del protocolo.
- Configurar el TIMER/COUNTER del microcontrolador para contar los tiempos de bit y capturar los comandos enviados por el control remoto.
- Conectar los LEDs en fila para visualizar el valor correspondiente, si el valor no alcanza en 8, debe ser mostrado en varios tiempos cada uno de los bytes.

La configuración anterior para conectar el receptor, es la misma que la del receptor utilizada para este laboratorio, se utilizó el control remoto de TV LG, de modo que este usa el protocolo NEC, que será mostrado en la siguiente imagen:

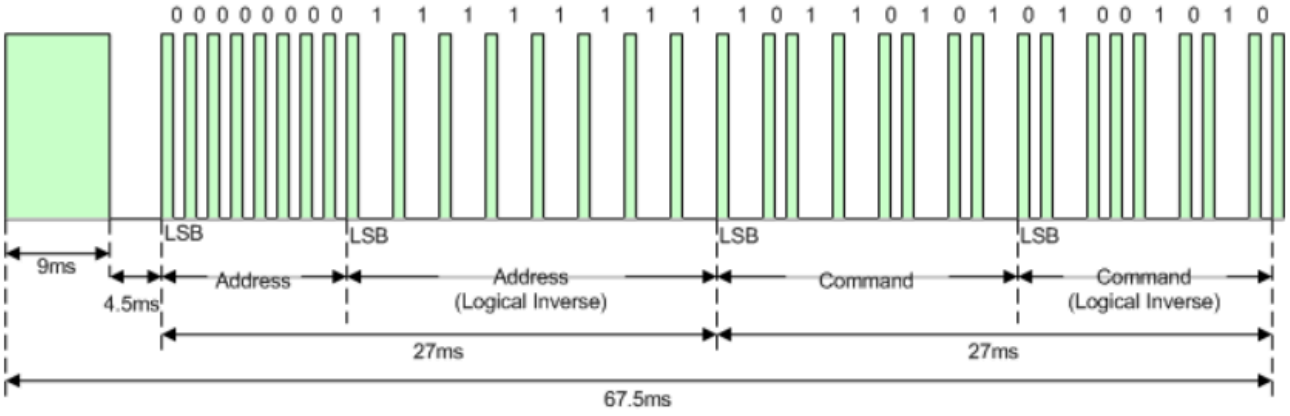


Figure 1. Example message frame using the NEC IR transmission protocol.

En el siguiente link se puede ver la información del protocolo más detallada:
https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol

