Microcontroller & Interfacing

CE205T

	CLO-2		CLO-3					Total
Part	B	C	A	D	E	F	G	
Marks	50	50	50	50	50	50	50	
Obt.								

Project Name

Water Monitoring And Controlling System

Group Number

1: Tayyab Akram	BSCE-23055
2: Muqarab Nazir	BSCE-23060
3: Minahil Imran	BSCE-23030

A. Overview [CLO-3, 50 Marks]

The Water Level Monitoring & Controlling project is an embedded system built using the STM32F407vgt6 microcontroller to monitor and manage water levels in a tank while also keeping track of environmental conditions. The system uses an ultrasonic sensor to measure water level and a DHT22 sensor to monitor temperature and humidity. A 16×2 LCD displays real-time readings, while an HC-05 Bluetooth module transmits the data to a mobile device. Three LEDs are used to indicate different water levels visually. The system includes two buzzers—one activates when the water level exceeds a user-defined threshold, and the other alerts when the temperature limit is crossed. A potentiometer is integrated to set the water level threshold dynamically, triggering the corresponding buzzer when the limit is reached. This project demonstrates the integration of multiple sensors and output components with efficient threshold-based control and user interaction

GOALS

□ Setting a configurable threshold for the water level using a potentiometer and activating a buzzer when the threshold is exceeded.

□ Measuring water level using an ultrasonic sensor and displaying it along with temperature and humidity data (from the DHT22 sensor) on a 16×2 LCD.

□ Transmitting real-time sensor data to a mobile device via the HC-05 Bluetooth module.

□ Indicating different water levels using three LEDs for intuitive visual monitoring.

□ Activating a separate buzzer when the ambient temperature exceeds a predefined limit.

□ Using an interrupt-driven user button on the STM32F407 microcontroller to cycle through displayed information on the LCD.

List of Components Used [CLO-2, 50 Marks]

Sr #	Component	Cost (Rs) (Unit Price)	Link to Datasheet	Operating Principle
1	STM32F407VGT6	From LAB		Microcontroller
2	Ultrasonic HCSR04	180		Time-of-Flight
3	16*2 LCD Display	250		Electronic Visual
4	Potentiometer	30		Variable Resistor
5	Buzzer	30		Sound Alert
6	ST-Link	From LAB		Serial Communication
7	DHT22 Sensor	180		Digital Humidity
8	Hc05 Bluetooth module	750		Wireless Serial
9	LEDs			Visual Indicators

B. Peripherals of STM Microcontroller being used [CLO-2, 50 Marks]

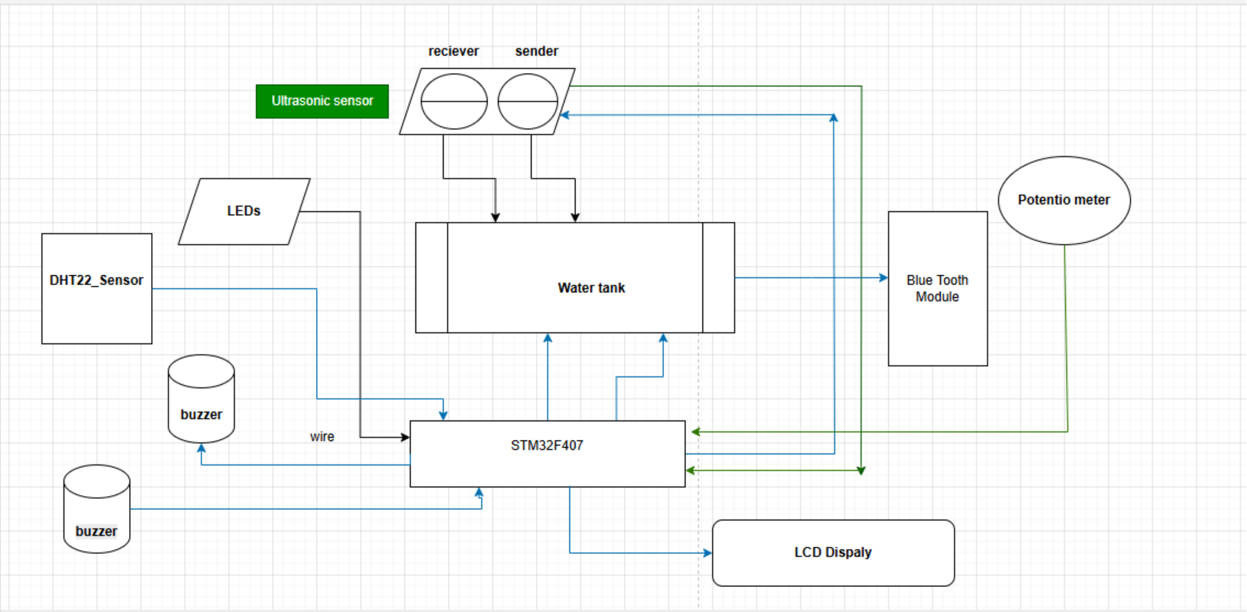
□ **GPIO Pins**: These are used to interface with various components such as the water level sensor, LCD display, potentiometer, buzzer, LEDs, HC05 and DHT22.

