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.								

Water Level Monitoring & Control

1: Ahmad Waleed Akhtar BSCE22003

2: Muhammad Arham BSCE22007

A. Overview [CLO-3, 50 Marks]

Effective water management is crucial in various sectors, from agriculture to industrial processes and domestic use. Monitoring and controlling water levels accurately is essential for sustainability and efficiency. This project focuses on developing a Water Level Monitoring and Control system using STM microcontroller technology. By integrating sensors, actuators, and microcontroller logic, the system enables real-time monitoring, precise control, and data analysis of water levels in reservoirs and tanks. Through automation and user-friendly interfaces, this system aims to improve water resource management, optimize usage, and prevent wastage.

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

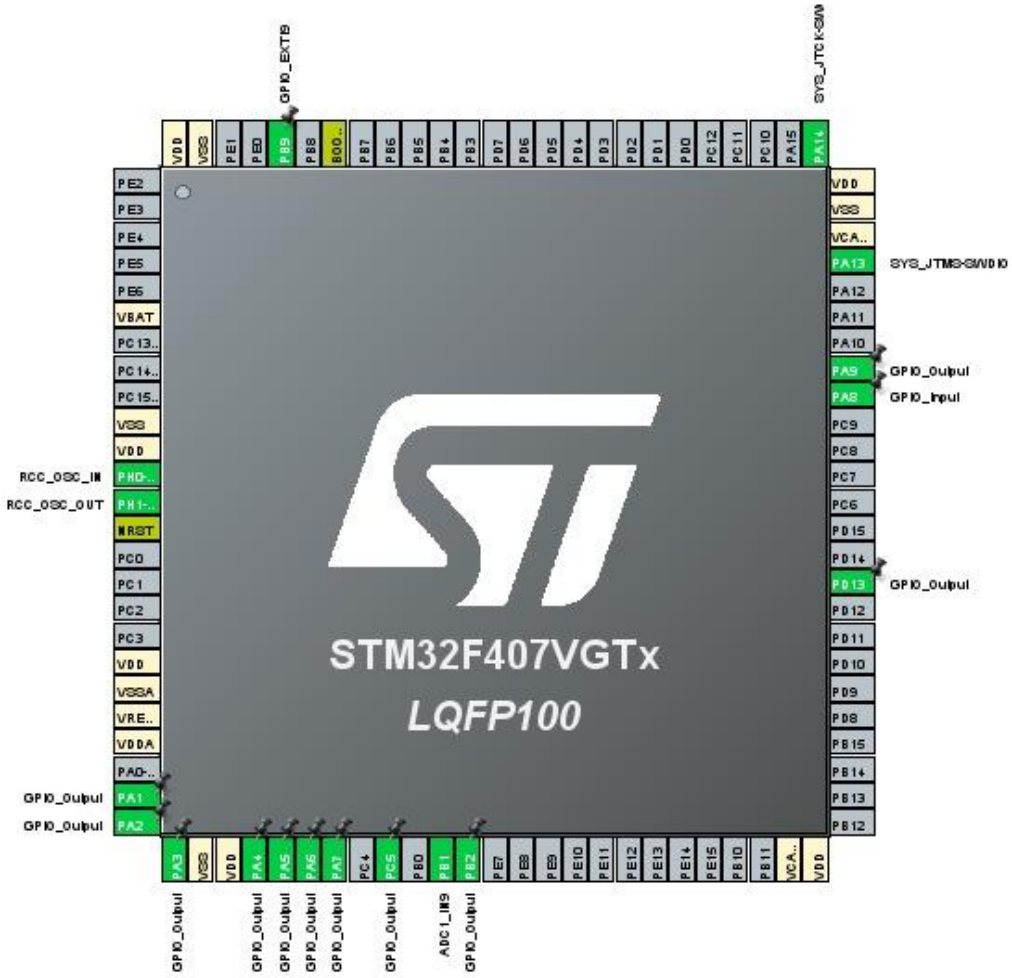
Sr#	Component
1	STM32F407VG
2	DHT22 (Moisture & Temperature)
3	5V Relay
4	Buzzer
5	16x2 Display
6	Potentiometer
7	FC28 (Soil Moisture Module)

