

**UNIVERSIDAD DE BUENOS AIRES
Facultad de Ingeniería
Introducción Sistemas Embebidos**

Memoria del Trabajo Final:

Control de Sistema de Tratamiento de Agua

Autor:

Ing Jesús Enrique García

*Este trabajo fue realizado en las Ciudad Bucaramanga, Colombia,
entre agosto y diciembre de 2024.*

RESUMEN

El trabajo de control para un sistema de tratamiento de agua consiste en el diseño de un controlador que opera sobre dos bombas, de acuerdo con las condiciones determinadas por sensores de nivel y presión en el sistema.

El proyecto está diseñado para implementarse en sistemas de tratamiento de agua básicos, industriales, hospitalarios, entre otros. Su objetivo es garantizar que el agua pase a través de una serie de filtros e incluso membranas, para posteriormente ser almacenada en un tanque.

La implementación se realizó en el entorno de desarrollo Mbed Keil Studio, utilizando los lenguajes C/C++ y una placa NUCLEO-F. Además, se emplearon un módulo de comunicación Bluetooth, un teclado matricial, un display LCD 16x2, un potenciómetro, una serie de pulsadores y LEDs.

Esta memoria describe la planificación, los requisitos y el funcionamiento de los diferentes módulos que conforman el sistema, así como la idea general del funcionamiento integral del sistema desarrollado para este proyecto.



Agradecimientos

Agradezco al docente Ariel Lutenberg por la oportunidad de cursar esta materia y por la paciencia brindada con todo lo relacionado a entregas y revisiones.

Índice General

Introducción general	6
1.1 Objetivo	6
1.1 Resumen general	7
Introducción específica	9
2.1 Requisitos	9
2.2 Casos de uso	10
2.3 Descripción de Módulos	12
2.3.1 Tarjeta NUCLEO	12
2.3.2 Módulo Bluetooth HC05	13
2.3.3 Display LCD	14
2.3.4 Modulo I2C	15
Diseño e implementación	16
3.1 Diagrama de bloques del sistema	16
3.2 Firmware del sistema	19
3.2.1 main	20
3.2.2 control_system	21
3.2.3 control_panel	21
3.2.4 BL	24
3.2.5 buttons	24
3.2.6 level_tank	24
3.2.7 pressure_sensor	25
3.2.8 pumps	25
3.2.9 serial_communication	25
Ensayos y resultados	27
4.1 Pruebas funcionales	27
4.1.1 Comunicación serie	27
4.1.2 Display	29

4.1.2 Comunicación bluetooth	29
4.1.3 Modulo de presiones	31
4.1.4 Sensores de nivel	33
4.2 Pruebas de integración	34
4.3 Cumplimiento de requisitos	35
4.5 Documentación del desarrollo realizado	36
Conclusiones	37
5.1 Resultados obtenidos	37
5.2 Próximos pasos	37
Bibliografía	39

CAPÍTULO 1

Introducción general

1.1 Objetivo

Con el propósito de encontrar una aplicación práctica a los contenidos de la materia Introducción a Sistemas Embebidos, se ha tomado como referencia una problemática laboral en una empresa del área de la salud. Esta empresa utiliza, como parte fundamental de su operación, agua permeada que debe cumplir con los estándares de calidad AMI/ISO.

Actualmente, la empresa cuenta con un controlador genérico que regula todo el sistema de tratamiento. Aunque el sistema funciona correctamente, se ha identificado la necesidad de mejorar la interacción con el usuario e incluir múltiples parámetros que permitan detectar posibles fallos e inconvenientes en diferentes partes del sistema. Un ejemplo concreto de esto es el menú de configuración: actualmente está en inglés, y aunque algunos operadores tienen conocimientos básicos del idioma, resulta fundamental que todos los empleados que interactúan con el sistema puedan acceder a un menú claro, conciso y en su idioma nativo.

Por otra parte, el monitoreo de parámetros como la producción de agua, las distintas presiones del sistema y la calidad del agua es crucial para evitar incidentes relacionados con la calidad del agua tratada. En la situación actual, el seguimiento del sistema se realiza mediante videollamadas o llamadas telefónicas, lo cual resulta ineficiente cuando el personal técnico requiere acceso inmediato a información relevante sobre el funcionamiento del sistema.

Esta problemática evidencia la necesidad de modernizar el sistema mediante un software que permita la monitorización completa y en tiempo real, trasladando estos datos a una plataforma en línea. Esto facilitaría el acceso a la información, permitiendo al personal técnico tomar acciones preventivas y correctivas con mayor rapidez, y evitando paradas de emergencia que podrían impactar negativamente en las operaciones de la empresa.

La modernización propuesta no solo mejorará la eficiencia del sistema, sino que también aportará múltiples beneficios, como la optimización de la toma de decisiones y la posibilidad de interactuar con los datos recolectados para implementar estrategias preventivas de mantenimiento

1.1 Resumen general

El propósito del presente trabajo es desarrollar un primer diseño básico que se adapte a las necesidades fundamentales del sistema. El diseño propuesto incorpora una serie de funcionalidades esenciales que permiten el correcto funcionamiento del sistema, entre las cuales se destacan las siguientes:

- Control de bombas en función de los parámetros establecidos en el sistema.
- Monitorización constante de parámetros para su visualización por parte del operador.
- Control y verificación continua del estado del tanque.
- Acceso sencillo al menú de configuración para facilitar su uso por parte del usuario.

1.2 Análisis de sistemas similares en el mercado

Según lo mencionado en la sección 1.1, se realizó un análisis de mercado en busca de productos que brinden soluciones para las necesidades encontradas. En la tabla 1.1 se elabora un resumen de características principales del producto para tres fabricantes distintos.

Tabla 1.1: Características de productos similares en el mercado.

Característica	ROC-2015	ROC-2315	ROC-8221
Pantalla LED	No	No	Sí
Protección baja presión	Sí	Sí	Sí
Protección alta presión	Sí	Sí	Sí
Bomba de entrada	Sí	Sí	Sí
Bomba de alta presión	No	Sí	Sí
Nivel alto de tanque	Sí	Sí	Sí
Nivel medio de tanque	No	No	Sí
Nivel bajo de tanque	No	No	Sí
precio	107\$	130\$	300\$

Una aclaración importante es que los precios observados en la Tabla 1.1 están expresados en dólares, y los productos mostrados deben ser importados, ya que actualmente no existen proveedores locales que ofrezcan estos modelos. Al analizar la tabla, se observa cómo cada producto añade características específicas que son importantes según la necesidad del sistema.