□ **ADC (Analog-to-Digital Converter)**: This is used to read the analog voltage from the potentiometer to configure the water level threshold.

□ **Timers**: These are employed in the functionality of ultrasonic sensor to accurately calculate the water level distance.

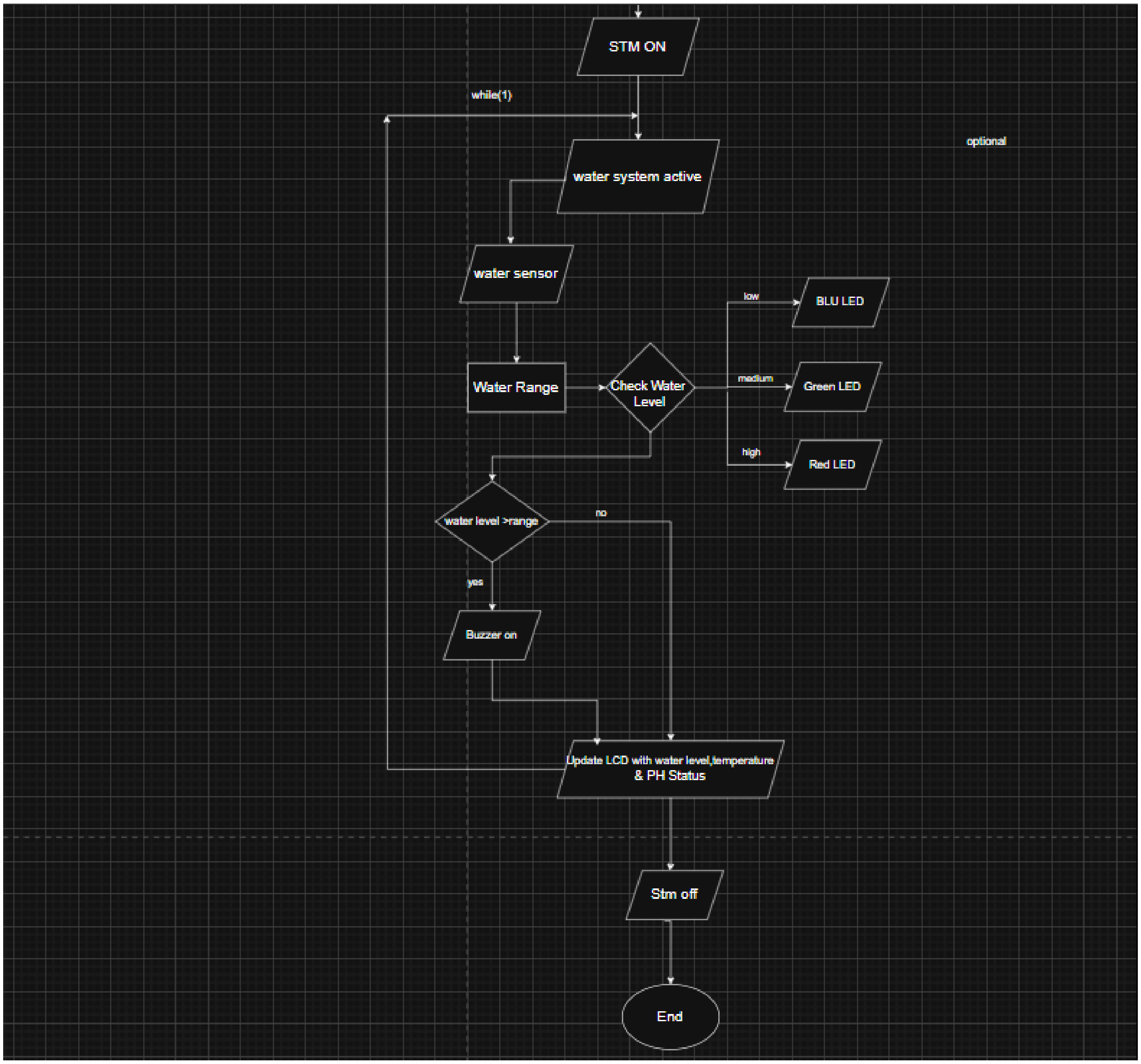
□ **USART/UART**: This interface is used with Blue tooth Module HC05 to send data to Mobile.