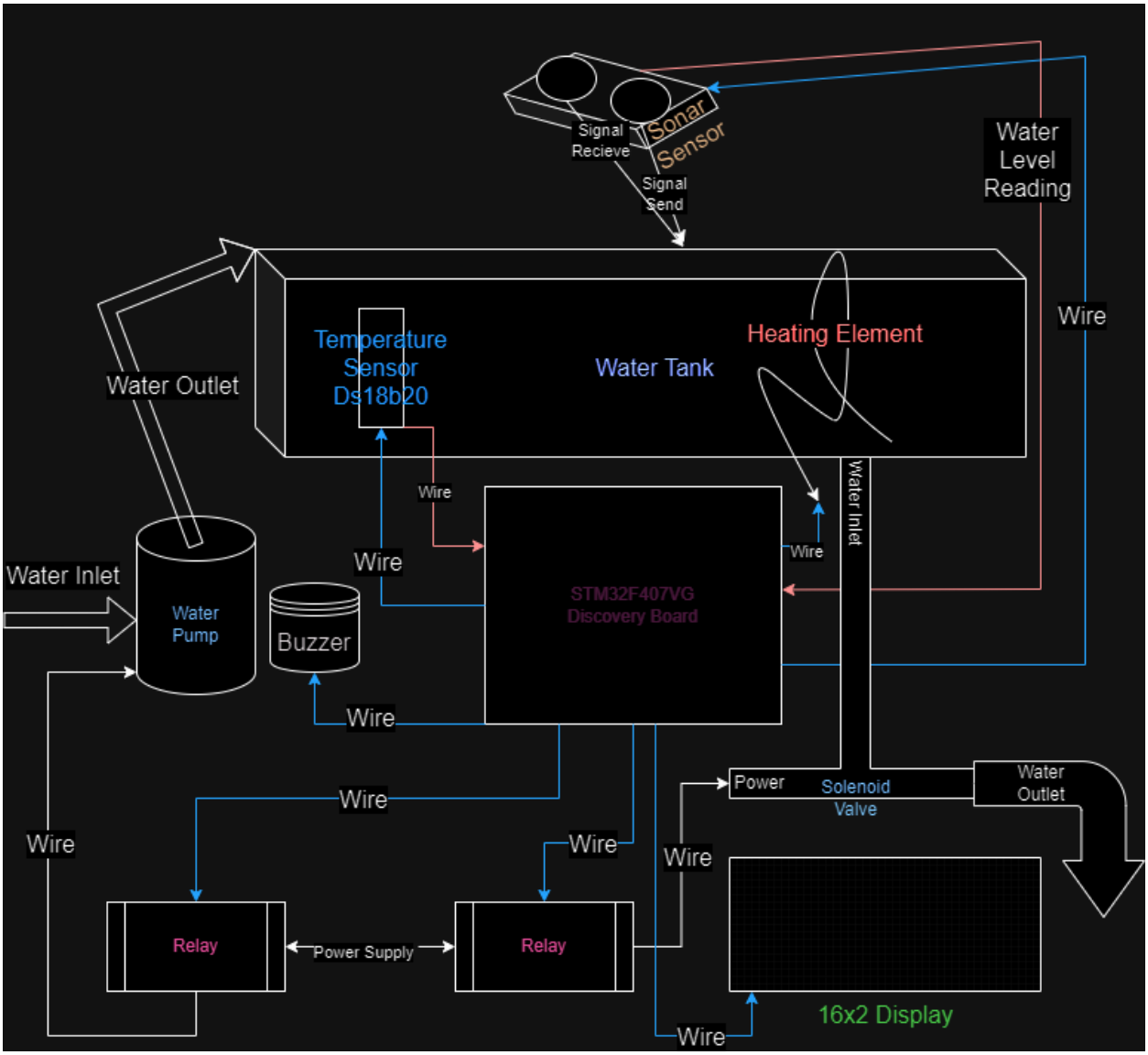
Sr#	Component	Reason
1	Black Pill (STM32F411CEU6) **(DISCARDED)**	The MC was not working properly. Pin mapping was different then the datasheet.
2	Ds18B20 Temperature Sensor **(DISCARDED)**	Shortage of time.
3	Heating Element **(DISCARDED)**	Didn't received it on time. Still is in custom's department.
4	Solenoid Valve UD-08 **(DISCARDED)**	Shortage of time.
5	Sonar Sensor HC-SR04 **(DISCARDED)**	Couldn't integrate it with other components. Is working separately.

