

Carrera: Ingeniería Electrónica

Asignatura: Técnicas Digitales II

Año: 2024

Grupo: N°3

Alumnos: Décima Enrique Emanuel,

Castro Oscar Martín,

Ortiz Nicolás Agustín.

Profesor: Ing. Rubén Darío Mansilla

Informe de proyecto Final

Título del Proyecto Final: RADAR ULTRASÓNICO.

1.- Consideraciones sobre el hardware del proyecto

1.1 Descripción del proyecto

El proyecto consiste en el desarrollo de un **Radar Ultrasónico**, diseñado para detectar objetos dentro de un rango de distancia determinado y alertar al usuario mediante mensajes a través de una pantalla LCD y un dispositivo conectado vía Bluetooth. Este sistema se basa en la medición de distancias utilizando un **sensor ultrasónico HC-SR04**, montado sobre un **servomotor**, que le permite escanear su entorno en un rango de 180°.

El sistema está implementado sobre una placa de desarrollo **STM32 BlackPill** que posee como núcleo un microcontrolador **STM32F401CCU6**. Esta placa de desarrollo posee los periféricos necesarios para el desarrollo de la aplicación.

Se justifica el uso de esta placa de desarrollo por sobre la STM32NUCLEO-F429ZI debido a los siguientes motivos:

- Es de tamaño mas compacto, esto la hace ideal para montarla directamente en una PCB con conectores soldados.
- Menor costo
- Menor consumo
- La potencia extra que posee la NUCLEO-F429ZI no es aprovechada por esta aplicación ya que no requiere de procesamiento excesivo.

Objetivos

- **Desarrollar un sistema embebido** capaz de detectar objetos en su entorno mediante un sensor ultrasónico y visualizar la información en tiempo real.
- **Implementar una máquina de estados finitos (MEF)** para gestionar el control del servomotor y la toma de mediciones de distancia.
- **Transmitir alertas a un dispositivo externo** mediante Bluetooth cuando un objeto es detectado dentro de un umbral definido.

- **Optimizar el consumo de energía y los recursos del microcontrolador**, implementando un código eficiente y modular.
- **Desarrollar una interfaz de usuario** sencilla utilizando una pantalla LCD para mostrar la distancia detectada y el ángulo de giro instantáneo del servomotor.

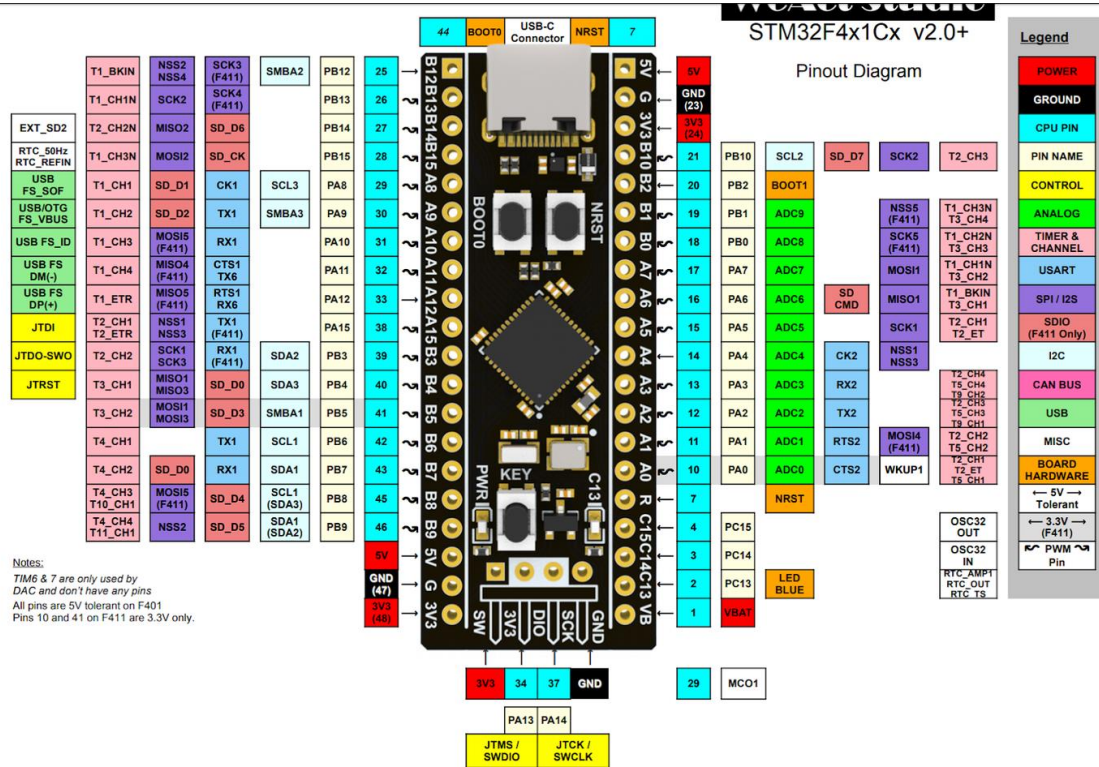
Funcionamiento

El sistema opera mediante una máquina de estados finitos (MEF) que controla el flujo de ejecución. Su funcionamiento se basa en los siguientes pasos:

1. **Inicialización del sistema:** Se configuran los periféricos del microcontrolador, posicionando el servomotor en 0° y estableciendo los valores iniciales de las variables. Se muestra un mensaje en la pantalla LCD y se envía una notificación por Bluetooth indicando que el sistema está activado.
2. **Movimiento del servomotor:** El servomotor gira en pasos de 3° en sentido horario desde 0° hasta 180° y luego en sentido antihorario desde 180° hasta 0°.
3. **Medición de distancia:** En cada posición del servomotor, el sensor ultrasónico HC-SR04 mide la distancia hasta el objeto más cercano.
4. **Verificación del umbral:** Si la distancia medida es menor que el umbral predefinido (7 cm), el sistema cambia al estado de alerta. Si no, la alarma permanece desactivada y el sistema procede a verificar el ángulo del servomotor.
5. **Cambio de dirección:** Si el servomotor alcanza los límites de 0° o 180°, se invierte el sentido del giro. Si el ángulo supera los límites permitidos (entre 0° y 180°), se activa la rutina de manejo de errores.
6. **Generación de alerta:** Si se detecta un objeto dentro del umbral, el sistema activa la alarma de forma intermitente, enciende el LED de alerta y muestra un mensaje en la pantalla LCD indicando "ALARMA ACTIVADA". Además, se envía una notificación por Bluetooth alertando sobre la detección de un objeto.
7. **Repetición del ciclo:** El proceso se ejecuta en bucle, asegurando una vigilancia constante del entorno.
8. **Rutina de manejo de errores:** Si se genera una excepción en el caso de que el programador establezca pasos que permitan llegar a un ángulo superior a 180° o inferior a 0°. Se apaga el LED ON, y se desactiva el sistema, bloqueando su funcionamiento mediante un bucle infinito sin acciones.