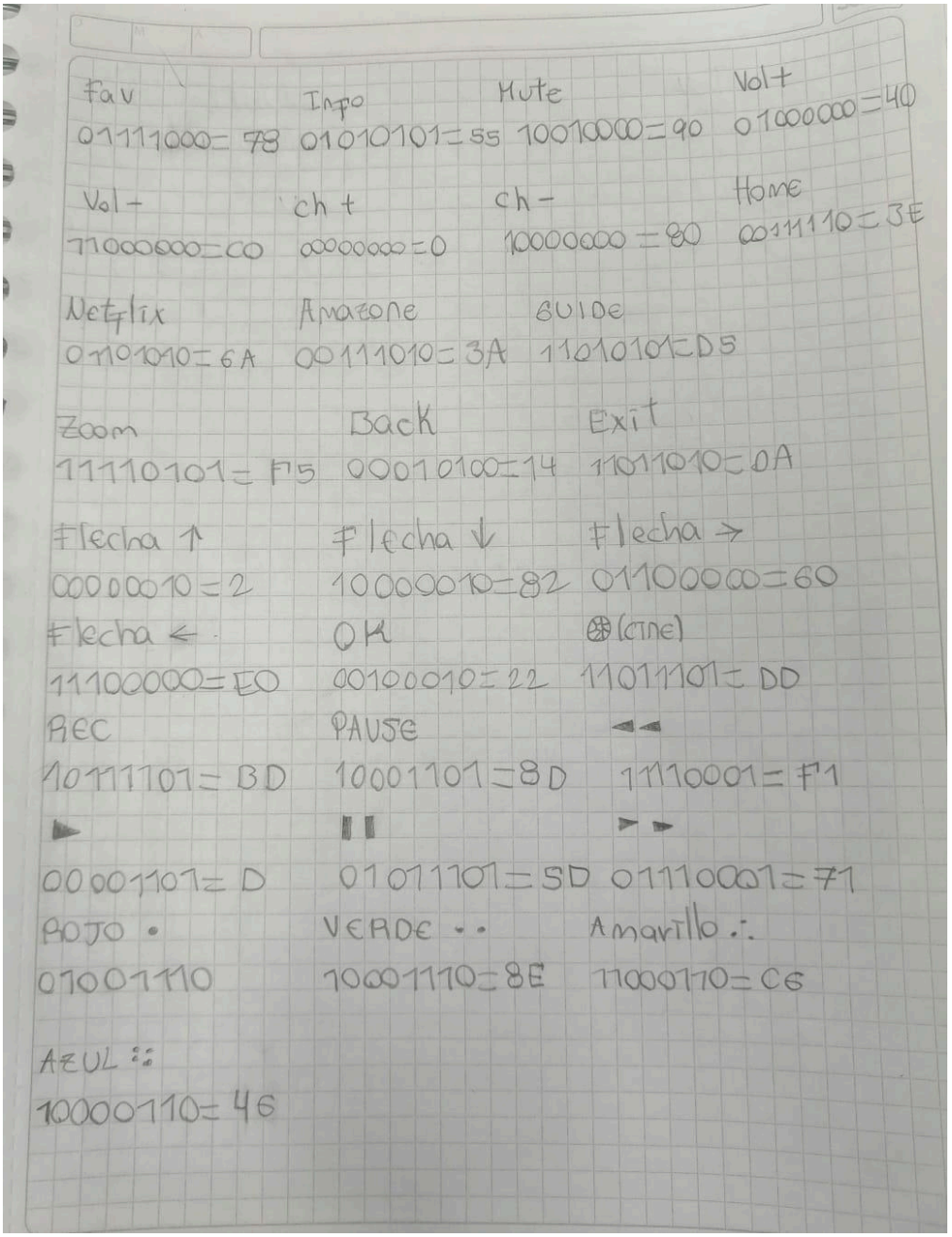
Resumen explicación 1 y 0 del protocolo:

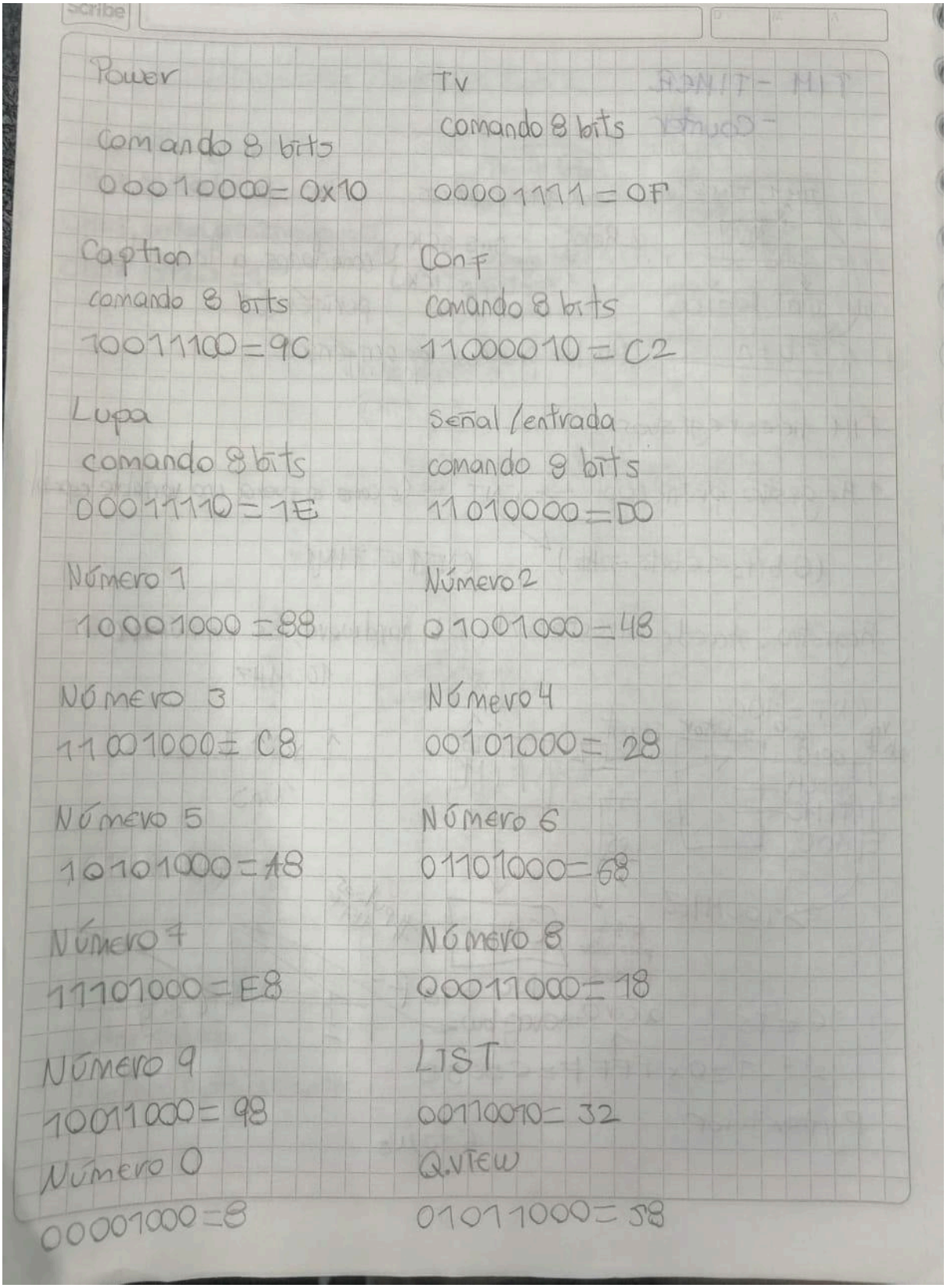
- Logical '0' – a 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms

- Logical '1' – a 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms

De esta forma se puede diferenciar el pulso con valor 0 en binario y el pulso con valor 1 en binario. Otro dato importante es que el tiempo en que comienzan los pulsos del comando es de 40.5ms aproximadamente, por lo que desde ahí se pueden contar 8 pulsos y obtener el comando completo para su correcta abstracción y visualización en los leds.