En este caso, aunque varias opciones podrían encajar bien en el diseño propuesto, la mejor alternativa es el modelo ROC-8221, ya que incluye todas las funcionalidades listadas y satisface por completo los requerimientos del sistema.

De la búsqueda realizada en el mercado de controladores para resolver la problemática planteada, se concluye que las opciones presentadas en el diseño inicial son acertadas y ofrecen una solución viable.

CAPÍTULO 2

Introducción específica

2.1 Requisitos

Después de revisar las características de los productos existentes en el mercado, se realizó el planteamiento de los requerimientos necesarios para cumplir con las expectativas del proyecto. En las tablas 2.1. a 2.4. se identifican los requerimientos por categoría.

Tabla 2.1: Requerimientos de firmware.

Req ID	Descripción
1.1	Implementación de máquinas de estado para gestionar el sistema.
1.2	Control de interrupciones para control de nivel.
1.3	Comunicación UART para interacción con el menú del sistema.
1.4	Procesamiento de datos de sensores de presión y nivel.
1.5	Actualización periódica del LCD con el estado del sistema.

Tabla 2.2: Requerimientos de hardware.

Req ID	Descripción
2.1	Integración de dos bombas y cuatro sensores de nivel.
2.2	Alimentación adecuada para todos los módulos.
2.3	Botón de emergencia conectado al sistema de control.

Tabla 2.3: Requerimientos de interfaz.

Req ID	Descripción
Menú UART para:	
3.1	<ul style="list-style-type: none">- Cambiar la frecuencia de la bomba de ósmosis.- Verificar el estado del sistema.- Detener el sistema completo.
3.2	LCD para mostrar las presiones, nivel del tanque el estado de las bombas.

Tabla 2.4: Requerimientos de seguridad.

Req ID	Descripción
4.1	Implementación del botón de parada de emergencia para detener ambas bombas.

2.2 Casos de uso

Para identificar las funcionalidades básicas y determinar cómo debe comportarse el sistema, se definieron diferentes casos de uso. Cada uno de estos casos contempla los escenarios que el sistema debe resolver, incluyendo sus precondiciones y disparadores.

Tabla 2.5: Caso de uso N° 1.

Elemento del caso de uso	Definición
Disparador	Presión por debajo del umbral mínimo.
Precondición	La bomba está activa.
Flujo básico	La bomba se detiene automáticamente para evitar daños.

Tabla 2.6: Caso de uso N° 2.

Elemento del caso de uso	Definición
Disparador	Se selecciona "Ver estado del sistema" en el menú UART.
Precondición	La conexión UART está establecida.
Flujo básico	Se muestra en el terminal el estado de las bombas y presiones.

Tabla 2.7: Caso de uso N° 3.

Elemento del caso de uso	Definición
Disparador	Se presiona el botón de emergencia.
Precondición	Una o ambas bombas están activas.
Flujo básico	Ambas bombas se detienen inmediatamente.

2.3 Descripción de Módulos

En esta sección se presentan de manera breve los componentes de hardware utilizados durante el desarrollo del trabajo.

2.3.1 Tarjeta NUCLEO

La STM32 Nucleo-F767ZI es una placa de desarrollo diseñada por STMicroelectronics que facilita la creación de prototipos y la evaluación de aplicaciones basadas en el microcontrolador STM32F767ZI. Este microcontrolador se basa en un núcleo ARM® Cortex®-M7 de 32 bits, que opera a una frecuencia de hasta 216 MHz, e integra 2 MB de memoria Flash y 512 KB de RAM, ofreciendo un rendimiento elevado para aplicaciones exigentes [1].

La placa Nucleo-F767ZI pertenece a la familia Nucleo-144, caracterizada por su formato de 144 pines y su compatibilidad con diversas interfaces de expansión, como los conectores Arduino Uno Revision 3 y los encabezados de extensión ST Zio y ST Morpho, que proporcionan acceso completo a todos los pines de entrada/salida del STM32 [2].

Esta placa es una opción versátil y potente para el desarrollo de aplicaciones embebidas que requieren alto rendimiento y múltiples opciones de conectividad,

facilitando la transición del prototipo al producto final, en la figura 2.1 se observa la tarjeta seleccionada [3].

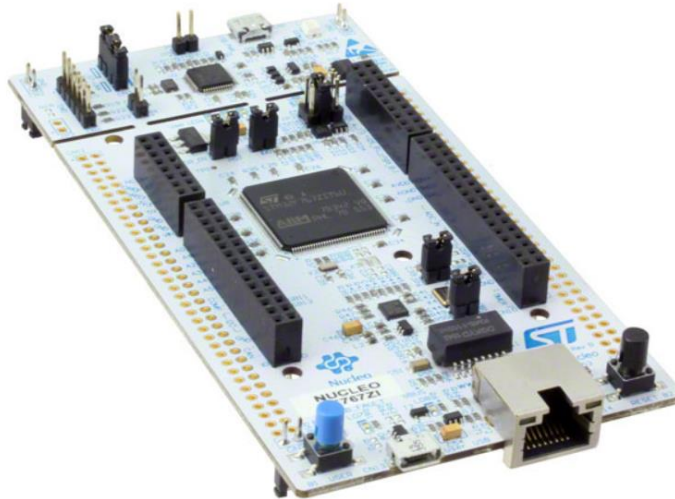


Figura 2.1: Tarjeta NUCLEO-F767ZI.

2.3.2 Módulo Bluetooth HC05

El Bluetooth es un protocolo inalámbrico que ha demostrado ser una solución eficiente para reemplazar la comunicación cableada entre dispositivos electrónicos, destacándose por su bajo consumo de energía y bajo costo. Estas características lo convierten en una opción atractiva para los diseñadores de sistemas embebidos, ya que se adapta bien al desarrollo general de estos sistemas.

El módulo HC-05 se puede observar en la figura 2.2. es sencillo de utilizar. Para integrarlo, se conectan los pines RX y TX del módulo al microcontrolador mediante un convertidor de nivel lógico. Asimismo, los pines Vcc y GND del HC-05 se conectan a 5V y a tierra, respectivamente.

El HC-05 puede configurarse en modo AT para realizar ajustes específicos. Esto se logra presionando un pequeño botón en el módulo al momento de encenderlo. Cabe mencionar que las versiones más antiguas del módulo pueden requerir un procedimiento distinto para activar este modo, por lo que se recomienda verificar el método correspondiente para el modelo en uso. Es importante destacar que la velocidad en baudios para el modo AT es de 38400.

El código para recibir comandos con el HC-05 es relativamente sencillo y permite, por ejemplo, controlar un LED. Los comandos pueden enviarse desde un ordenador o un dispositivo móvil. En el caso de Android o un PC, al emparejar el dispositivo, se introduce el código predeterminado "1234". En un ordenador, el módulo aparece como un puerto COM [4].