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

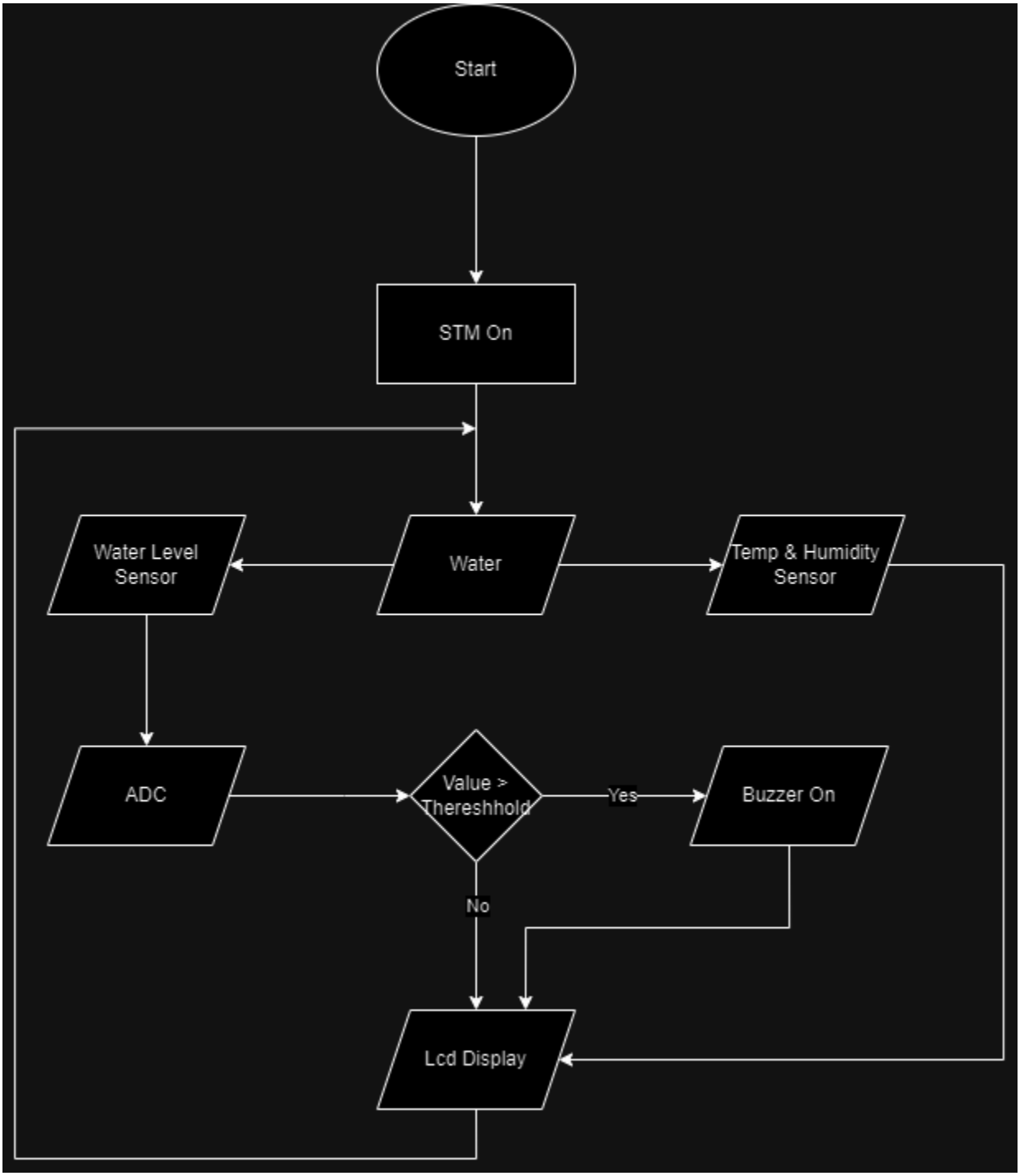
1. GPIO Pins: Used for interfacing with various components such as the water level sensor, LCD display, potentiometer, buzzer, solenoid valve, and heating element.
2. ADC (Analog-to-Digital Converter): Utilized to read the analog voltage value from the potentiometer for configuring the water level threshold.