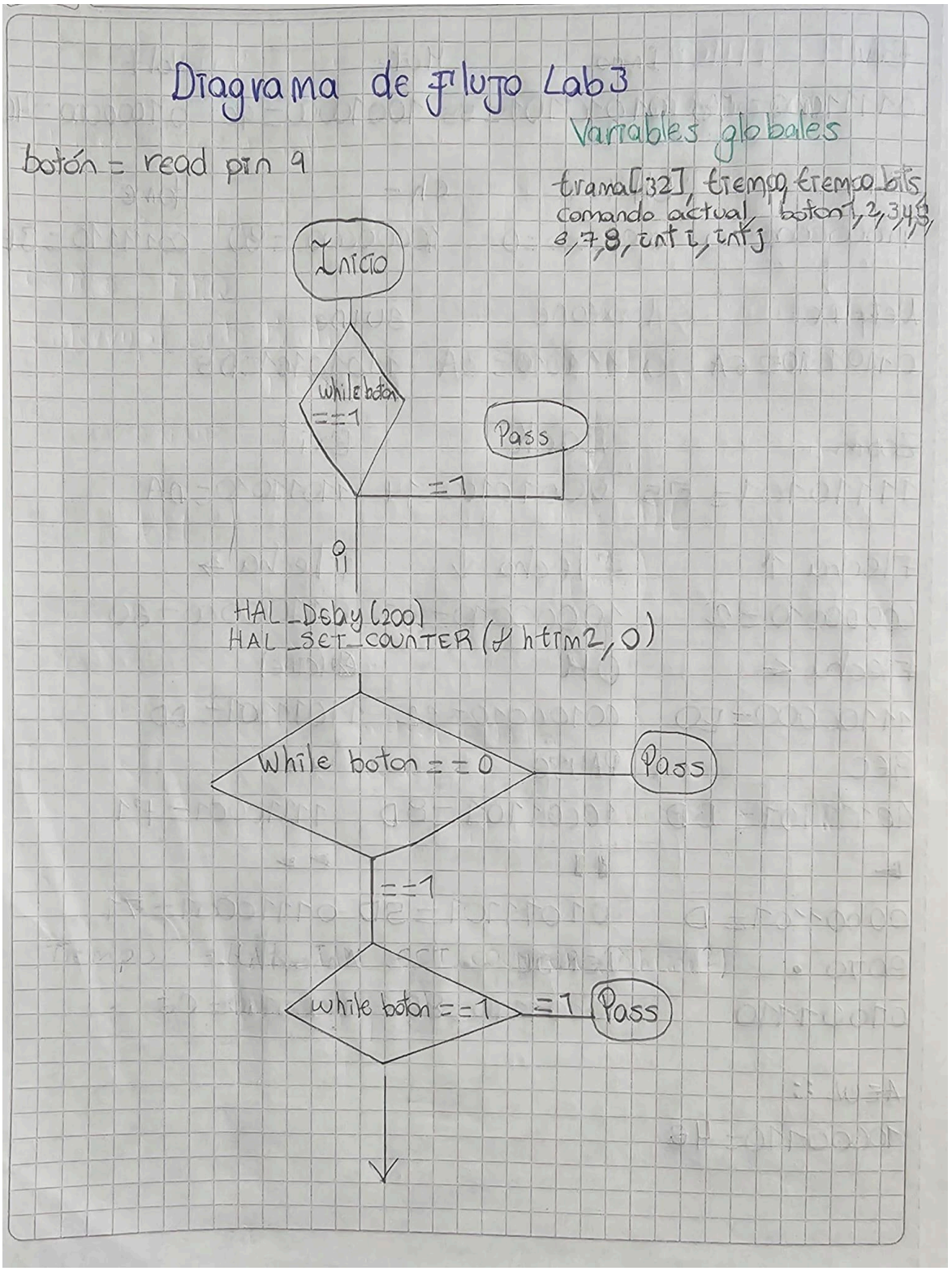
Todos los comandos de cada botón fueron evidenciados en el osciloscopio y recolectados como número binario convertido a hexadecimal por conveniencia, a continuación se muestran los datos recolectados de cada botón para el control LG elegido:

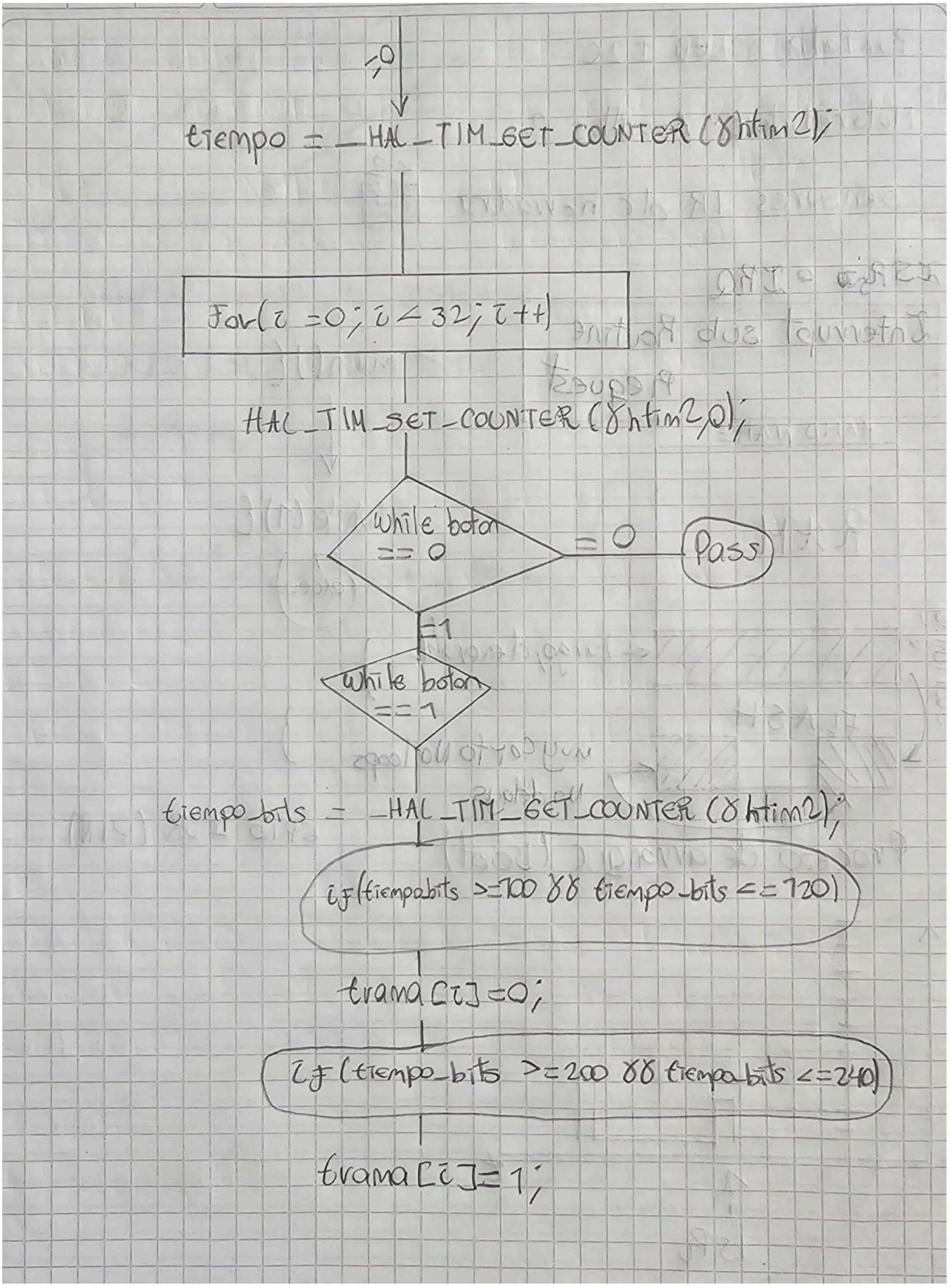




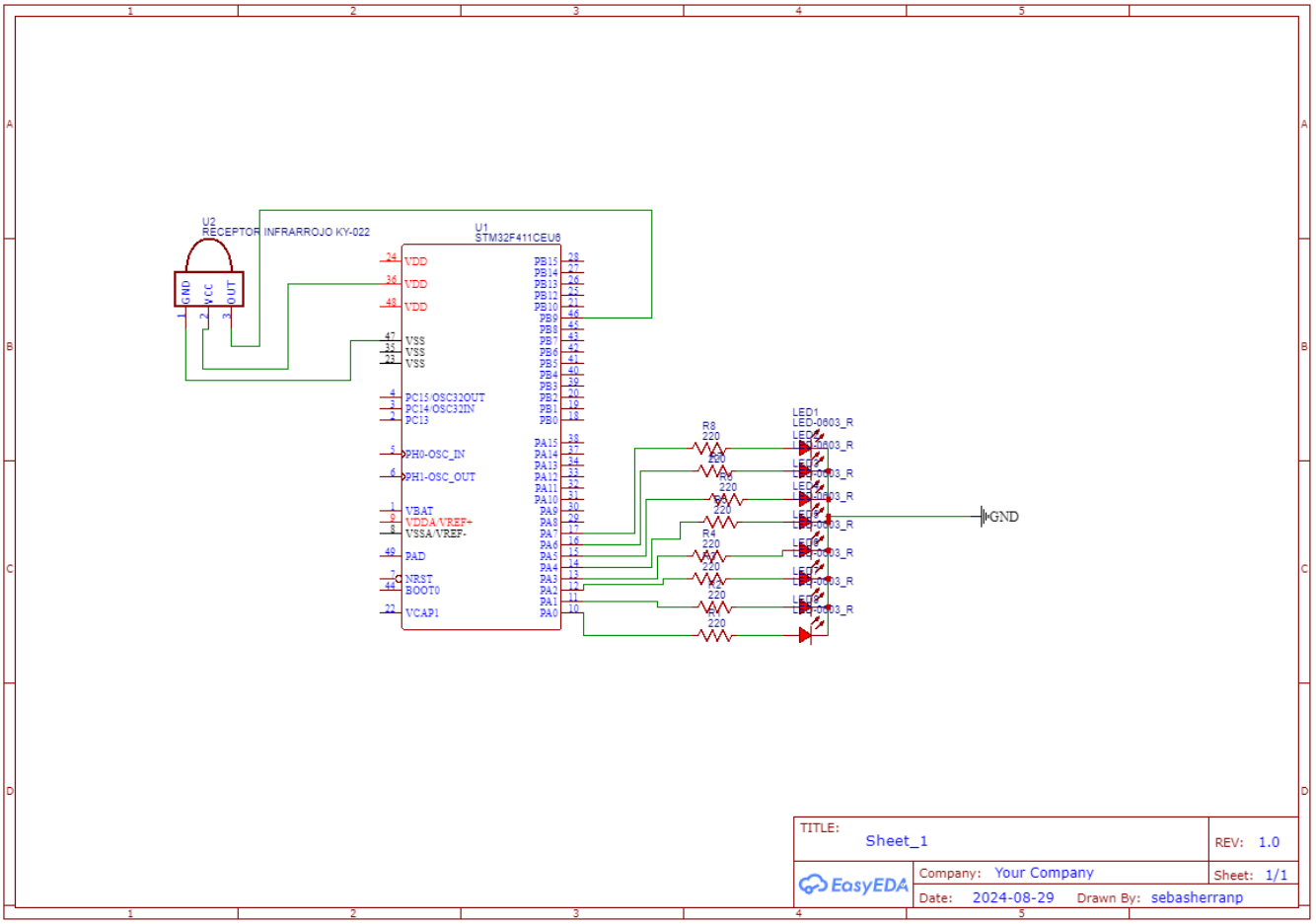
Teniendo así 45 botones capturados por los comandos visualizados en el osciloscopio.

4. Diagrama de flujo




i=0
tiempo = __HAL_TIM_GET_COUNTER(&htim2);
for(i =0; i<32; i++)
HAL_TIM_SET_COUNTER(&htim2,0);
while boton == 0
=0
Pass
=1
while boton == 1
tiempo_bits = _HAL_TIM_GET_COUNTER(&htim2);
if(tiempobits >=100 && tiempo_bits <=120)
trama[i]=0;
if(tiempo_bits >=200 && tiempo_bits <=240)
trama[i]=1;

5. Esquemático



6. Problemáticas en el desarrollo

- ➔ Al principio se tuvo bastante problema con la conexión y prueba del receptor, puesto que funcionó en un instante, pero luego se dañó. Al final se descubrió que era la protoboard, la cual tiene bastantes conexiones dañadas y por lo tanto se cambió y se logró su óptimo funcionamiento.