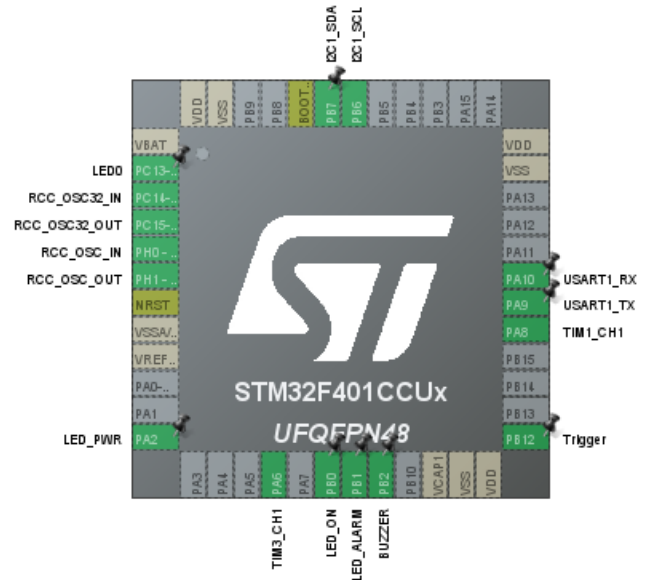
1.2. Circuito del proyecto.

Placa de desarrollo utilizada

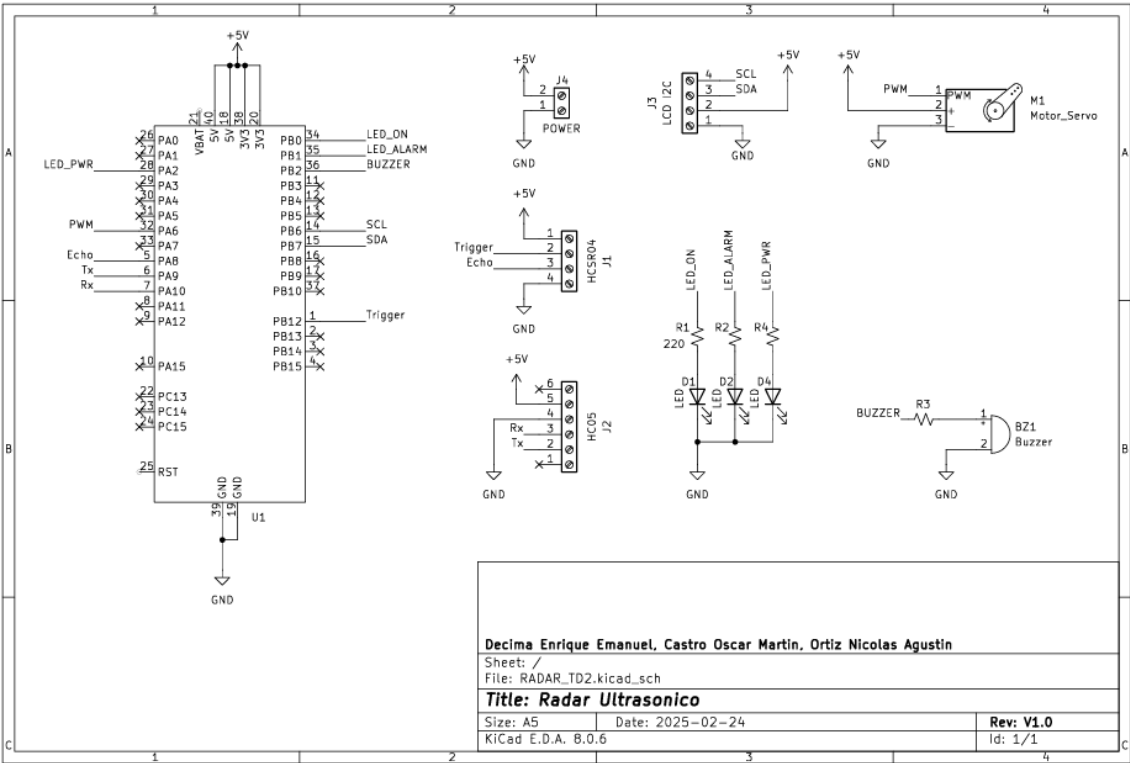


Distribución de Pines (Pinout)

Pin	Etiqueta	Descripción
PC13	LED0	LED integrado a la placa
PA2	LED_PWR	LED Indicador de Alimentación
PB0	LED_ON	Indicador de Sistema Activado
PB1	LED_ALARM	LED indicador de alarma
PB2	BUZZER	Buzzer de Alarma
PA6	TIM3_CH1	Servomotor PWM
PB12	Trigger	HCSR04 Trigger Pin
PA8	TIM1_CH1	HCSR04 Echo Pin
PA9	USART_TX	HC-05 Rx
PA10	USART_RX	HC-05 Tx
PB6	I2C_SCL	PCF8574 para Comunicación con LCD
PB7	I2C_SDA	PCF8574 para Comunicación con LCD