Figura 2.2: Módulo Bluetooth HC-05.

2.3.3 Display LCD

El controlador LCD HD44780 es uno de los más populares en el mercado, siendo utilizado ampliamente en aplicaciones industriales, comerciales y por aficionados. Este módulo LCD es monocromático y está disponible en diversas formas y tamaños. Existen modelos con capacidades de 8, 16, 20, 24, 32 y 40 caracteres. Dependiendo del modelo, el display ofrece un conector de 14 o 16 pines para la interfaz. En general, la configuración de pines y las funciones correspondientes de un módulo típico de 14 pines, en la figura 2.3. se observa el display LCD 16x2 [5].



Figura 2.3: Display 16x2.

2.3.4 Modulo I2C

El módulo I2C (Inter-Integrated Circuit) es un adaptador que facilita la conexión y comunicación entre microcontroladores y pantallas LCD, como el modelo 16x2. Este módulo permite reducir la cantidad de pines necesarios para operar el LCD, pasando de 12 o 16 pines, dependiendo de la configuración estándar, a solo 4 (VCC, GND, SDA y SCL). Esto lo convierte en una solución eficiente para proyectos donde el ahorro de pines es crucial, especialmente en sistemas embebidos.

El módulo I2C mostrado en la figura 2.4. emplea un chip controlador, como el PCF8574, que actúa como un expander de puertos y traductor entre la señal serial I2C y los comandos necesarios para el control del LCD. Gracias a este adaptador, se simplifica el código necesario para interactuar con la pantalla, permitiendo configuraciones rápidas y flexibles para la visualización de datos en proyectos electrónicos [6].



Figura 2.4: Modulo I2C conectado al LCD 16x2.

CAPÍTULO 3

Diseño e implementación

3.1 Diagrama de bloques del sistema

En la figura 3.1. se presenta el diagrama de conexiones entre los elementos externos utilizados y el microcontrolador, que actúa como componente principal. En este diagrama se pueden observar todos los componentes empleados, proporcionando una visión más clara y detallada de cómo está estructurado el sistema

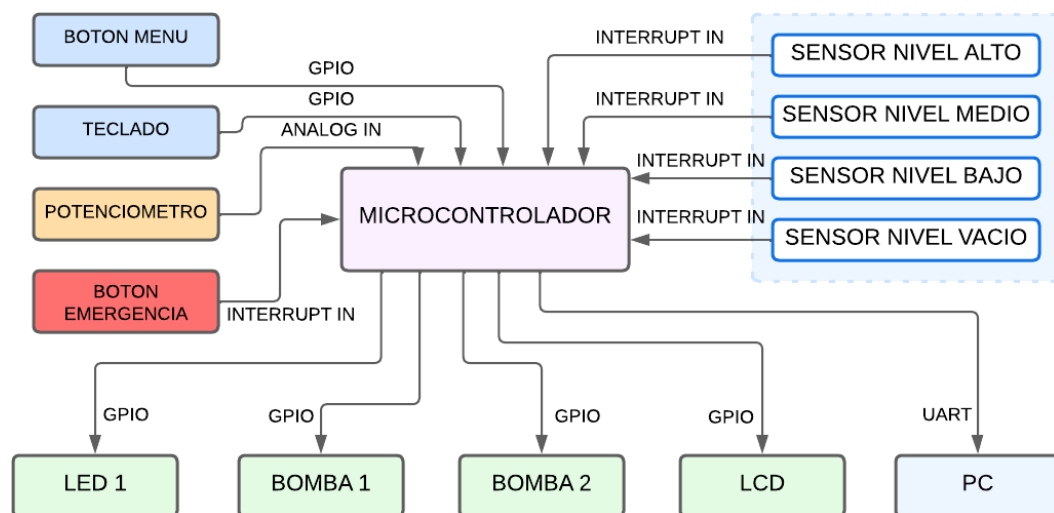


Figura 3.1: Diagrama de bloques del sistema.

A continuación, se detallan tablas el conexionado de pines para cada módulo, en las tablas 3.1. a 3.8. se listan las conexiones cada módulo y componente utilizado.

Tabla 3.1: Conexión entre la tarjeta NUCLEO y HC-05.

Microcontrolador Pin	Componente (HC-05) Pin	Función
PD_5	RX	Transmisión (TX)
PD_6	TX	Recepción (RX)
VCC	VCC	Alimentación
GND	GND	Tierra (GND)

Tabla 3.2: Conexión entre la tarjeta NUCLEO y botones.

Microcontrolador Pin	Componente (Función)	Función
BUTTON1	Botón de menú	Entrada (DigitalIn)
LED1	LED	Salida (DigitalOut)
PA_14	Botón de paro del sistema	Entrada (InterruptIn)

Tabla 3.3: Conexión entre la tarjeta NUCLEO y modulo I2C.

Microcontrolador Pin	Componente (Función)	Función
PB_9	I2C1 SDA	Línea de datos I2C
PB_8	I2C1 SCL	Línea de reloj I2C

Tabla 3.4: Conexión entre la tarjeta NUCLEO y sensores del tanque.

Microcontrolador Pin	Componente (Función)	Función
PC_10	Sensor de nivel alto	Entrada (InterruptIn y DigitalIn)
PC_12	Sensor de nivel medio	Entrada (InterruptIn y DigitalIn)
PF_6	Sensor de nivel bajo	Entrada (InterruptIn y DigitalIn)
PF_7	Sensor de vacío	Entrada (InterruptIn y DigitalIn)

Tabla 3.5: Conexión entre la tarjeta NUCLEO y teclado matricial.

Microcontrolador Pin	Componente (Función)	Función
PE_11	Fila 1 del teclado	Salida (DigitalOut)
PF_3	Fila 2 del teclado	Salida (DigitalOut)
PF_15	Fila 3 del teclado	Salida (DigitalOut)
PF_11	Fila 4 del teclado	Salida (DigitalOut)
PE_0	Columna 1 del teclado	Entrada (DigitalIn)
PG_8	Columna 2 del teclado	Entrada (DigitalIn)
PG_5	Columna 3 del teclado	Entrada (DigitalIn)

Tabla 3.6: Conexión entre la tarjeta NUCLEO y potenciómetro.

Microcontrolador Pin	Componente (Función)	Función
A0	Potenciómetro	Entrada analógica (AnalogIn)

Tabla 3.7: Conexión entre la tarjeta NUCLEO y bombas.