C. Block Diagram/Schematic [CLO-3, 50 Marks]



D. Flow Chart (Required at the time of final submission)

[CLO-3, 50 Marks]



E. CEP (Project Complexity) Attributes - Describe Briefly [CLO-3, 50 Marks]

Attribute	Description	Complexity Level in your project
WP1: Depth of	The project shall involve in-	This project applied concepts from microcontroller

knowledge	depth engineering knowledge related to the area of Microprocessors, Microcontrollers & Interfacing [WK-4, Engineering Specialization].	programming, sensor interfacing, and communication protocols learned during the course. It helped me integrate ultrasonic and DHT22 sensors, Bluetooth communication, and implement interrupt-driven controls effectively.
WP2: Range of conflicting requirements	The project has multiple conflicting requirements in terms of optimal usage of peripheral resources available on a Microcontroller.	One major challenge was the limited number of GPIO pins and interrupt lines on the STM32F407, as we had to interface multiple components like ultrasonic sensor, DHT22, LCD, buzzers, LEDs, a potentiometer (via ADC), and a Bluetooth module. Managing their timing and functionality simultaneously—especially for real-time display and alarms—required careful pin mapping and efficient interrupt handling.
WP5 Extent of applicable codes	The projects expose the students to broadly defined problems which require the development of codes that may be partially outside those encompassed by well-documented standards.	Our code involved moderate complexity as it required real-time monitoring of water level and temperature using interrupts and ADC. We modeled real-time control by continuously comparing sensor data against user-defined thresholds and triggering outputs (LEDs, buzzers, and Bluetooth messages) instantly based on those conditions.
WP7 Interdependence	The projects shall have multiple components at the hardware and software level.	The main challenge we faced during the project was integrating a waterproof temperature sensor with the STM32 microcontroller. Although it was essential for our water-based system, we encountered compatibility issues, and aligning its communication protocol with the STM32 proved to be the hardest part of the interfacing process.

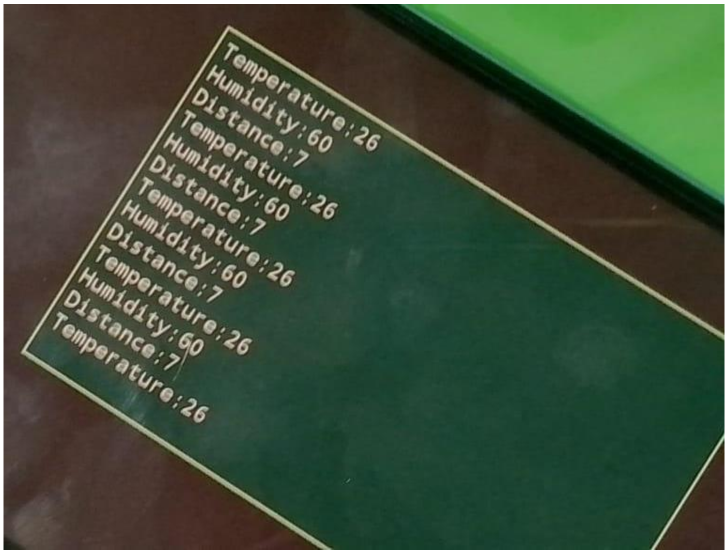
F. Code [CLO-3, 50 Marks]