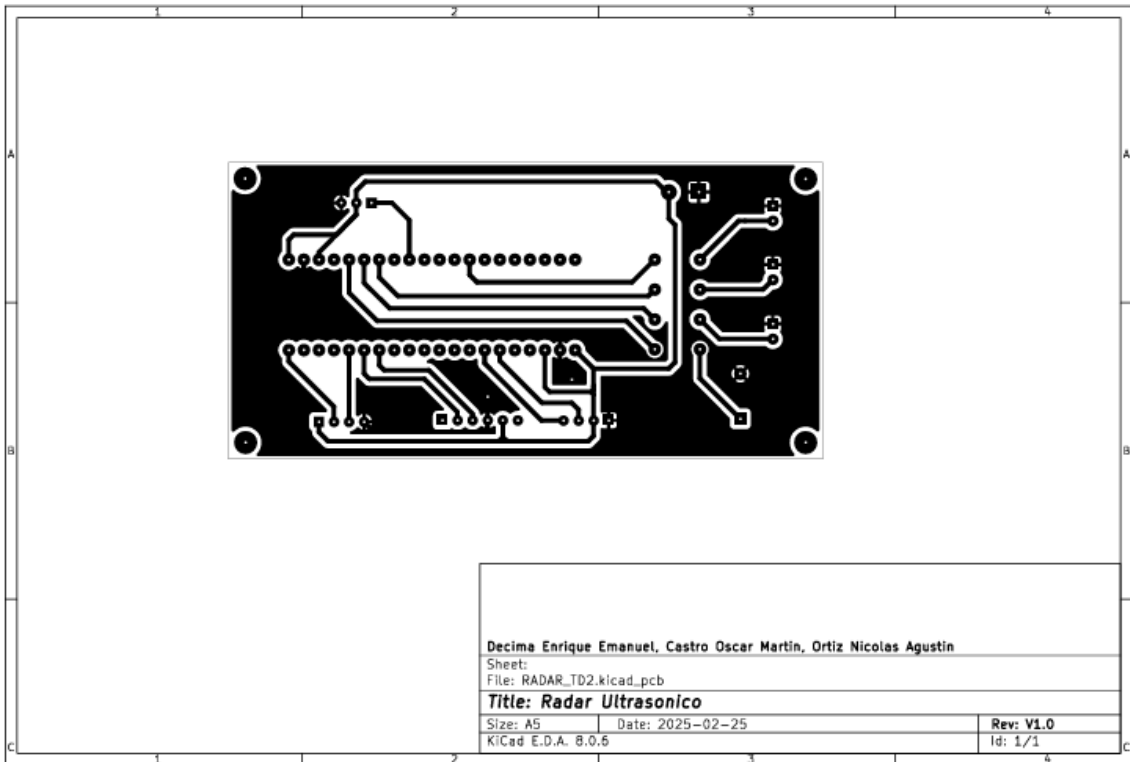


Esquemático

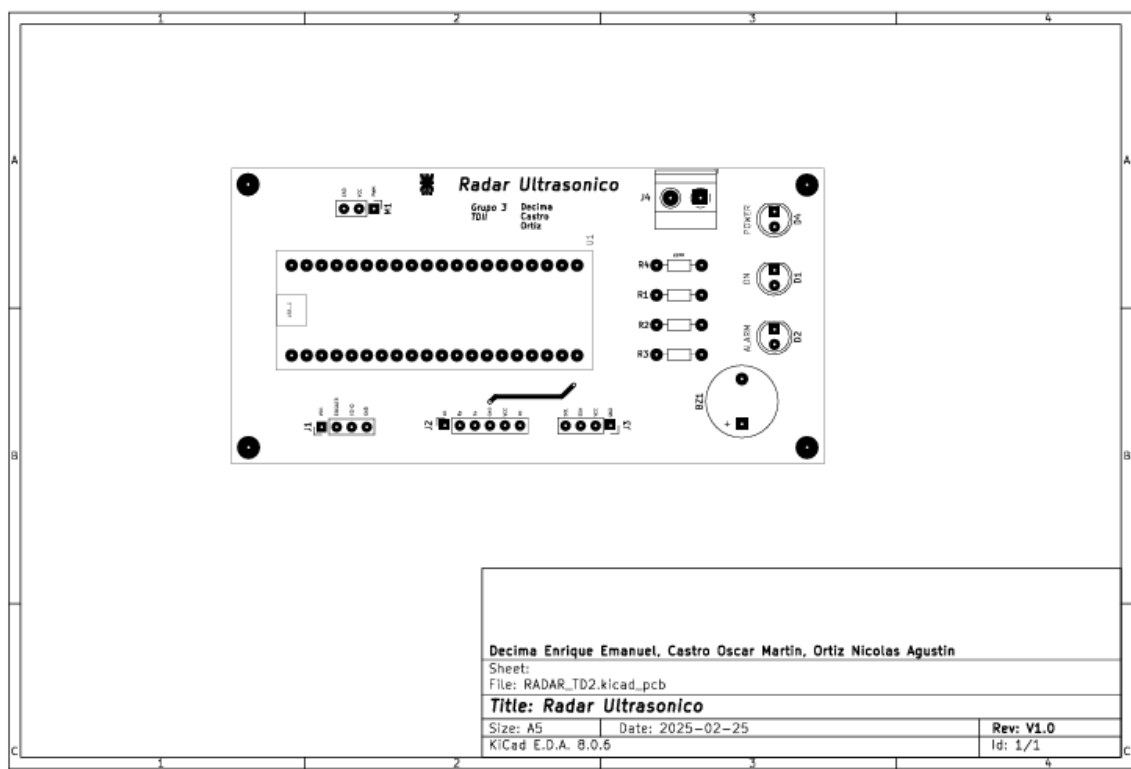


Placa de Circuito Impreso (PCB)