Microcontrolador Pin	Componente (Función)	Función
LED2	Bomba 1	Salida (DigitalOut)
LED3	Bomba 2	Salida (DigitalOut)

Tabla 3.8: Conexión entre la tarjeta NUCLEO y comunicación UART.

Microcontrolador Pin	Componente (Función)	Función
USBTX	Comunicación USB (Transmisión)	Salida (UnbufferedSerial)
USB RX	Comunicación USB (Recepción)	Entrada (UnbufferedSerial)

3.2 Firmware del sistema

El firmware del proyecto fue desarrollado utilizando la plataforma ARM Keil Studio Cloud, empleando una versión compatible con C/C++ que presenta una funcionalidad limitada de las bibliotecas estándar debido a restricciones del hardware. Además, se utilizó MBed OS como sistema operativo para garantizar la compatibilidad durante la compilación del proyecto.

El firmware se desarrolló de manera modular, es decir, cada componente o función específica del sistema se implementó en un módulo independiente. Este enfoque facilitó la integración final y ofreció diversos beneficios, como una mayor escalabilidad en caso de implementar nuevas funciones y una depuración más sencilla. Esto último se debe a que, al momento de identificar errores, es posible localizar y corregir problemas específicos de un componente determinado, en la figura 3.2. se puede observar la estructura del firmware.

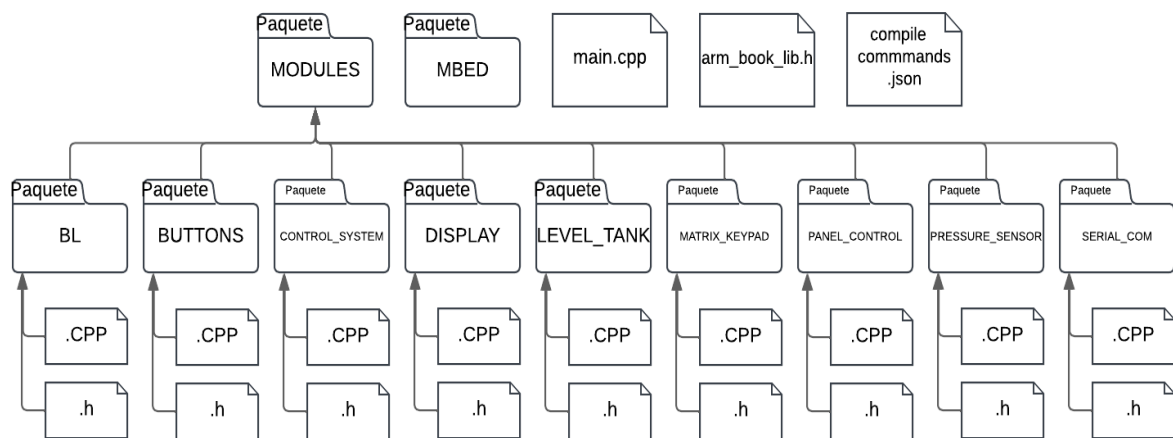


Figura 3.2: Diagrama de archivos del sistema

3.2.1 main

El módulo main.cpp coordina la ejecución del firmware al inicializar el sistema y mantener un bucle infinito que actualiza el sistema de control en tiempo real. Este archivo organiza el flujo general de trabajo y delega las operaciones específicas a otros módulos, en la figura 3.2.1. se muestra el código.

```
TP2 > G+ main.cpp > main
1 //=====[Libraries]=====
2
3 #include "control_system.h"
4
5 //=====[Main function, the program
6
7 int main()
8 {
9     controlSystemInit();
10    while (true) {
11        controlSystemUpdate();
12    }
13 }
```

Figura 3.2.1: Main principal del programa.

3.2.2 control_system

Este módulo se encarga de implementar la lógica central del sistema de control, definiendo cómo se inicializa y actualiza continuamente el sistema durante su operación. Actúa como un intermediario entre los diferentes módulos funcionales, permitiendo una integración fluida de los componentes que conforman el proyecto. La figura 3.2.2. muestra el código usado.

La función `controlSystemInit()` tiene como objetivo realizar la configuración inicial del sistema al invocar las rutinas de inicialización de otros módulos, como el panel de control. Esto asegura que todos los componentes estén listos y configurados correctamente antes de comenzar la operación continua.

Por otro lado, la función `controlSystemUpdate()` se ejecuta de manera cíclica para actualizar el estado del sistema. En este proceso, delega tareas específicas a otros módulos, como el panel de control, y administra pausas sincronizadas para mantener un funcionamiento estable. El diseño modular del archivo permite la incorporación de nuevos componentes en el futuro con facilidad.

```
TP2 > modules > control_system > G+ control_system.cpp
3  #include "arm_book_lib.h"
4
5  #include "control_system.h"
6  #include "control_panel.h"
7  #include "serial_communication.h"
8
9  void controlSystemInit(){
10 |     controlPanelInit();
11 | }
12
13 void controlSystemUpdate()
14 {
15 |     controlPanelUpdate();|
16 |     delay(SYSTEM_TIME_INCREMENT_MS);
17 | }
18
```

Figura 3.2.2: Código del archivo `control_system.cpp`.

3.2.3 control_panel

Este módulo implementa el sistema de control basado en un panel interactivo, estructurado como una máquina de estados para gestionar diversas operaciones del

sistema. Está diseñado para manejar entradas de sensores, teclados matriciales y botones, así como para controlar bombas y mostrar información en un display y vía Bluetooth.

La funcionalidad principal del módulo gira en torno a tres máquinas de estado que coordinan las operaciones del sistema:

1. Estados del display (lcdState_t): La máquina de estados del sistema configura los módulos necesarios y se establece el estado inicial como INIT, preparando el sistema para su funcionamiento.

Un Ticker cambia automáticamente el estado del display cada 4 segundos, recorriendo una secuencia predefinida de estados. Cuando se alcanza el último estado, la secuencia se reinicia, comenzando nuevamente desde el primer estado.

Dependiendo del estado actual, se muestra información específica en la pantalla. Esto puede incluir datos de sensores, el estado de las bombas, o mensajes de emergencia, lo que permite al usuario obtener información relevante en todo momento.

El sistema tiene la capacidad de cambiar a estados específicos según eventos. Estos pueden ser, para la activación de un modo de emergencia o ajustes de configuración, reflejándose inmediatamente en el display, en la figura 3.2.3. se puede observar los primeros cuatro estados.