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2025 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* 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);
```

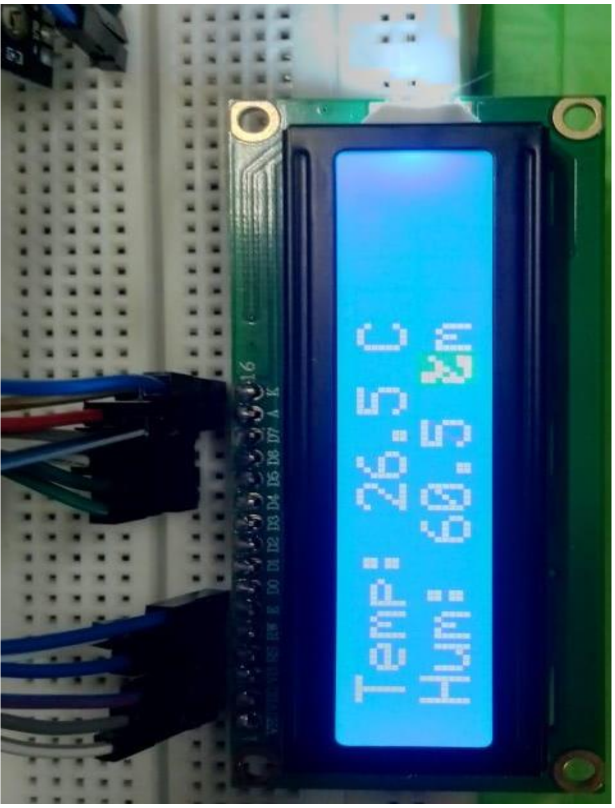
```
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#define DHT22_PORT GPIOA
#define DHT22_PIN GPIO_PIN_1
//#define LED_PORT GPIOD
//#define LED_PIN GPIO_PIN_13

#define TRIG_PIN GPIO_PIN_7
#define TRIG_PORT GPIOA
#define ECHO_PIN GPIO_PIN_6
#define ECHO_PORT GPIOA

#define LCD_RS_GPIO_Port GPIOB
#define LCD_RS_Pin GPIO_PIN_0

#define LCD_EN_GPIO_Port GPIOB
#define LCD_EN_Pin GPIO_PIN_1

#define LCD_D4_GPIO_Port GPIOB
#define LCD_D4_Pin GPIO_PIN_4

#define LCD_D5_GPIO_Port GPIOB
#define LCD_D5_Pin GPIO_PIN_5

#define LCD_D6_GPIO_Port GPIOB
#define LCD_D6_Pin GPIO_PIN_6

#define LCD_D7_GPIO_Port GPIOB
#define LCD_D7_Pin GPIO_PIN_7
char lcdBuffer[32];


uint8_t RH1, RH2, TC1, TC2, SUM, CHECK;
uint32_t pMillis, cMillis;
float tCelsius = 0;
float tFahrenheit = 0;
float RH = 0;
uint8_t Response = 0;
```

```
volatile int lcd_debug = 0;
float humidity=0.0;
//////////////
uint32_t pMillis;
uint32_t val1 = 0;
uint32_t val2 = 0;
uint16_t distance  = 0;
////////////
void microDelay(uint16_t delay)
{

  __HAL_TIM_SET_COUNTER(&htim2, 0);

  while (__HAL_TIM_GET_COUNTER(&htim2) < delay);

}
uint8_t DHT22_Start(void)
{

  GPIO_InitTypeDef GPIO_InitStructPrivate = {0};

  // Set pin as output
  GPIO_InitStructPrivate.Pin = DHT22_PIN;
  GPIO_InitStructPrivate.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStructPrivate.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStructPrivate.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DHT22_PORT, &GPIO_InitStructPrivate);