Vista Posterior

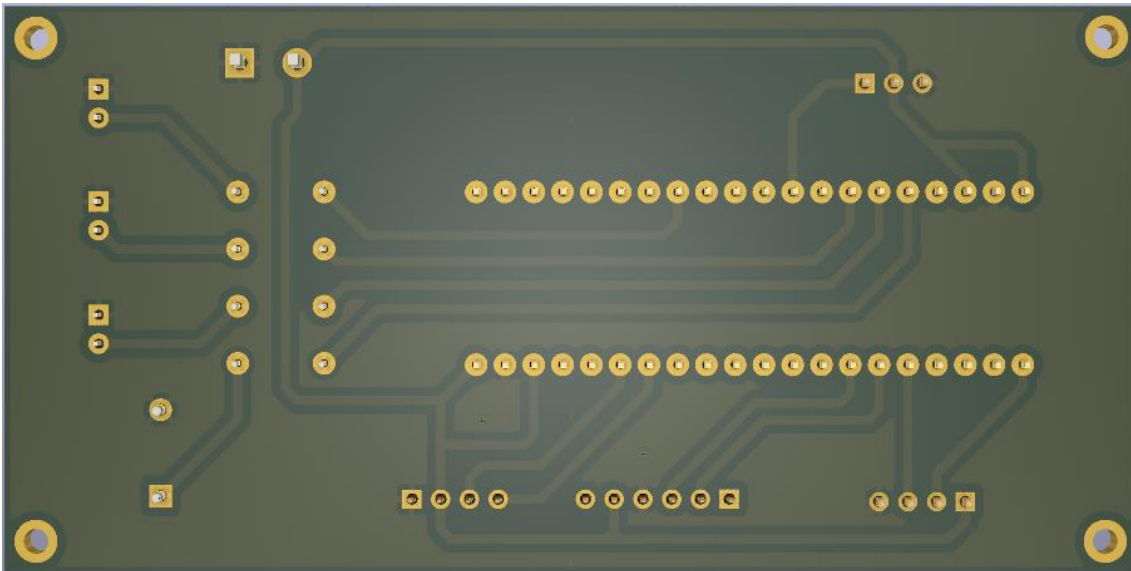


Vista Frontal

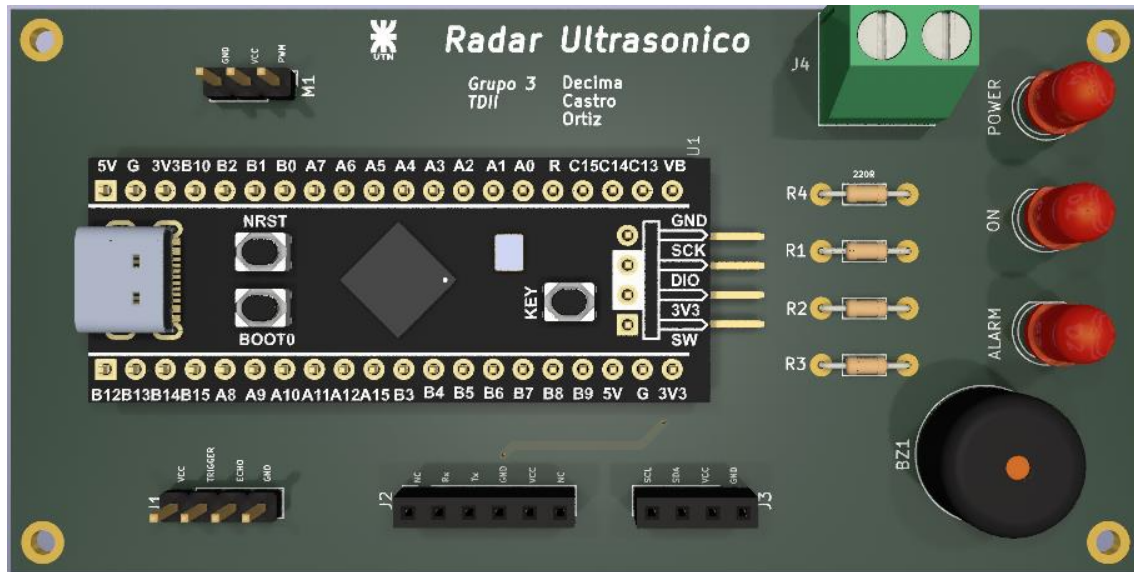


Modelo 3D






Vista Posterior

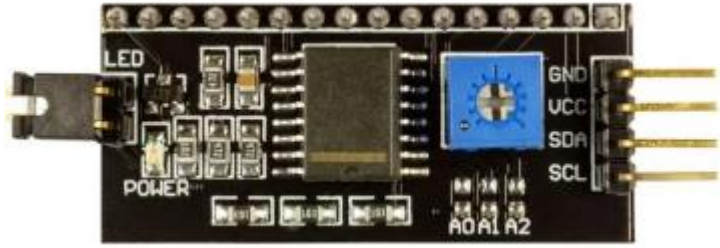


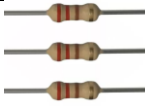
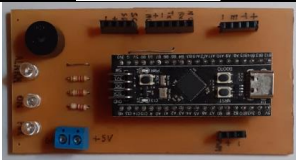




Vista Frontal



1.3. Listado de componentes.

Componente	Imagen	Cantidad
Placa de desarrollo STM32F401CCU6		1
Sensor de Proximidad HC-SR04		1
Servomotor (SG90)		1
Modulo Bluetooth HC-05		1
Pantalla LCD 16x2		1

Modulo I2C PCF8574		1
Buzzer		1
LEDs		3
Resistencias 220Ω		3
Placa PCB Personalizada		1
Cables Dupont		11
Bornera de 2 Vias		1

2.- Consideraciones sobre el software

2.1. Link al repositorio con el código fuente del proyecto.

https://github.com/EmanuelDecima/Grupo_3_TDII_2024/tree/main/AFP_5_TDII_2024_%20STM32F401

Nombre del Proyecto: AFP_5_TDII_2024 _STM32F401

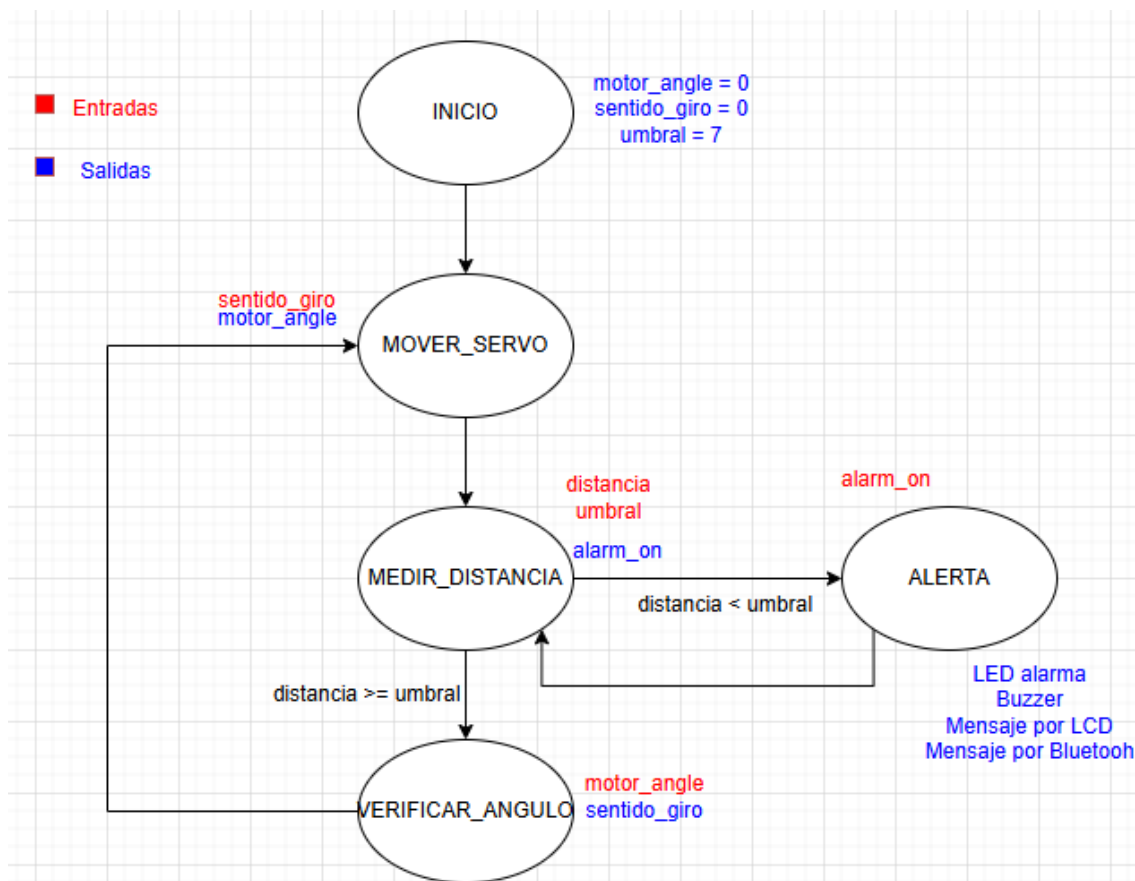
2.2. Descripción de funcionamiento de la aplicación del proyecto.