2. Ajuste de frecuencia (ajustFrequency_t): Permite la configuración de parámetros del sistema a través del teclado matricial y un menú interactivo. Los estados van desde la presentación del menú, lectura de teclas, confirmación de datos, y salida del proceso de ajuste.
3. Opciones del menú (options_menu_t): Define las acciones disponibles, como cambiar la frecuencia, consultar el estado del sistema, y detener el sistema. Cada opción ejecuta una rutina específica para cumplir su función.



```
static void updateMenuDisplay() {
    char buffer[16];
    switch (lcdState) {
        case INIT:
            displayCharPositionWrite(0, 0);
            displayStringWrite("Iniciando");
            writeB1("\nIniciando",0);
            displayCharPositionWrite(0, 1);
            displayStringWrite("Sistema...");
            writeB1("\nSistema...",0);
            break;

        case PRESSURE_SP1:
            displayCharPositionWrite(0, 0);
            displayStringWrite("Presion SP1: ");
            writeB1("\nPresion SP1: ",1);
            displayCharPositionWrite(0, 1);
            strPressureM1(buffer,
                sizeof(buffer));
            displayStringWrite(buffer);
            displayStringWrite(" psi ");
            writeB1(buffer,1);
            break;

        case PRESSURE_SP2:
            displayCharPositionWrite(0, 0);
            displayStringWrite("Presion SP2: ");
            writeB1("\nPresion SP2: ",2);
            displayCharPositionWrite(0, 1);
            strPressureM2(buffer, sizeof(buffer));
            displayStringWrite(buffer);
            displayStringWrite(" psi ");
            writeB1(buffer,2);
            break;

        case PUMP1:
            displayCharPositionWrite(0, 0);
            displayStringWrite("Bomba 1: ");
            writeB1("\nBomba 1: ",3);
            displayCharPositionWrite(0, 1);
            length = snprintf(buffer, sizeof(buffer), "%s",
                StatePump1Read() ? "Encendida" : "Apagada ");
            displayStringWrite(buffer);
            writeB1(buffer,3);
            break;
    }
}
```

Figura 3.2.3: Máquina de estados para actualizar el lcd.

3.2.4 BL

Este módulo gestiona la comunicación con un dispositivo Bluetooth a través de un puerto serie UART, permitiendo el envío de mensajes y su control de visualización en una pantalla LCD. Utiliza un objeto `UnbufferedSerial` para establecer la comunicación en un puerto específico con una velocidad de transmisión de 9600 baudios. Para evitar el envío redundante de mensajes, el módulo implementa un sistema de control basado en la variable `lcdCase`, que asegura que solo se envíen nuevos mensajes cuando el estado de la pantalla LCD cambia. Además, la función `writeBl` maneja el envío de los mensajes, asegurando que no se repitan innecesariamente y controlando la organización de la visualización en dos filas del LCD. El módulo también cuenta con una función personalizada para calcular la longitud de los mensajes, optimizando el proceso de transmisión.

La función `bluetoothInit()` establece la inicialización del sistema Bluetooth, de esta manera, el código proporciona un manejo eficiente de los mensajes enviados y su visualización en el dispositivo, manteniendo la comunicación con el Bluetooth y controlando la interfaz LCD.

3.2.5 buttons

Se utiliza para monitorear el estado de un botón (`menuButton`) y un botón de parada de emergencia (`stopSystemButton`), ambos configurados con ciertas características de hardware y debouncing para garantizar una lectura precisa y estable de los estados del botón.

La función principal `buttonsUpdate()` controla el estado del `menuButton` a través de una máquina de estados que garantiza que el sistema sea capaz de manejar correctamente los rebotes del botón (debouncing). La parada de emergencia (`stopSystemButton`), se activa cuando hay un cambio en el estado del pin de bajo a alto, deteniendo el sistema de control en caso de emergencia.

3.2.6 level_tank

Este código está diseñado para gestionar y controlar el nivel de un tanque utilizando varios sensores de nivel en diferentes alturas. A través de interrupciones y el monitoreo de señales de los sensores, el sistema determina el nivel de líquido en el tanque y ejecuta las acciones necesarias, como detener el llenado del tanque o iniciar el proceso de llenado según el estado del nivel.

El sistema utiliza varios sensores de nivel ubicados en diferentes puntos del tanque: alto, medio, bajo y vacío. Cada uno de estos sensores está configurado para generar interrupciones cuando el nivel del tanque alcanza los puntos correspondientes. Cuando se detecta un cambio en la señal de los sensores, se activan las funciones de callback, que actualizan el estado del nivel del tanque (por ejemplo, "ALTO", "MEDIO", "BAJO", "VACÍO", o "ERROR SENSOR").

El sistema maneja errores mediante la transición a un estado de "FAIL" si se detecta alguna inconsistencia en los sensores. Este control es esencial para automatizar el proceso de monitoreo y regulación de niveles en el tanque.

3.2.7 pressure_sensor

Este código está diseñado para monitorear y controlar el sistema de presión de dos sensores de presión (presión 1 y presión 2) en un sistema. El sistema utiliza un potenciómetro para leer las señales analógicas, que luego se convierten en valores de presión para activar o detener las bombas según los umbrales establecidos.

Se definen dos umbrales de presión: `PRESSURE_CUT_IN_P1` (umbral mínimo) y `PRESSURE_CUT_OFF_P1` (umbral máximo). Cuando la presión medida por los sensores de presión cae por debajo del umbral mínimo, las bombas se detienen. Si la presión está entre los dos umbrales, las bombas se mantienen en funcionamiento. Si la presión excede el umbral máximo, las bombas también se detienen. El sistema monitorea continuamente las presiones de ambos sensores de forma sincronizada.

Se utiliza un *ticker* (`timeCheckPressure`) que asegura la verificación de la presión se realice de manera repetida a intervalos regulares de 1500 ms.

3.2.8 pumps

El código gestiona el control de las dos bombas según la presión, permite encender o apagar ambas bombas mediante las funciones `runPumps()` y `stopPumps()`, que modifican el estado de las bombas estableciendo a `true` o `false` el estado de `pump1` y `pump2`.

3.2.9 serial_communication

Este código está diseñado para facilitar la comunicación serial entre la tarjeta NUCLEO y la PC través de un puerto USB, utilizando `uartUsb`. Permite la interacción con el usuario para realizar configuraciones o verificar el estado del sistema, está configurado para operar

con una tasa de baudios de 115200, lo que significa que la comunicación entre el dispositivo y el puerto USB se realiza a esta velocidad, y permite a los usuarios interactuar con el sistema a través de un menú simple en el terminal.

CAPÍTULO 4

Ensayos y resultados

4.1 Pruebas funcionales

Se realizaron diferentes pruebas funcionales del hardware y el firmware asociado a cada módulo por separado.