  // Send start signal
  HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, GPIO_PIN_RESET);
//  microDelay(1300);
  HAL_Delay(20);

  HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, GPIO_PIN_SET);
  microDelay(30);


  // Set pin as input
  GPIO_InitStructPrivate.Mode = GPIO_MODE_INPUT;
  GPIO_InitStructPrivate.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(DHT22_PORT, &GPIO_InitStructPrivate);

  microDelay(40);
  if (!(HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)))
  {
    microDelay(80);

    if ((HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)))
            {

            Response = 1;
            }
  }

  pMillis = HAL_GetTick();
  cMillis = HAL_GetTick();
  while ((HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)) && pMillis + 2 > cMillis)
  {

    cMillis = HAL_GetTick();
  }

  return Response;
}

uint8_t DHT22_Read(void)
{
  uint8_t i, b = 0;
  for (i = 0; i < 8; i++)
  {
    pMillis = HAL_GetTick();
    cMillis = HAL_GetTick();
    while (!(HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)) && pMillis + 2 > cMillis)
    {
      cMillis = HAL_GetTick();
    }

    microDelay(40);

    if (!(HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)))
      b &= ~(1 << (7 - i));
    else
      b |= (1 << (7 - i));

    pMillis = HAL_GetTick();
    cMillis = HAL_GetTick();
    while ((HAL_GPIO_ReadPin(DHT22_PORT, DHT22_PIN)) && pMillis + 2 > cMillis)
    {
      cMillis = HAL_GetTick();
    }
  }

  return b;
}
void LCD_Enable(void)
{    lcd_debug = 150;
  HAL_GPIO_WritePin(LCD_EN_GPIO_Port, LCD_EN_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(LCD_EN_GPIO_Port, LCD_EN_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
  lcd_debug = 160;
}

void LCD_Send4Bits(uint8_t data)
{
            lcd_debug = 170;
  HAL_GPIO_WritePin(LCD_D4_GPIO_Port, LCD_D4_Pin, (data >> 0) & 0x01);
  HAL_GPIO_WritePin(LCD_D5_GPIO_Port, LCD_D5_Pin, (data >> 1) & 0x01);
  HAL_GPIO_WritePin(LCD_D6_GPIO_Port, LCD_D6_Pin, (data >> 2) & 0x01);
  HAL_GPIO_WritePin(LCD_D7_GPIO_Port, LCD_D7_Pin, (data >> 3) & 0x01);
}

void LCD_SendCmd(uint8_t cmd)
{
            lcd_debug = 180;
  HAL_GPIO_WritePin(LCD_RS_GPIO_Port, LCD_RS_Pin, GPIO_PIN_RESET);
  LCD_Send4Bits(cmd >> 4);
  LCD_Enable();
  LCD_Send4Bits(cmd & 0x0F);
  LCD_Enable();
  HAL_Delay(2);
  lcd_debug = 190;
}

void LCD_SendData(uint8_t data)
{
           lcd_debug = 200;
  HAL_GPIO_WritePin(LCD_RS_GPIO_Port, LCD_RS_Pin, GPIO_PIN_SET);
  LCD_Send4Bits(data >> 4);
  LCD_Enable();
  LCD_Send4Bits(data & 0x0F);
  LCD_Enable();
  HAL_Delay(45);
  lcd_debug = 210;
}
void LCD_Init(void) {
            lcd_debug=1;
    HAL_Delay(100);  // Wait for LCD to power up

    // Reset sequence
    HAL_GPIO_WritePin(LCD_RS_GPIO_Port, LCD_RS_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(LCD_EN_GPIO_Port, LCD_EN_Pin, GPIO_PIN_RESET);

    // Initialization sequence
    LCD_Send4Bits(0x03);
    LCD_Enable();
    HAL_Delay(5);
    lcd_debug=2;

    LCD_Send4Bits(0x03);
    LCD_Enable();
    HAL_Delay(1);


    LCD_Send4Bits(0x03);
    LCD_Enable();
    HAL_Delay(1);