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




Signal Recieve
Sonar Sensor
Signal Send
Water Level Reading
Wire
Temperature Sensor Ds18b20
Water Tank
Heating Element
Water Outlet
Wire
Water Inlet
Wire
Wire
STM32F407VG Discovery Board
Water Inlet
Water Pump
Buzzer
Wire
Wire
Power
Solenoid Valve
Water Outlet
Wire
Wire
Wire
Relay
Power Supply
Relay
16x2 Display
Wire

E. Flow Chart (Required at the time of final submission) [CLO-3, 50 Marks]



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

Attribute	Description	Complexity Level in your project
WP1: Depth of knowledge	The project shall involve in-depth engineering knowledge related to the area of Microprocessors, Microcontrollers & Interfacing [WK-4, Engineering Specialization].	Interfacing Techniques: Methods to interface microcontrollers with sensors, actuators, and peripherals using protocols like I2C, SPI, and UART. Sensor Interfacing: Analog and digital sensor interfaces, signal conditioning, and techniques for accurate sensor readings. Peripheral Utilization: Configuring and controlling GPIOs, timers, UART, and ADC to interface with various components
WP2: Range of conflicting requirements	The project has multiple conflicting requirements in terms of optimal usage of peripheral resources available on a Microcontroller.	• Limited number of I/O lines, requiring careful allocation for interfacing with various components. • Conflicting timing requirements between different peripherals, such as LCD updates, sensor readings, and control operations.
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.	• Synchronizing multiple tasks, such as sensor readings, LCD updates, and control operations, within a real-time framework.
WP7 Interdependence	The projects shall have multiple components at the hardware and software level.	• Proper integration of sensors, LCD, Potentiometer and buzzer with the microcontroller, ensuring correct wiring, signal conditioning, and compatibility. • Coordinating communication and control between multiple components while optimizing the utilization of microcontroller peripherals.

G. Code [CLO-3, 50 Marks]

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 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.
  *
  ******************************************************************************
  ******************************************************************************
  */
#include "main.h"
#include "../../ECUAL/LCD16X2/LCD16X2.h"
#include "stm32f4xx.h"
#include <stdio.h>

#define ARM_MATH_CM4

#define MyLCD LCD16X2_1
/* 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 */
```

```c
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;

TIM_HandleTypeDef htim1;

/* USER CODE BEGIN PV */
#define TRIG_PIN GPIO_PIN_9
#define TRIG_PORT GPIOA
#define ECHO_PIN GPIO_PIN_8
#define ECHO_PORT GPIOA
uint32_t pMillis;
uint32_t Value1 = 0;
uint32_t Value2 = 0;
uint16_t Distance  = 0;  // cm
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
static void MX_ADC1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
float readValue;
#define DHT11_PORT GPIOB
#define DHT11_PIN GPIO_PIN_9
uint8_t RHI, RHD, TCI, TCD, SUM;
uint32_t pMillis, cMillis;
float tCelsius = 0;
float tFahrenheit = 0;
float RH = 0;

void microDelay (uint16_t delay)
{
```

```
  __HAL_TIM_SET_COUNTER(&htim1, 0);
  while (__HAL_TIM_GET_COUNTER(&htim1) < delay);
}

uint8_t DHT11_Start (void)
{
  uint8_t Response = 0;
  GPIO_InitTypeDef GPIO_InitStructPrivate = {0};
  GPIO_InitStructPrivate.Pin = DHT11_PIN;
  GPIO_InitStructPrivate.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStructPrivate.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStructPrivate.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DHT11_PORT, &GPIO_InitStructPrivate); // set the pin as output
  HAL_GPIO_WritePin (DHT11_PORT, DHT11_PIN, 0);   // pull the pin low
  HAL_Delay(20);   // wait for 20ms
  HAL_GPIO_WritePin (DHT11_PORT, DHT11_PIN, 1);   // pull the pin high
  microDelay (30);   // wait for 30us
  GPIO_InitStructPrivate.Mode = GPIO_MODE_INPUT;
  GPIO_InitStructPrivate.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(DHT11_PORT, &GPIO_InitStructPrivate); // set the pin as input
  microDelay (40);
  if (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)))
  {
    microDelay (80);
    if ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN))) Response = 1;
  }
  pMillis = HAL_GetTick();
  cMillis = HAL_GetTick();
  while ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
  {
    cMillis = HAL_GetTick();
  }
  return Response;
}

uint8_t DHT11_Read (void)
{
  uint8_t a,b;
  for (a=0;a<8;a++)
  {
    pMillis = HAL_GetTick();
    cMillis = HAL_GetTick();
    while (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
    {  // wait for the pin to go high
      cMillis = HAL_GetTick();
```