4.1.1 Comunicación serie

La primera prueba consistió en verificar la comunicación serial. En esta fase, también se realizaron pruebas con el teclado matricial y el botón de inicio de la comunicación UART, el cual se activa al mantener presionado el botón. La prueba consistió en establecer una conexión con la terminal del PC y verificar que las opciones se desplegaran correctamente, tal como se muestra en la figura 4.1. en la cual se despliegan tres opciones.

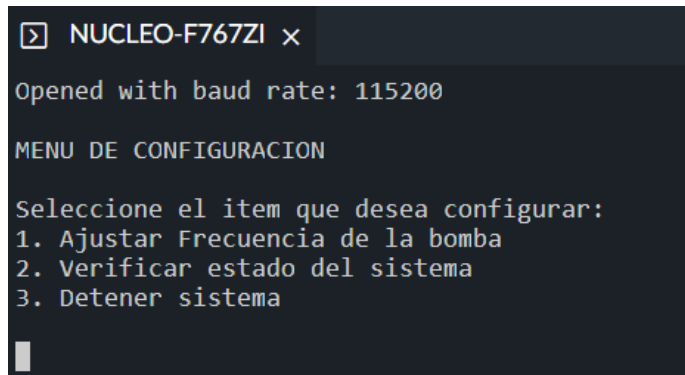


Figura 4.1: Opciones de configuración desde el PC.

La primera opción consiste en cambiar la frecuencia de funcionamiento de las bombas. El objetivo es que este valor se transmita a un variador de frecuencia, lo que permitirá ajustar el funcionamiento de las bombas según lo establezca el usuario. En este caso, se muestra en la figura 4.2 una frecuencia de 45 Hz, ingresada por el usuario, la cual fue procesada de manera exitosa.

```
> NUCLEO-F767ZI x
Opened with baud rate: 115200

MENU DE CONFIGURACION

Seleccione el item que desea configurar:
1. Ajustar Frecuencia de la bomba
2. Verificar estado del sistema
3. Detener sistema

Ingrese Frecuencia:
45 Hz
Para confirmar frecuencia presione #
Para cancelar frecuencia presione *
Frecuencia cambiada!
```

Figura 4.2: Cambio de frecuencia de la bomba desde el PC.

La segunda opción permite listar los parámetros actuales del sistema, revisando las presiones y el estado de las bombas, tal como se muestra en la figura 4.3.

```
> NUCLEO-F767ZI x
Opened with baud rate: 115200

MENU DE CONFIGURACION

Seleccione el item que desea configurar:
1. Ajustar Frecuencia de la bomba
2. Verificar estado del sistema
3. Detener sistema

El estado del sistema es:

Presión 1: 64.26
Presión 2: 59.26
Bomba 1: Encendida
Bomba 2: Encendida
```

Figura 4.2: Verificación de parametros desde el PC.

La tercera opción se ejecuta directamente sobre las bombas, sin tener en cuenta las presiones o el estado de las mismas, activando un paro de emergencia en el sistema.

4.1.2 Display

Para las pruebas con el display, se comenzó conectando el módulo de comunicación I2C al LCD como se observa en la figura 4.3. lo que permitió utilizar menos puntos de conexión en la tarjeta manteniendo la misma funcionalidad. Básicamente, se verificó que cada 4 segundos el sistema muestre los parámetros de las presiones 1 y 2, el estado de las bombas 1 y 2, así como el estado de nivel del tanque. El control del estado del LCD se gestionará mediante una máquina de estados. En condiciones normales, el LCD mostrará únicamente los parámetros mencionados previamente. Sin embargo, si se presenta un paro de emergencia, el LCD cambiará a un estado de paro, indicando que se encuentra en esa situación.

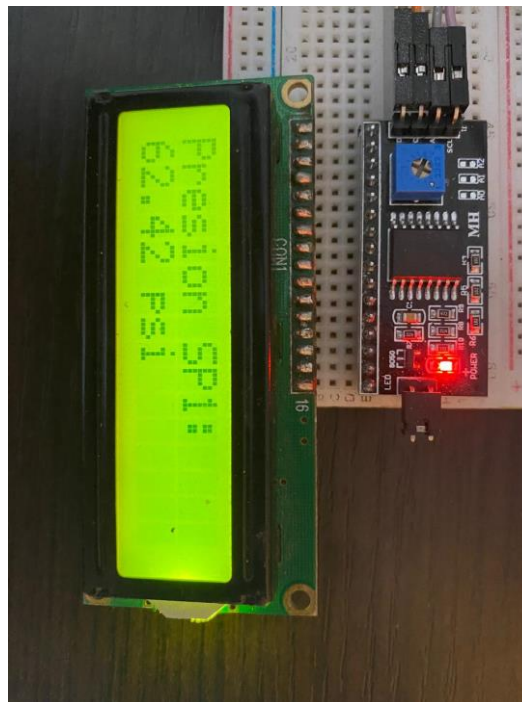


Figura 4.4: Prueba de comunicación entre tarjeta NUCLEO y LCD.

4.1.2 Comunicación bluetooth

La comunicación Bluetooth se implementó mediante el código mostrado en la Figura 4.5. Básicamente, el sistema envía los mismos datos que se están visualizando en la pantalla LCD. Se implementó un control de envío basado en el estado actual y el último estado, con el objetivo de que, durante los 4 segundos que dura la transacción de cada estado en la

pantalla LCD, solo se realice un único envío por Bluetooth. Esto se hizo para evitar el envío de información redundante y garantizar una transmisión eficiente de los datos.

```
//====[Libraries]=====
#include "mbed.h"
#include <cstring>
//====[Declaration of private defines]=====
UnbufferedSerial uartBl(PD_5, PD_6, 9600);
//====[Declaration of private data types]=====
Ticker tickerSendBl;
static int lcdCase=10;
static int lastLcdCase=10;
static int counterRowsLcd=0;
//====[Declarations (prototypes) of private functions]=====
static void sendDataBl();
//====[Implementations of public functions]=====
void bluetoothInit(){
    //tickerSendBl.attach(sendDataBl, 4000ms);
}

size_t custom_strlen(const char *str) {
    size_t len = 0;
    while (str[len] != '\0') {
        len++;
    }
    return len;
}

void writeBl(const char *message, int lcdCase) {
    if (lcdCase != lastLcdCase){
        size_t bufferLength = custom_strlen(message);
        uartBl.write(message, bufferLength);
        counterRowsLcd=counterRowsLcd+1;
        if (counterRowsLcd==2){
            lastLcdCase=lcdCase;
            counterRowsLcd=0;
        }
    }
}
```

Figura 4.5: Código del módulo BL.