    // Set to 4-bit mode
    LCD_Send4Bits(0x02);
    LCD_Enable();
    HAL_Delay(1);
    lcd_debug=4;
    // Function set: 4-bit, 2-line, 5x8 dots
    LCD_SendCmd(0x28);
    HAL_Delay(1);
    lcd_debug=5;
    // Display control
    LCD_SendCmd(0x0C);  // Display ON, Cursor OFF, Blink OFF
    HAL_Delay(1);
    lcd_debug=25;
    // Clear display
    LCD_SendCmd(0x01);
    HAL_Delay(3);
    lcd_debug=67;
    // Entry mode set
    LCD_SendCmd(0x06);  // Increment, no shift
    HAL_Delay(1);
    lcd_debug=8;
}
void LCD_SendString(char *str)
{
            lcd_debug = 15670;
  while (*str)
  {
    LCD_SendData(*str++);
  }
}

void LCD_SetCursor(uint8_t row, uint8_t col)
{
            lcd_debug = 111100;
  uint8_t pos = (row == 0) ? 0x80 + col : 0xC0 + col;
  LCD_SendCmd(pos);
}
/* 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 */
    HAL_TIM_Base_Start(&htim2);
    HAL_GPIO_WritePin(DHT22_PORT, DHT22_PIN, 0);
    HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, 0);

    LCD_Init();
      LCD_SendString("Temperature:");
      HAL_Delay(1000);
      // After LCD_Init()
      LCD_SendCmd(0x01);  // Clear display
      HAL_Delay(2);
      LCD_SetCursor(0,0);
      LCD_SendString("TEST");
      LCD_SetCursor(1, 0);
      LCD_SendString("1234 ABCD");
      HAL_Delay(2000);  // Keep message visible
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
              HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, 1);
                              __HAL_TIM_SET_COUNTER(&htim2, 0);
                              while (__HAL_TIM_GET_COUNTER(&htim2) < 10);
                              HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, 0);

                              pMillis = HAL_GetTick();
                              while (!(HAL_GPIO_ReadPin(ECHO_PORT, ECHO_PIN)) && pMillis + 10 > HAL_GetTick());
                              val1 = __HAL_TIM_GET_COUNTER(&htim2);
```

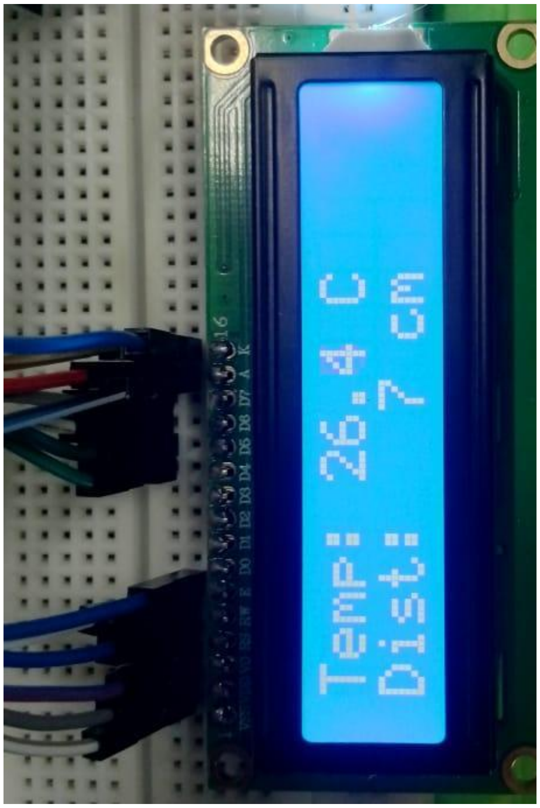
```
                pMillis = HAL_GetTick();
                while ((HAL_GPIO_ReadPin(ECHO_PORT, ECHO_PIN)) && pMillis + 50 >
HAL_GetTick());
                val2 = __HAL_TIM_GET_COUNTER(&htim2);

                distance = (val2 - val1) * 0.034 / 2;

                // LED based on distance
                if (distance < 15.0)
                    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET); // LED
OFF
                else
                    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);   // LED
ON