La aplicación se basa en una placa de desarrollo basada en un STM32F401CCU6 y utiliza un sensor ultrasónico HC-SR04 para medir la distancia a un objeto en centímetros. Tanto el ángulo instantáneo del servomotor como la medición del módulo de proximidad se muestra en una pantalla LCD conectada a la placa mediante un módulo adaptador I2C-LCD, el cual utiliza el driver PCF8574. Además, cuando se detecta un objeto a una distancia menor al umbral (7cm) se muestra un mensaje por pantalla, se emite una

alarma sonora mediante el buzzer, se enciende un led indicador y se emplea un módulo Bluetooth HC-05 para enviar alertas a un dispositivo externo.

El programa principal implementa una Máquina de Estados Finitos (MEF) que gestiona la secuencia de funcionamiento del radar ultrasónico. Antes de entrar en el lazo infinito, se inicializan los módulos utilizados. Luego, la MEF se ejecuta en el lazo infinito, alternando entre distintos estados para realizar las mediciones, girar el servomotor y generar las alertas.

El sistema opera en los siguientes estados:



1. INICIO (Inicialización)

- Se configuran los periféricos (GPIO, PWM, I2C, UART).
- Se coloca el servomotor en la posición inicial (0°).
- Se espera 3 segundos para asegurar la correcta inicialización.
- Se muestra el mensaje "Sistema Activado" en la pantalla LCD.
- Se enciende el LED etiquetado como PWR y ON para indicar que el sistema está en correcto funcionamiento.
- Una vez transcurrido el tiempo de inicialización, el sistema avanza al siguiente estado: MOVER_SERVO.

2. MOVER_SERVO (Movimiento del Servomotor)

- El servomotor gira un paso (3°) en la dirección definida.
- Se verifica si ha alcanzado 0° o 180°:
 - Si alcanza uno de los extremos, cambia la dirección de giro.
 - Si no, sigue girando en el mismo sentido.
- Después de mover el servomotor, el sistema avanza al estado MEDIR_DISTANCIA.

3. MEDIR_DISTANCIA (Lectura de Datos del HC-SR04)

- Se activa el sensor ultrasónico HC-SR04 para medir la distancia al objeto más cercano.
- Se ejecutan varias mediciones y se calcula la mediana de varios valores para aumentar la precisión
- Se almacena la distancia medida y se compara con un umbral predefinido.
- Si la distancia es menor al umbral, se activa el estado de ALERTA.
- Si la distancia es mayor al umbral, el sistema pasa al estado VERIFICAR_ANGULO.

4. VERIFICAR_ANGULO

- Se analiza el ángulo en que se encuentra el servomotor
 - Si se encuentra en 180° invierte el sentido de giro a antihorario
 - Si se encuentra en 0° invierte el sentido de giro a horario
 - Si no se encuentra en ninguno de estos valores pasa al estado MEDIR_DISTANCIA para continuar con un ciclo infinito.

5. ALERTA (Generación de Alerta)

- Se muestra el mensaje "Alarma Activada" en la pantalla LCD.
- Se enciende el LED de alerta (rojo).
- Se activa el buzzer para emitir una señal de advertencia.
- Se envía un mensaje por Bluetooth (HC-05) a un dispositivo emparejado.
- Se espera 3 segundos antes de volver al estado MEDIR_DISTANCIA para verificar si el objeto sigue presente.

2.3. Listado de los módulos de software desarrollados en el proyecto.

RADAR_Delay: Contiene las funciones para implementar delays no bloqueantes, en esta aplicación se usa para actualizar la pantalla LCD cada 800ms.

RADAR_HC05: Contiene las funciones para controlar el modulo Bluetooth, cuenta con funciones para enviar texto y datos numericos.

RADAR_HCSR04: Contiene las funciones para obtener los datos de proximidad medidos con el modulo HCSR04, utiliza el TIM1 CH1 en modo Input Capture para calcular el pulso que devuelve el modulo al realizar la medicion. Posee funciones adicionales para filtrar los datos y obtener mediciones estables.

RADAR_LCD: Utilizada para controlar la pantalla LCD mediante el modulo I2C PCF8574. La librería cuenta con funciones para Inicializar el modulo con su combinacion de comandos especifica, y para enviar tanto cadenas de caracteres como datos. Tambien cuenta con las funciones adicionales necesarias para manejar la pantalla LCD como mover el cursos o limpiar la pantalla.

RADAR_MEF: Implementacion de la maquina de estados mencionada anteriormente.

RADAR_SECUENCIAS: Posee dos funciones, la secuencia de inicio y la secuencia que sigue el dispositivo cuando la alarma esta activada.

RADAR_Servo: Contiene la inicializacion del TIM3 CH1 para el uso de señales PWM que permiten controlar el servomotor y otra funcion que realiza el calculo del ancho de pulso necesario para establecer un angulo determinado en el servomotor.

2.4 Utilizo en la carpeta API los siguientes módulos:

Modulo Delay

- **void delayInit(delay_t * delay, tick_t duration):** Inicialización del delay, establece la duración.
- **bool_t delayRead(delay_t* delay):** Revisa si ya pasó el tiempo correspondiente al delay. Retorna false si no ha transcurrido el tiempo y true si ya ha transcurrido.
- **void delayWrite(delay_t* delay, tick_t duration):** Reescribe el tiempo de duración.

Modulo HC05

- **void HC05_SendString(char* message):** Funcion para enviar una cadena de caracteres por Bluetooth.
- **void HC05_SendData(uint8_t data):** Funcion para enviar un dato numerico por Bluetooth

Modulo HCSR04

- **void HCSR04_Init():** Inicialización del módulo Timer1 Canal1 como Input Capture.
- **static void HCSR04_Delay_us(uint16_t us):** Genera un delay en el orden de los microsegundos.
- **static void HCSR04_Read():** Emite un pulso de 10 µs para la posterior lectura del sensor.
- **static void HCSR04_CaptureMeasures():** Captura varias lecturas y las almacena en un arreglo llamado Filter.

- **static void HCSR04_ResetFilter():** Resetea el arreglo Filter para eliminar valores de mediciones anteriores.
- **static void HCSR04_BubbleSort():** Ordena el arreglo Filter de menor a mayor para calcular la mediana de sus valores.
- **uint8_t HCSR04_GetMeasure():** Realiza varias mediciones y calcula la mediana de los valores obtenidos.
- **void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim):** Calcula la distancia instantánea a través del módulo TIMER Input Capture.

Modulo LCD

- **void LCD_Init():** Inicializa el LCD con una secuencia de comandos específica.
- **void LCD_SendCmd(char cmd):** Envía un comando al LCD.
- **void LCD_SendData(char data):** Envía un carácter al LCD.
- **void LCD_Clear(void):** Limpia la pantalla del LCD.
- **void LCD_PutCur(int row, int col):** Posiciona el cursor en una fila y columna específicas.
- **void LCD_SendString(char *str):** Envía una cadena de caracteres al LCD.
- **void LCD_SendNumber(int number):** Convierte un entero a cadena de caracteres y lo envía al LCD.
- **void LCD_ShowData(uint8_t distance, uint8_t angle):** Muestra los datos de distancia y ángulo en la pantalla LCD.