Los parámetros del dispositivo HC-05 se dejaron por defecto, razón por la cual no fue necesario configurarlo mediante comandos AT. La recepción de los datos se realizó a través de un celular con Android, utilizando una aplicación llamada "Serial Bluetooth Terminal". En la Figura 4.6 se puede observar cómo se reciben todos los estados mostrados en el LCD incluyendo el mensaje de paro de emergencia.

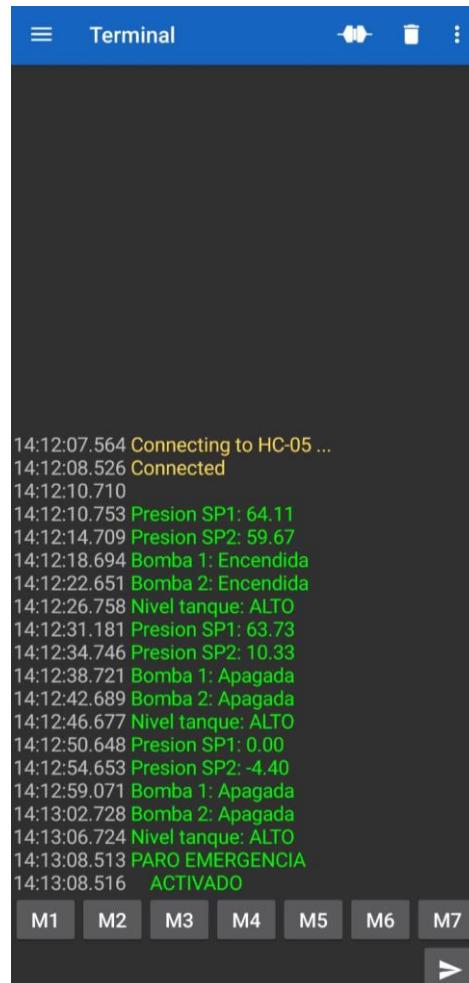


Figura 4.6: Recepción de datos en un terminal android.

4.1.3 Modulo de presiones

Para iniciar con el módulo de presiones, se configuraron inicialmente las lecturas analógicas proporcionadas por el potenciómetro, de modo que, al girar en los extremos, el potenciómetro se convierte en una señal que emula una presión de 0 y 99 PSI, según el

extremo, para la presión 1. Para la presión 2, básicamente se resta 5 PSI de la lectura de la presión 1.

La verificación de este módulo se realizó utilizando la comunicación serial, donde se comprobó que, si la presión se encuentra dentro del rango de 20 a 80 PSI, las bombas deben funcionar. Si alguna presión está fuera de este rango, las bombas no deben funcionar, tal como se muestra en las figuras 4.7. a 4.8.

```
NUCLEO-F767ZI x
Opened with baud rate: 115200
MENU DE CONFIGURACION
Seleccione el item que desea configurar:
1. Ajustar Frecuencia de la bomba
2. Verificar estado del sistema
3. Detener sistema
El estado del sistema es:
Presión 1: 96.75
Presión 2: 91.75
Bomba 1: Apagada
Bomba 2: Apagada
```

Figura 4.6: Estado de las bombas con una presión por encima del umbral.

```
MENU DE CONFIGURACION
Seleccione el item que desea configurar:
1. Ajustar Frecuencia de la bomba
2. Verificar estado del sistema
3. Detener sistema
El estado del sistema es:
Presión 1: 1.16
Presión 2: -3.84
Bomba 1: Apagada
Bomba 2: Apagada
```

Figura 4.7: Estado de las bombas con una presión por debajo del umbral.


```
Seleccione el item que desea configurar:
1. Ajustar Frecuencia de la bomba
2. Verificar estado del sistema
3. Detener sistema

El estado del sistema es:

Presión 1: 62.66

Presión 2: 57.66

Bomba 1: Encendida

Bomba 2: Encendida
█
```

Figura 4.8: Estado de las bombas con una presión dentro del rango configurado.

4.1.4 Sensores de nivel

Para probar los sensores de nivel, se utilizó un DIP switch para emular, a través de los cambios de estado, el comportamiento del sensor. Una posición en alto del switch se considera como la presencia de líquido, y viceversa, una posición baja se considera que no hay líquido en ese nivel.

En la lógica, se implementó una verificación por cambio de estado, lo que significa que, si se detecta presencia de líquido en el sensor de nivel medio, se verifican los otros sensores para revisar si el nivel del líquido está aumentando o disminuyendo. Esto también permitió implementar un control de errores, es decir, no es posible que se detecte líquido en el sensor superior y que no se haya detectado líquido en el sensor inferior. En caso de que esto ocurra, se genera un estado de fallo en los sensores.

El estado de los sensores también controla el funcionamiento de las bombas, lo que significa que, si se activa el sensor de nivel del tanque alto, indicando que el tanque está lleno, las bombas deben detenerse. Para fines prácticos, si el tanque está vacío, las bombas también se detendrán sin importar las presiones. Esto se ilustra en la figura 4.8.

```
14:42:33.071 Bomba 2: Encendida
14:42:37.077 Nivel tanque: ALTO
14:42:41.113 Presion SP1: 60.80
14:42:45.040 Presion SP2: 56.82
14:42:49.033 Bomba 1: Encendida
14:42:53.123 Bomba 2: Encendida
14:42:57.575 Nivel tanque: BAJO
14:43:01.078 Presion SP1: 62.95
14:43:05.096 Presion SP2: 57.83
14:43:09.540 Bomba 1: Encendida
14:43:13.164 Bomba 2: Encendida
14:43:17.130 Nivel tanque: BAJO
14:43:21.106 Presion SP1: 60.80
14:43:25.107 Presion SP2: 56.41
14:43:29.494 Bomba 1: Apagada
14:43:33.043 Bomba 2: Apagada
14:43:37.162 Nivel tanque: VACIO
14:43:41.465 Presion SP1: 61.41
14:43:45.142 Presion SP2: 56.41
14:43:49.113 Bomba 1: Apagada
```

Figura 4.9: Verificación de los sensores de nivel.

4.2 Pruebas de integración

Para las pruebas de integración, se conectaron todos los componentes a la tarjeta NUCLEO y se verificó que no hubiera conflictos de funcionamiento. Es decir, al interactuar con un módulo, no debía causar interferencias sobre la funcionalidad de otro. En la figura 4.10. se puede observar el montaje del diseño, y en el [video](#) se puede ver la funcionalidad del sistema con todos sus componentes.