```
    }
    microDelay (40);   // wait for 40 us
    if (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)))   // if the pin is low
      b&= ~(1<<(7-a));
    else
      b|= (1<<(7-a));
    pMillis = HAL_GetTick();
    cMillis = HAL_GetTick();
    while ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
    {  // wait for the pin to go low
      cMillis = HAL_GetTick();
    }
  }
  return b;
}

void GPIO_Init() {
    // Enable GPIOB clock
    RCC->AHB1ENR |= (1 << 1);

    // Configure PB10 (Trig pin) as output
    GPIOB->MODER |= (1 << 20);

    // Configure PB11 (Echo pin) as input
    GPIOB->MODER &= ~(3 << 22);
}

void TIM2_us_Delay(uint32_t delay) {
    RCC->APB1ENR |= 1; // Start the clock for the timer peripheral
    TIM2->ARR = (int)(delay / 0.0625); // Total period of the timer
    TIM2->CNT = 0;
    TIM2->CR1 |= 1; // Start the Timer
    while (!(TIM2->SR & TIM_SR_UIF)) {} // Polling the update interrupt flag
    TIM2->SR &= ~(0x0001); // Reset the update interrupt flag
}

void Moisture_Sensor()
{


          char hello_strr[20];
           HAL_ADC_PollForConversion(&hadc1,100);
           readValue = HAL_ADC_GetValue(&hadc1);
           snprintf(hello_strr, sizeof(hello_strr), "%ff",readValue);
           LCD16X2_Clear(MyLCD);
```

```
          LCD16X2_Set_Cursor(MyLCD, 1, 1);
            LCD16X2_Write_String(MyLCD, " Moisture Level");
             LCD16X2_Set_Cursor(MyLCD, 2, 1);
            LCD16X2_Write_String(MyLCD,hello_strr);
             HAL_Delay(2000);
            LCD16X2_Clear(MyLCD);

            if(readValue<=2300)
            {
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_SET);
            }

            else
            {

                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_5,GPIO_PIN_RESET);

            }

            if(readValue>=2240)
            {

                HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_SET);
            }


            else
            {
                HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_RESET);

            }

        if (readValue > 2300) {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 1);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 0);
        }
        else if (readValue > 1800 ) {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 1);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 0);
        }
        else if (readValue > 1500) {
```

```
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 1);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 0);
        }
        else {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 1);
        }




}



/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);


    uint32_t data;
    double time;
    int dist;
    RCC->CFGR |= 0 << 10; // set APB1 = 16 MHz
        GPIO_Init();
        GPIOB->BSRR = 0x00000000; // Setting tr
  /* 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_TIM1_Init();
  MX_ADC1_Init();
  /* USER CODE BEGIN 2 */
  HAL_ADC_Start(&hadc1);
  HAL_TIM_Base_Start(&htim1);
  HAL_TIM_Base_Start(&htim1);
  HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET);  // pull the TRIG pin
low
  char hello_str[20]; // Assuming the maximum length of the number won't exceed
10 characters
  char hello_ster[20];
     // Initialize an integer variable with the value 32

     // Convert the numerical value to a string
  LCD16X2_Init(MyLCD);
              LCD16X2_Clear(MyLCD);
  /* USER CODE END 2 */

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

         data = 10;
          Moisture_Sensor();

      if(DHT11_Start())
         {
           RHI = DHT11_Read(); // Relative humidity integral
           RHD = DHT11_Read(); // Relative humidity decimal
           TCI = DHT11_Read(); // Celsius integral
           TCD = DHT11_Read(); // Celsius decimal
```