- ➔ En la fase de visualización de los pulsos en el osciloscopio, se tuvieron varios inconvenientes, al parecer el receptor no captaba tan bien algunos botones, se intuye que puede ser porque los botones no mandan la información tan efectivamente como otros que si se lograba obtener el tren de pulsos al instante.

7. Código principal

```
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
```

```c
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
uint16_t trama[32];
uint16_t tiempo;
uint16_t tiempo_bits;
uint16_t comando_actual = 0;
// Declarar el comando de los botones del 1 al 8
uint16_t boton_1 = 0x88;
uint16_t boton_2 = 0x48;
uint16_t boton_3 = 0xC8;
uint16_t boton_4 = 0x28;
uint16_t boton_5 = 0xA8;
uint16_t boton_6 = 0x68;
uint16_t boton_7 = 0xE8;
uint16_t boton_8 = 0x18;
int i;
int j;
void verificar_y_cambiar_estado(uint16_t valor_trama) {
  // Apagar todos los LEDs
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
  // Encender el LED correspondiente
  switch (valor_trama) {
    case 0x88:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET); // Botón 1 - Pin A1
      break;
    case 0x48:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET); // Botón 2 - Pin A2
      break;
    case 0xC8:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET); // Botón 3 - Pin A3
      break;
```

```
    case 0x28:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET); // Botón 4 - Pin A4
      break;
    case 0xA8:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET); // Botón 5 - Pin A5
      break;
    case 0x68:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET); // Botón 6 - Pin A6
      break;
    case 0xE8:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET); // Botón 7 - Pin A7
      break;
    case 0x18:
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET); // Botón 8 - Pin A8
      break;
    default:
      // No coincide con ningún botón
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, trama[17] ? GPIO_PIN_SET : GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, trama[18] ? GPIO_PIN_SET : GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, trama[19] ? GPIO_PIN_SET : GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, trama[20] ? GPIO_PIN_SET : GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, trama[21] ? GPIO_PIN_SET : GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, trama[22] ? GPIO_PIN_SET : GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, trama[23] ? GPIO_PIN_SET : GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, trama[24] ? GPIO_PIN_SET : GPIO_PIN_RESET);
      break;
  }
}
/* USER CODE END 0 */
/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void)
{
 /* USER CODE BEGIN 1 */
 /* USER CODE END 1 */
 /* MCU Configuration--------------------------------------------------------*/
 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
 HAL_Init();
 /* USER CODE BEGIN Init */
 /* USER CODE END Init */
 /* Configure the system clock */
 SystemClock_Config();
 /* USER CODE BEGIN SysInit */
 /* USER CODE END SysInit */
 /* Initialize all configured peripherals */
 MX_GPIO_Init();
 MX_TIM2_Init();
 /* USER CODE BEGIN 2 */
 /* USER CODE END 2 */
 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 HAL_TIM_Base_Start(&htim2);
 while (1)
 {
```

```
		/* Empieza el código, se hace un bucle que espere cuando el PIN 9 del  puerto B cambie de estado*/
		while(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9)== 1);
		/*Se agrega un delay para el rebote y capturar los bits correctamente*/
		HAL_Delay(200);
		/* Se pone el contador del TIM2 en 0 para que empiece a contar el tiempo  */
		__HAL_TIM_SET_COUNTER(&htim2, 0);
		/* Espera a que el PIN 9 deje de estar en 0 y luego cuando vuelva a 1 espere de nuevo*/
		while(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9)== 0);
		while(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9)== 1);
		/* Se crea una variable tiempo, esta obtiene el tiempo del TIM2*/
		tiempo = __HAL_TIM_GET_COUNTER(&htim2);
		/* Se crea un bucle for que inicia en 0 y acaba a las 32 veces de recorrido, va sumando 1 cada vez
		 * que acaba el bucle*/
			for(i=0 ;i<32 ;i++)
			{
				/*Se pone el TIM2 en 0 para volver a contar*/
				__HAL_TIM_SET_COUNTER(&htim2, 0);
				/*Se hace un bucle que mientras el pin este en 0 espere y luego que cuando este en 1
espere*/
				while(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9)== 0);
				while(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9)== 1);
				/*Se crea una variable tiempo_bits, que obtiene el valor del contador TIM2 */
				tiempo_bits = __HAL_TIM_GET_COUNTER(&htim2);
				/* Se crea una condición, si el tiempo_bits esta entre 100 y 120 el arreglo trama en el
indice
				 * correspondiente sea igual a 0, esto se hace para que guarde el valor lógico de
acuerdo con
				 * el protocolo NIC*/
				/*100 equivale a 1ms y 120 a 1.2ms*/
				if (tiempo_bits >=100 && tiempo_bits<=120 )
				{
					trama[i]=0;
				}
				/*De la misma forma se hace una condición pero esta vez con las restricciónes para
que almacene
				 * un 1 lógico*/
				if (tiempo_bits >=200 && tiempo_bits<=240 )
				{
					trama[i]=1;
				}
			}
			// Convertir los bits de trama[17] a trama[24] en un valor hexadecimal
				comando_actual = 0;
				for (i = 0; i < 8; i++) {
					comando_actual |= (trama[17 + i] << (7 - i)); // Combinando los bits en un
solo valor
				}
				// Verificar y cambiar el estado de los pines según el valor de comando_actual
				verificar_y_cambiar_estado(comando_actual);
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
 }
 /* USER CODE END 3 */
}
```