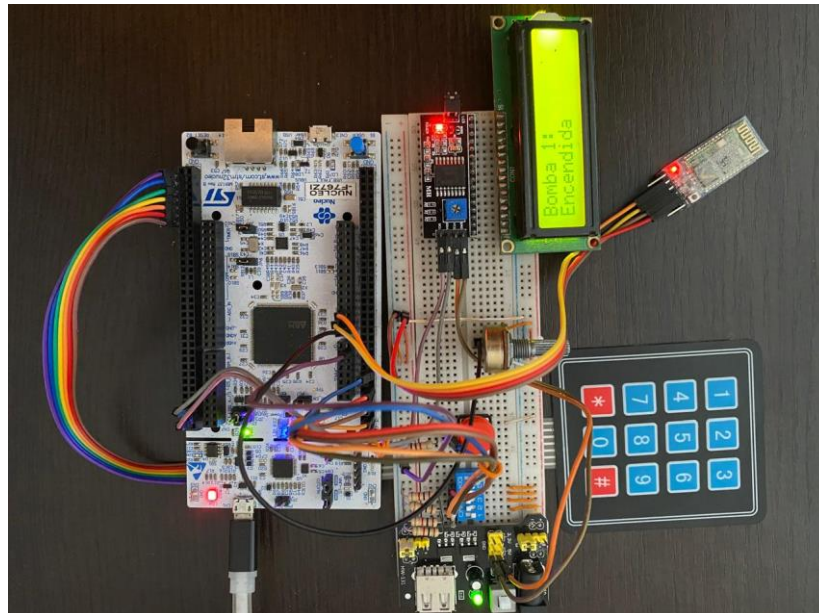


Figura 4.10: Implementación inicial del sistema.

4.3 Cumplimiento de requisitos

Finalmente, se realiza un análisis del cumplimiento de los requisitos del trabajo. En la tabla 4.1 se presentan los requisitos iniciales junto con el estado final de cada uno de ellos.

Tabla 4.1: Requisitos iniciales del proyecto.

Categoría	Req ID	Descripción	Estado
Firmware	1.1	Implementación de máquinas de estado para gestionar el sistema.	✓
	1.2	Control de interrupciones para control de nivel.	✓
	1.3	Comunicación UART para interacción con el menú del sistema.	✓
	1.4	Procesamiento de datos de sensores de presión y nivel.	✓
	1.5	Actualización periódica del LCD con el estado del sistema.	✓
Hardware	2.1	Integración de dos bombas y cuatro sensores de nivel.	✓
	2.2	Alimentación adecuada para todos los módulos.	✓
	2.3	Botón de emergencia conectado al sistema de control.	✓
Interfaz	3.1	Menú UART para configurar: - Cambiar la frecuencia de la bomba de ósmosis. - Verificar el estado del sistema. - Detener el sistema completo.	✓
	3.2	LCD para mostrar las presiones, nivel del tanque y el estado de las bombas.	✓
Seguridad	4.1	Implementación del botón de parada de emergencia para detener ambas bombas.	✓

4.5 Documentación del desarrollo realizado

Tabla 4.2: Elementos del sumario.

Elemento	Referencia
Presentación del proyecto	Sección 1.1
Listado de requisitos	Tabla 2.1 a 2.4
Casos de uso del proyecto	Sección 2.2
Tabla de conexiones entre módulos	Tablas 3.1 a 3.8
Diagrama en bloques del sistema	Figura 3.1
Implementación del hardware	Sección 3.1
Módulos de hardware del sistema	Sección 2.3
Módulos del software del sistema	Sección 3.2
Repositorio	Link a GitHub
Cumplimiento de requisitos	Tabla 4.1
Conclusiones finales	Capítulo 5
Propuestas para próximos pasos	Sección 5.2

CAPÍTULO 5

Conclusiones

5.1 Resultados obtenidos

El resultado principal del trabajo es el prototipo del sistema funcionando con los requisitos y casos de uso listados. Los logros obtenidos al finalizar el trabajo son los siguientes:

- Interacción entre el sistema y el pc mediante una conexión serie.
- Conexión remota mediante bluetooth para visualización de datos.
- Verificación de parámetros en tiempo real del sistema mediante un LCD.
- Paro de emergencia implementado por software y hardware.
- Actualización y verificación de sensores de nivel para evitar errores.

En general, el presente trabajo permitió desarrollar un diseño con funcionalidades básicas, ofreciendo un primer acercamiento a una necesidad concreta. Esto proporcionó una mejor comprensión de cómo abordar el desarrollo de un sistema más robusto y completo. Es relevante destacar que modularizar el código facilitó la organización y el reuso de componentes en diferentes partes del sistema, logrando una implementación más sencilla y ordenada.

5.2 Próximos pasos

En el futuro, el sistema tiene pendiente incorporar nuevas funcionalidades que le permitan adaptarse de manera más eficiente a un sistema de tratamiento de agua. Algunas de las funcionalidades que podrían ser útiles en este contexto son las siguientes:

- Generar un componente de reloj en tiempo real y complementarlo con un sistema de logs, para asegurar la trazabilidad y el seguimiento del sistema.
- Adaptar los sensores de presión a un módulo de 4 a 20 mA, dado que la mayoría de los sensores de presión disponibles en el mercado utilizan este tipo de salida.
- Establecer una comunicación entre la tarjeta NUCLEO y el variador de velocidad para optimizar el control de las bombas.
- Implementar una comunicación bidireccional utilizando Bluetooth para facilitar el intercambio de información entre el sistema y dispositivos externos.

- Revisar una estrategia más eficiente para monitorear el llenado del tanque, considerando el uso de un sensor de ultrasonido, que podría adaptarse mejor a este propósito.

En conclusión, con estas mejoras, el sistema se consolidará como un controlador más robusto, lo que incrementará su utilidad en el contexto del tratamiento de agua, proporcionando una solución más eficiente y escalable.

Bibliografía

- [1] «STMicroelectronics,» 30 10 2024. [En línea]. Available:
<https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html>.
- [2] 01 10 2024. [En línea]. Available:
<https://www.st.com/resource/en/datasheet/stm32f767zi.pdf>.
- [3] A. LUTENBERG, P. GOMEZ y E. PERNIA, A Beginner's Guide to Designing Embedded System Applications on Arm® Cortex®-M Microcontrollers, Arm Education Media, 2022.
- [4] A. Subero, Programming PIC Microcontrollers with XC8, Apress, 2017.
- [5] D. Ibrahim, Advanced PIC Microcontroller Projects in C, Newnes, 2011.
- [6] F. R. Domínguez, Microcontrolador PIC16F84. Desarrollo de proyectos. 3ª edición, Grupo Editorial RA-MA, 2004.