                // Display distance on LCD Line 1
                LCD_SetCursor(1, 0);
                sprintf(lcdBuffer, "Dist: %3d cm ", (int)distance);
                LCD_SendString(lcdBuffer);


                ////////////////// DHT22 TEMPERATURE //////////////////
                if (DHT22_Start())
                {
                    RH1 = DHT22_Read();
                    RH2 = DHT22_Read();
                    TC1 = DHT22_Read();
                    TC2 = DHT22_Read();
                    SUM = DHT22_Read();
                    CHECK = RH1 + RH2 + TC1 + TC2;

                    if (CHECK == SUM)
                    {
                        if (TC1 > 127)
                            tCelsius = (float)TC2 / 10 * -1;
                        else
                            tCelsius = (float)((TC1 << 8) | TC2) / 10;

                        humidity = (float)((RH1 << 8) | RH2) / 10;
                        // LED based on temperature
                        if (tCelsius > 20.0)
//                          HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_SET);   //
LED ON
//                      else
//                                  HAL_GPIO_WritePin(LED_PORT, LED_PIN, GPIO_PIN_RESET); //
LED OFF

                                // Display temperature on LCD Line 0
                                LCD_SetCursor(0, 0);
                                sprintf(lcdBuffer, "Temp: %.1f C  ", tCelsius);
                                LCD_SendString(lcdBuffer);
//                              Display humidity on LCD Line 1
                                                                                   LCD_SetCursor(1, 0);
                                                                                   sprintf(lcdBuffer, "Hum: %.1f
%%  ", humidity);
                                                                                   LCD_SendString(lcdBuffer);
                                                                           // Update every 1s

                             }
                         }

                         HAL_Delay(1000);  // Update every 1s
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 168;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 83;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 4294967295;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* USER CODE BEGIN MX_GPIO_Init_1 */

  /* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1|GPIO_PIN_7, GPIO_PIN_RESET);
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_4|GPIO_PIN_5
                          |GPIO_PIN_6|GPIO_PIN_7, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);

  /*Configure GPIO pins : PA1 PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PA6 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PB0 PB1 PB4 PB5
                           PB6 PB7 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_4|GPIO_PIN_5
                          |GPIO_PIN_6|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : PD13 */
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /* USER CODE BEGIN MX_GPIO_Init_2 */

  /* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```



```
Temperature:26
Humidity:60
Distance:7
Temperature:26
Humidity:60
Distance:7
Temperature:26
Humidity:60
Distance:7
Temperature:26
Humidity:60
Distance:7
Temperature:26
```