```
           SUM = DHT11_Read(); // Check sum
           if (RHI + RHD + TCI + TCD == SUM)
           {
             // Can use RHI and TCI for any purposes if whole number only needed
             tCelsius = (float)TCI + (float)(TCD/10.0);
             tFahrenheit = (tCelsius * 9/5 + 32)-20;
             RH = (float)RHI + (float)(RHD/10.0);
             // Can use tCelsius, tFahrenheit and RH for any purposes
           }

         }
      snprintf(hello_str, sizeof(hello_str), "%f",tFahrenheit);
      snprintf(hello_ster, sizeof(hello_ster), "%f",tCelsius);

//
//          HAL_Delay(2000);
//
//         // 1. Sending 10us pulse to trig pin (PB10)
//         GPIOB->BSRR &= ~(1 << 10); // PB10 is low
//         TIM2_us_Delay(2);
//         GPIOB->BSRR |= (1 << 10); // PB10 set to High
//         TIM2_us_Delay(10); // wait for 10us
//         GPIOB->BSRR |= (1 << 26); // Make PB10 low again
//
//         // 2. Measure the pulse width of the pulse sent from the echo pin
(PB11) by polling IDR for port B
//         while (GPIOB->IDR & (1 << 11)) {
//             data++;
//         }
//
//         // 3. Converting the gathered data into distance in cm
//         if (data > 0) {
//             time = data * (0.0625 * 0.000001);
//             dist = ((time * 340) / 2) * 100;
//         }
//

            // Print to LCD screen

                                LCD16X2_Set_Cursor(MyLCD, 1, 1);
                                 LCD16X2_Write_String(MyLCD, " WELCOME WATER");
                                 LCD16X2_Set_Cursor(MyLCD, 2, 1);
                                 LCD16X2_Write_String(MyLCD,"MANAGMENT
PROJECT");
                                 HAL_Delay(2000);
```

```
                                LCD16X2_Clear(MyLCD);



            LCD16X2_Set_Cursor(MyLCD, 1, 1);
            LCD16X2_Write_String(MyLCD, " Temperature");
            LCD16X2_Set_Cursor(MyLCD, 2, 1);
            LCD16X2_Write_String(MyLCD, hello_str);
            HAL_Delay(1000);
             LCD16X2_Clear(MyLCD);
            LCD16X2_Set_Cursor(MyLCD, 1, 1);
               LCD16X2_Write_String(MyLCD, " Humidity");
                    LCD16X2_Set_Cursor(MyLCD, 2, 1);
                    LCD16X2_Write_String(MyLCD, hello_ster);
            HAL_Delay(1000);
             LCD16X2_Clear(MyLCD);







    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

  }
  /* 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);
```

```c
  /** 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 = 72;
  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_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};
```

```
  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */

  /** Configure the global features of the ADC (Clock, Resolution, Data Alignment
and number of conversion)
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure for the selected ADC regular channel its corresponding rank in
the sequencer and its sample time.
  */
  sConfig.Channel = ADC_CHANNEL_9;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
```

```
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

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

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 71;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 65535;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_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_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_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_2|GPIO_PIN_3|GPIO_PIN_4
                          |GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_9,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_SET);

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

  /*Configure GPIO pins : PA1 PA2 PA3 PA4
                           PA5 PA6 PA7 PA9 */
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4
                          |GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_9;
  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 : PC5 */
  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
  /*Configure GPIO pin : PB2 */
  GPIO_InitStruct.Pin = GPIO_PIN_2;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  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_VERY_HIGH;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : PA8 */
  GPIO_InitStruct.Pin = GPIO_PIN_8;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PB9 */
  GPIO_InitStruct.Pin = GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOB, &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 */

  /* USER CODE END Error_Handler_Debug */
}
```

```c
#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,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

H. References

- Authors, S. W. (2023, March 1). Introduction to STM32CubeIDE. Retrieved from STM32 Arm® Cortex® MCU Wiki: Link
- Ruturaj N. "Interfacing an Ultrasonic Sensor with the STM32F407 Discovery Board Using Bare Metal Programming." Hashnode, 2023. [Online]: Link
- Controllerstech, "STM32 UART (7) - 1 Wire Protocol," 2023. [Online]: Link
- Controllerstech, "Temperature Measurement using DHT22 in STM32," 2023. [Online]: Link