MEF (Maquina de Estados)

- **void MEF_Init():** Inicializa la máquina de estados en el estado INICIO.
- **uint8_t MEF_GetDistance():** Retorna la última distancia medida.
- **uint8_t MEF_GetAngle():** Retorna el ángulo actual del servomotor.
- **bool MEF_GetAlarmState():** Retorna el estado de la alarma (activada o desactivada).
- **void MEF_Update():** Actualiza la máquina de estados finita (MEF) gestionando el movimiento del servomotor, la medición de distancia, la verificación de ángulos y la activación de la alarma.

Secuencias

- **void Secuencia_Inicio():** Ejecuta la secuencia de encendido del dispositivo, activando salidas digitales, mostrando un mensaje en el LCD y enviando una notificación por Bluetooth.
- **void Mensaje_Alerta():** Ejecuta la secuencia de alerta al detectar un objeto cercano, activando LEDs y el buzzer, mostrando un mensaje en el LCD y enviando una notificación por Bluetooth.

Módulo Servo

- **void Servo_Init():** Inicializa el Timer 3 Canal 1 como salida PWM para controlar el servomotor.
- **void Servo_SetAngle(TIM_HandleTypeDef *htim, uint32_t channel, uint8_t angle):** Calcula el duty cycle de la señal PWM y ajusta el ángulo del servomotor.

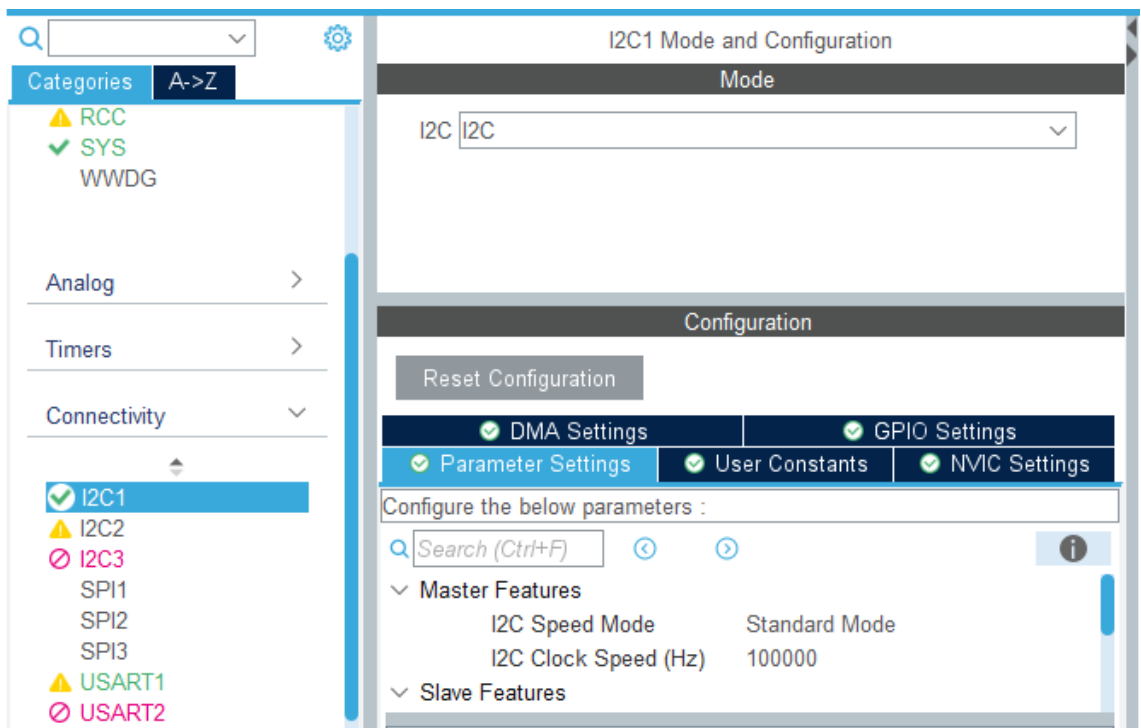
2.5. Listado de los periféricos que utiliza en el proyecto.

Son los siguientes:

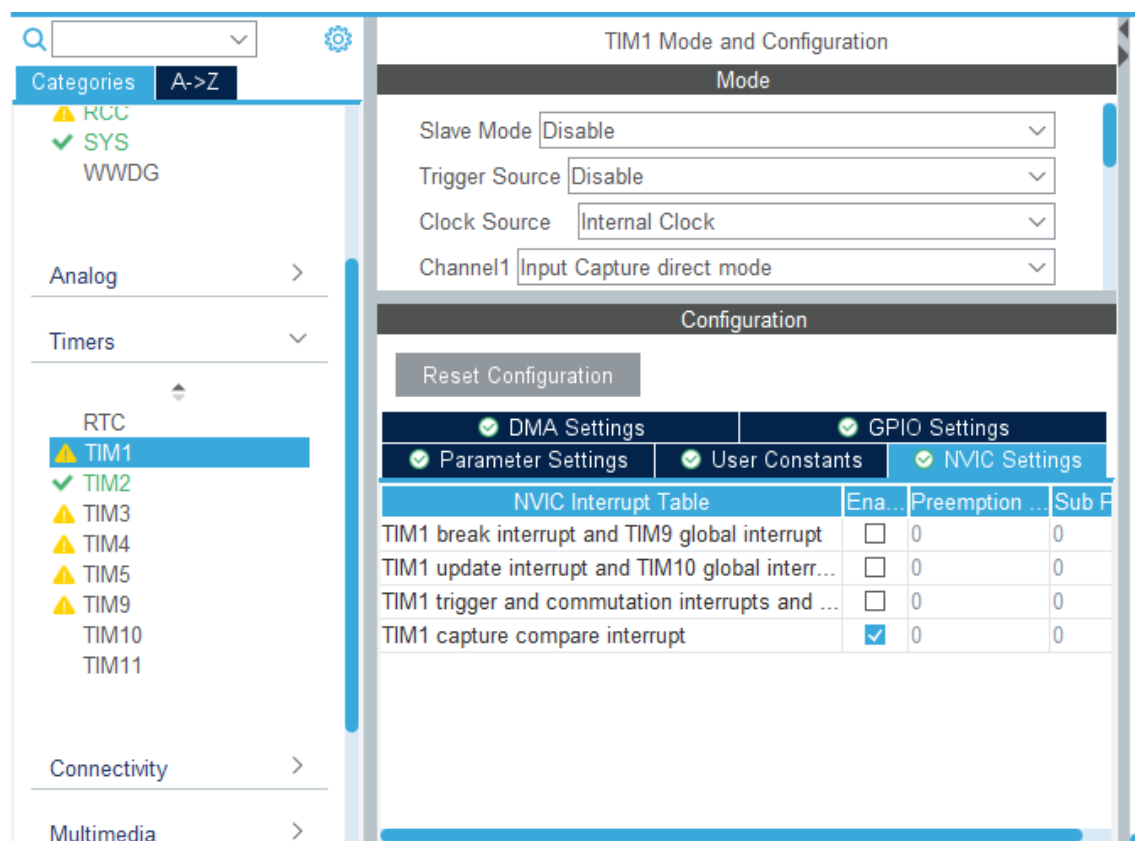
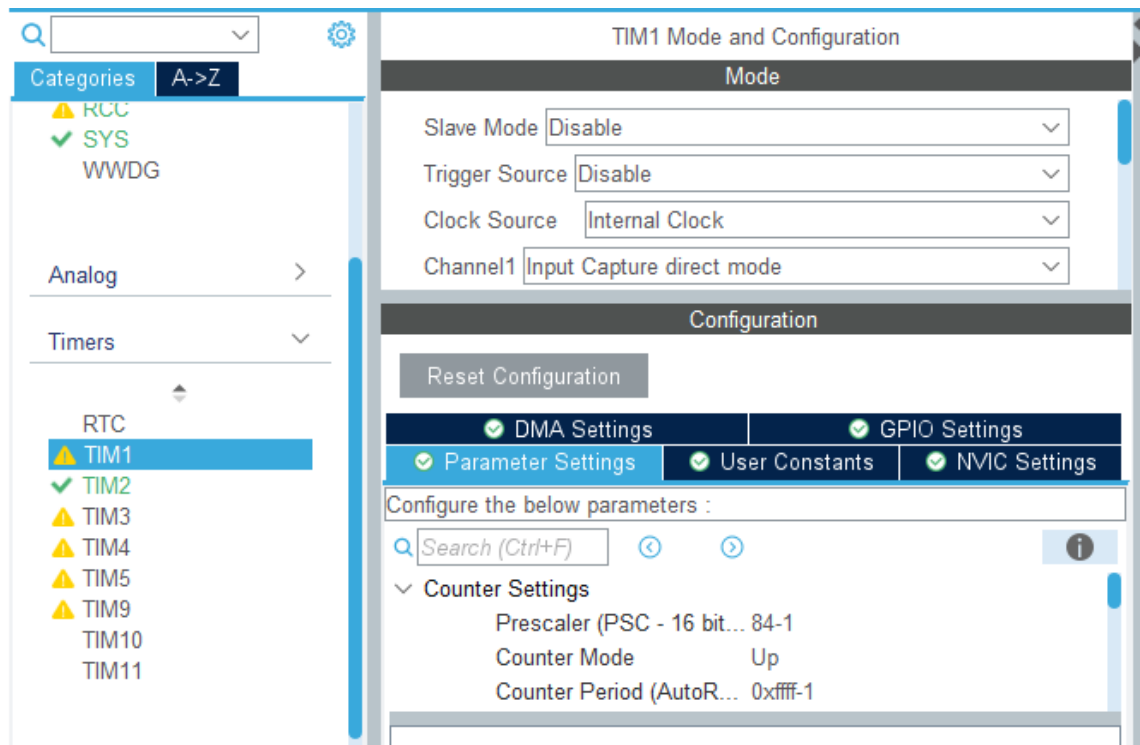
- **GPIO:** Se utiliza para el manejo de los LEDs de alimentación, encendido y alarma, para el buzzer y para la activación del Trigger del sensor ultrasónico HC-SR04. El led integrado a la placa (LED0) se utilizó como ayuda a la hora de depurar la aplicación.



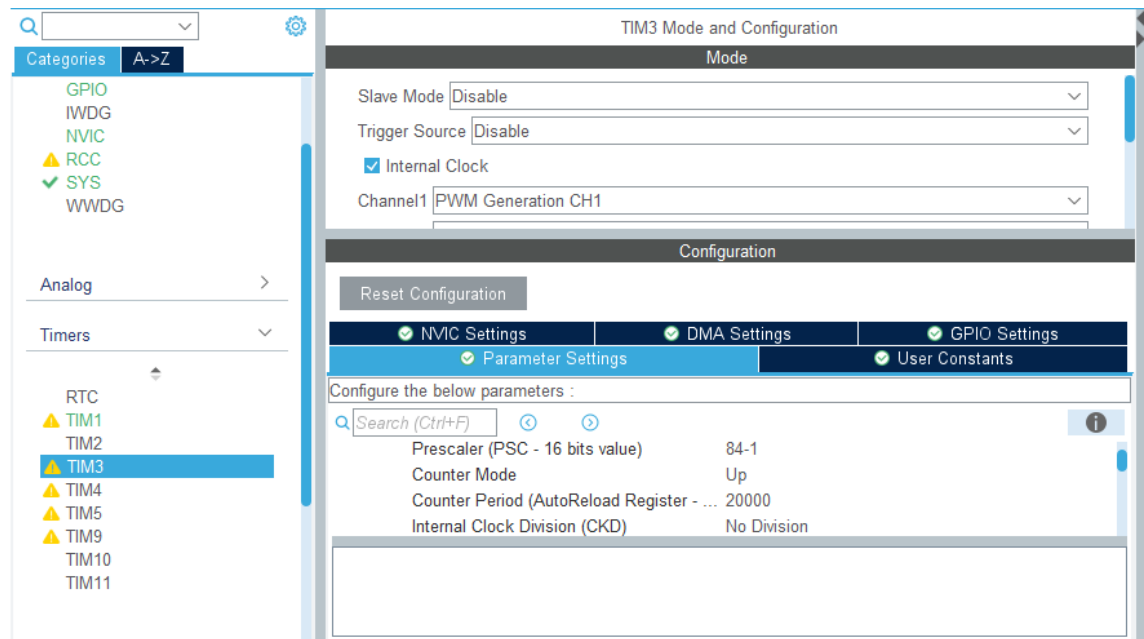
- **I2C1:** Se emplea para la comunicación con la pantalla LCD a través del módulo adaptador PCF8574, permitiendo mostrar información en tiempo real sobre la distancia medida, el ángulo del servomotor y los mensajes de alerta.