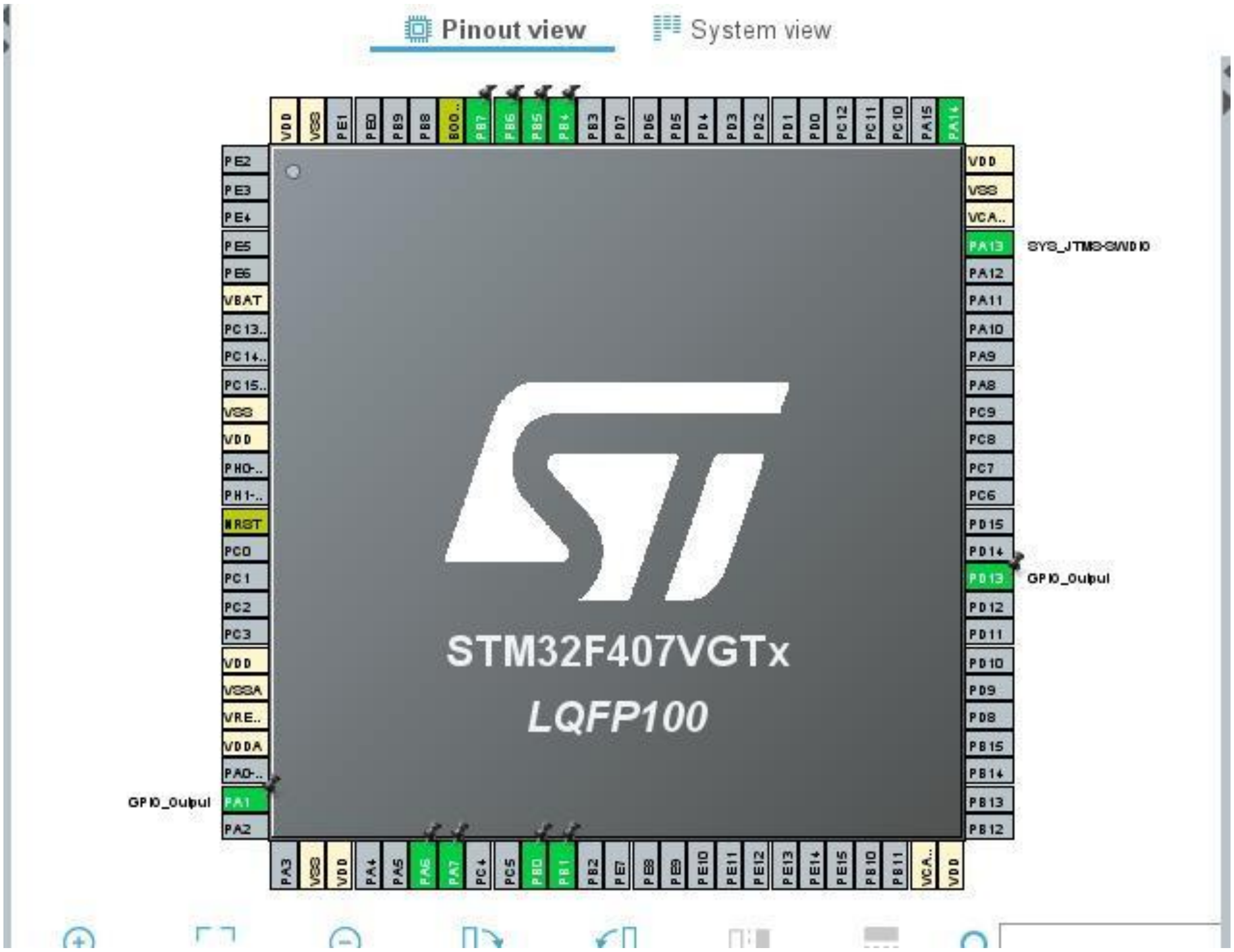
s 

```
Temp: 26.5 C
Hum: 60.5 %
```



```
Temp: 26.4 C
Dist: 7 cm
```

\



PA13: SYS_JTMS-SWDIO

PD13: GPIO_Output

PA1: GPIO_Output

G. References [Negative Marking of 20% if this section is skipped]

- Ultrasonic Sensor: (https://controllerstech.com/hcsr04-ultrasonic-sensor-and-stm32/) and (https://youtu.be/oD0mc6FXQ0s?si=g87e1WaUt_TsVT3E) and (https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf)
- 16*2 LCD Display: (https://www.elprocus.com/lcd-16x2-pin-configuration-and-its-working/) and (https://youtu.be/ITTBWSQTi3c?si=U5RwzGLfr3EJ500v) and (https://youtu.be/rfRJGfK2t-A?si=i9TNBS3xavfQLLZK)
- DHT22: (https://youtu.be/zuvvzTh4d4E?si=68WSt9Cx7oPCQbsc) and (https://controllerstech.com/temperature-measurement-using-dht22-in-stm32/) and (https://youtu.be/eA2ED7DNc5U?si=WPDYdh9EMGzRsAm7) and (https://microcontrollerslab.com/dht22-stm32-blue-pill-stm32cubeide/)

- Blue Tooth Module: (https://youtu.be/RPU2wzfewY8?si=rsDWJ0lL4AgkchVc) and (https://controllerstech.com/stm32-communication-using-hc-05/) and (https://deepbluembedded.com/stm32-hc-05-bluetooth-module-examples/)

  (https://youtu.be/9geREeE13jc?si=prRAF_Hh40YrStH2)
  (https://youtu.be/Iuxl0k4lnqA?si=bpov7WDYZ7rCMZj6)