8. Explicación simple del código

- ❖ Aunque en los comentarios está explicado el código, se hace una breve explicación general del mismo.

- ❖ El array trama que tiene tamaño 32, se utiliza para realizar la trama de los 32 bits, la variable tiempo obtiene el valor de TIM2 que es utilizada para contar el tiempo de los bits. la variable tiempo_bits que se utiliza para comparar si está entre los tiempos estipulados por el protocolo para que sea un 1 o un 0 según corresponda. La variable comando actual que se utiliza para saber cual es el comando y según sea prenda el led 1 para el número 1 y así sucesivamente para los 8 leds.

- ❖ La función verificar y cambiar estado se utiliza para apagar los leds según sea necesario para los comandos.

- ❖ El switch indica que se prenda cada led según se presione de los botones del 1 al 8 del control y si ninguno se está presionando que se puedan ver los comandos de todos los otros botones.

- ❖ La función principal, que ya está más detallada en los comentarios del código, pero que como breve definición mira si está presionando un botón, lee los bits y define si son unos y ceros y los almacena en el array trama.

7. Conclusiones

- ★ Se entendió el protocolo NEC, utilizado para captar el comando y detectar cuál botón se pulsa con efectividad, comprobando el comando normal con el comando inverso y utilizar estos comandos para el programa.

- ★ Se logró contar los pulsos del protocolo con el TIM, utilizando el prescaler a 1000-1 para medir el tiempo en 10µs POR CNT, teniendo una configuración de 100MHz.

- ★ Se utilizaron las librerías HAL_GPIO y la librería HAL_TIM para el COUNTER para medir tiempos y completar el programa en STM32CUBEIDE.

- ★ Es de vital importancia tener en cuenta el rebote que puede causar al pulsar un botón, pues sin este delay después de presionar 1 vez cualquier botón puede generar lecturas erróneas. Inclusive no se necesitó ningún bucle o algo parecido para capturar la señal completa después de la primera vez, simplemente con el delay ya se omite eso y funciona para cualquier botón.

★ Se logró cumplir con todas las partes del laboratorio, sus funcionalidades y todo los requisitos que se querían. También se comprende cada vez más cómo utilizar el software, haciendo debug, break points, viendo cual valor tiene cada variable etc. La parte más complicada del laboratorio fue al principio de programar, pues se tuvieron diferencias de cómo se hacía el programa y también la lógica de que métodos utilizar para leer los bits. De igual forma no se tenía claro cómo funcionaba el prescaler, pero luego de una explicación se logró entender y se tuvo una visión más clara para poder continuar haciendo el programa.