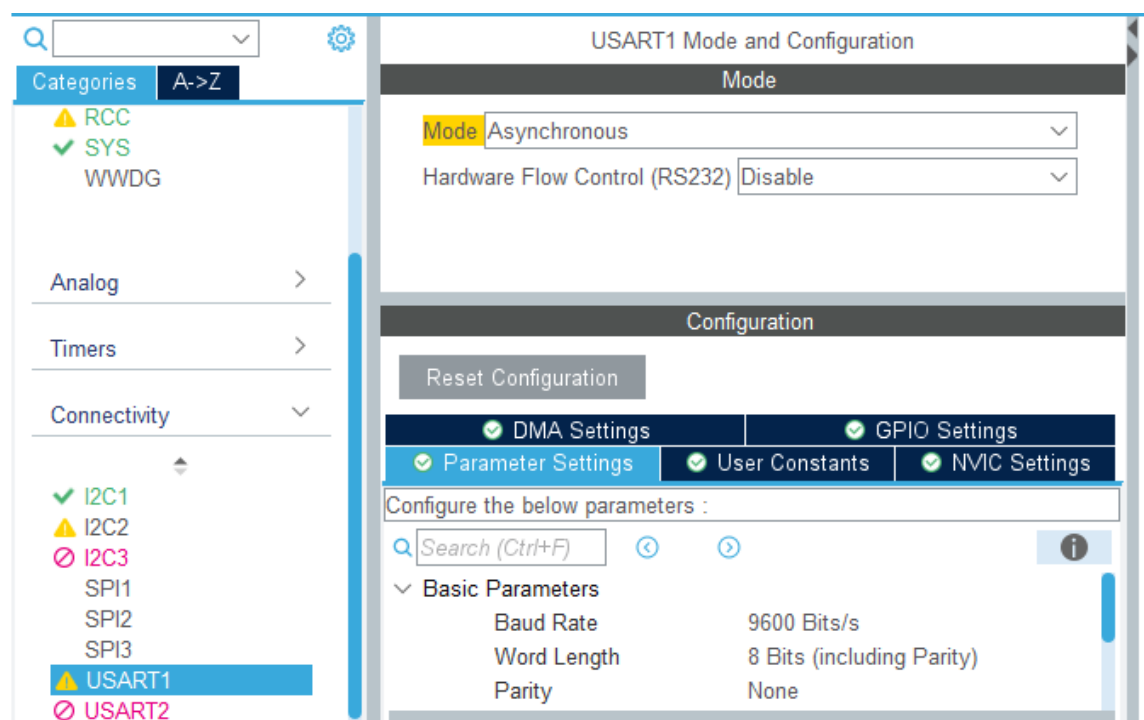
- **TIM1:** Configurado en modo Input Capture para medir el tiempo de respuesta del sensor ultrasónico, permitiendo calcular con precisión la distancia a los objetos detectados. Es necesario activar la interrupción “Capture Compare” para esta aplicación.



- **TIM3:** Configurado en modo PWM para generar la señal de control del servomotor. Se configura como generador de PWM a través del canal 1. Se deben hacer los respectivos cálculos para establecer los valores del “Prescaler” y “Counter Period”.



- **USART1:** Se utiliza para la comunicación con el módulo Bluetooth HC-05, permitiendo enviar alertas a dispositivos móviles o computadoras cuando un objeto es detectado dentro del umbral definido. Se configura en modo asíncrono, con un baud rate de 9600, 8 bits de palabra, sin bit de paridad.



3.- Buenas prácticas de programación

- El código compila sin warnings, No se utilizan recursiones ni control de flujo del tipo goto, los lazos usados tienen límites fijos, no hay lazos que lleven a una recursividad infinita, salvo el lazo infinito de main.c y la rutina de errores (en este caso el lazo infinito es a propósito para bloquear las acciones en la aplicación)
- No hay funciones extensas que superen una página imprimible simple. La función más extensa utilizada (elaborada por terceros) es la siguiente:
https://github.com/EmanuelDecima/Grupo_3_TDII_2024/blob/79e7f182b89b3be12b2bb32ed7c9f10aeff1d4e9/AFP_5_TDII_2024%20STM32F401/Drivers/API/Src/RADAR_HCSR04.c#L121
 Entre las funciones propias implementadas, la más extensa es la siguiente:
https://github.com/EmanuelDecima/Grupo_3_TDII_2024/blob/79e7f182b89b3be12b2bb32ed7c9f10aeff1d4e9/AFP_5_TDII_2024%20STM32F401/Drivers/API/Src/RADAR_SECUENCIAS.c#L40
 ninguna de ellas supera la página simple.
- Se utiliza el preprocesador de forma muy escasa. En su mayoría para incluir librerías necesarias. Poco uso de la etiqueta #define. Aquí se muestra dónde como máximo se utilizan dos constantes:
https://github.com/EmanuelDecima/Grupo_3_TDII_2024/blob/79e7f182b89b3be12b2bb32ed7c9f10aeff1d4e9/AFP_5_TDII_2024%20STM32F401/Drivers/API/Src/RADAR_HCSR04.c#L10
- No se usan punteros a funciones.
- No se hace uso de la asignación dinámica de memoria.

4.- Anexos

Hoja de datos

Microcontrolador:

- **STM32F401CCU6:** [Arm® Cortex®-M4 32b MCU+FPU, 105 DMIPS, 256KB Flash/64KB RAM, 11 TIMs, 1 ADC, 11 comm. interfaces](#)

Placa de Desarrollo:

- **BlackPill:** [ProductOverview_DFRobot-DFRobot-STM32F411-BlackPill-Development-Board.pdf](#)

Sensores:

- **HC-SR04:** [HC-SR04 Datasheet\(PDF\) - List of Unclassified Manufacturers](#)
- **HC-05:** [HC-05 Datasheet\(PDF\) - List of Unclassified Manufacturers](#)

Servomotor:

- **SG90:** [SG90 Datasheet\(PDF\) - List of Unclassified Manufacturers](#)

LCD:

- **LCD 16x02:** [lcd016n002bcfhet.pdf](#)
- **PCF8574:** [PCF8574 Remote 8-Bit I/O Expander for I2 C Bus datasheet \(Rev. K\)](#)

5.- Bibliografía

AdrianMendozaArriagaMX. (21 de Abr de 2024). *Clase I2C LCD STM32 STCUBE Studio*. Obtenido de youtube.com: <https://www.youtube.com/watch?v=W4Uk6aIM-Ao&t=904s>

CircuitGatorHQ. (20 de Ago de 2024). *STM32 Beginners Guide Part3: PWM, TIMERS, Frequency and Duty Cycle. LED Dimming with PWM example*. Obtenido de youtube.com: <https://www.youtube.com/watch?v=zHWvFchXhvw&t=1s>

ControllersTech. (07 de Jul de 2020). *HCSR04 and STM32 using Input Capture || Pulse width || CubeIDE*. Obtenido de youtube.com: https://www.youtube.com/watch?v=ti_1ZwRoIU4&t=372s

Harvey Deitel, P. J. (1995). *Cómo programar en C/C++*. Pearson - Prentice Hall.

Noviello, C. (2022). *Mastering STM32 - A step-by-step guide to the most complete ARM Cortex-M platform, using the official STM32Cube development environment*. Leanpub.