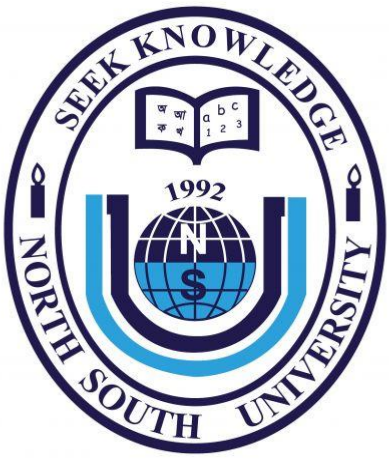

PROJECT REPORT ON TEMPERTAURE AND HUMIDITY MONITORING SYSTEM WITH STM32 MICROCONTROLLER

SUBMITTED TO:

Mohammad Abdul Qayum
Assistant Professor

SUBMITTED BY:

GROUP MEMBERS:

Name	Id
Iftekhar Mahbub Rafi	2021463642
Sagor Mahmud Opu	2021162642
Tasfia Tahmid Ushno	1831099642
Tanzila Rahman Rimi	2011225042

Table of Contents

Table of Figures

INTRODUCTION:

The intensity of light, temperature and humidity, and CO gas concentration in indoor environment become several important elements that affect human physical and mental health, and their changes will also counteract human mood and emotion. Especially, high CO gas concentration in indoor air can seriously affect human life. There are many ways to make the real-time monitoring system for indoor environment, but the function of display and alarm of the monitoring system is relatively single. Therefore, it is a key problem to solve in the design and production to realize multiple display and alarm functions of real-time in-door environment detection information based on single chip microcomputer, especially OneNET platform display and alarm or mobile phone display and alarm. This is also to provide the basis for intelligent management of living quarters. This designed indoor detection system can effectively detect the current indoor environment, and through real-time understanding of the indoor environment, it can prevent the occurrence of a series of situations that are harmful to human safety, such as excessive indoor temperature and humidity, CO gas poisoning, etc., and provide OneNET platform display, instrument display and other information display. With the development of modern technology and the improvement of people's living standards, home intelligence is becoming more and more common, and in the modern era of Internet of everything, the application of indoor environment intelligent detection system will enter every household.

TOOL DESCRIPTION:

1.CODING PLATFORM:

STM32CubeIDE - STM32 is a family of 32-bit microcontroller integrated circuits by STMicroelectronics. The STM32 chips are grouped into related series that are based around the same 32-bit ARM processor core: Cortex-M0, Cortex-M0+, Cortex-M3, Cortex-M4, Cortex-M7, Cortex-M33.

2.HARDWARE MATERIAL:

To build this setup we needed the following hardware:

1. STM32F103C8T6 Blue Pill Development Board -
2. ST-link V2 STM8 STM32 Simulator Download Programmer
3. 16*2 Serial LCD module display for Arduino Assembled

4. DHT11 Temperature and Relative Humidity sensor module for Arduino
5. Blue LED 5mm
6. Passive Buzzer
7. 4*4 Keypad – 16key- Matrix Membrane Type
8. Ultrasonic sonar sensor HC-SR04
9. Push button switch 4 pin (12mm*12mm)
10. Male to male jumper wires 20pcs 20cm
11. 3inch 12c OLED display module 4-pin – Blue

METHODOLOGY

Project Planning

The primary objective was to develop a system that monitors and displays temperature and humidity. Additional functionalities include recording maximum and minimum values and user interaction via a keypad. The initial concept was reviewed and approved by the lab instructor to ensure it met the course requirements.

System Design

A detailed flowchart was created to outline the operation of the system. This included the initialization process, sensor data acquisition, data processing, and display updates. Developed a concept map to visualize the interconnections between various components and their interactions.

Tool Setup & Initial Testing

A tool was installed to configure the STM32 microcontroller and initialize its peripherals. The 4x4 keypad and 16x2 LCD were connected to the STM32F103C8T6 microcontroller. Basic firmware was written to test the LCD and keypad functionalities, ensuring that keypad inputs were correctly read and displayed on the LCD.

Hardware-Software Integration

Integrate the hardware with the developed firmware. The DHT11 humidity and temperature sensor was connected to the microcontroller. Firmware was developed. This included initializing the sensor's settings, reading its data, and translating the results to a human-readable format.

Full System Integration

Other sensors, such as an ultrasonic sonar sensor, were optionally included to improve functionality. Firmware was created to handle data from all sensors and peripherals. This

includes reading sensor data, data processing involves identifying maximum and minimum values, handling user input through the keypad for functions such as resetting recorded data.

Testing

The complete system was rigorously tested to ensure all components functioned as expected. Checking the accuracy of temperature and humidity readings against known standard Sensor readings were compared with actual ambient conditions to validate accuracy.

Documentation and Presentation

A PowerPoint presentation was created to summarize the project's objectives, design, and results. A detailed project report was compiled, documenting the design process, methodologies, and outcomes. A live demonstration of the working embedded system was prepared for evaluation.

PROJECT DIAGRAMS AND PHYSICAL DESIGN

A. PROJECT DIAGRAM:

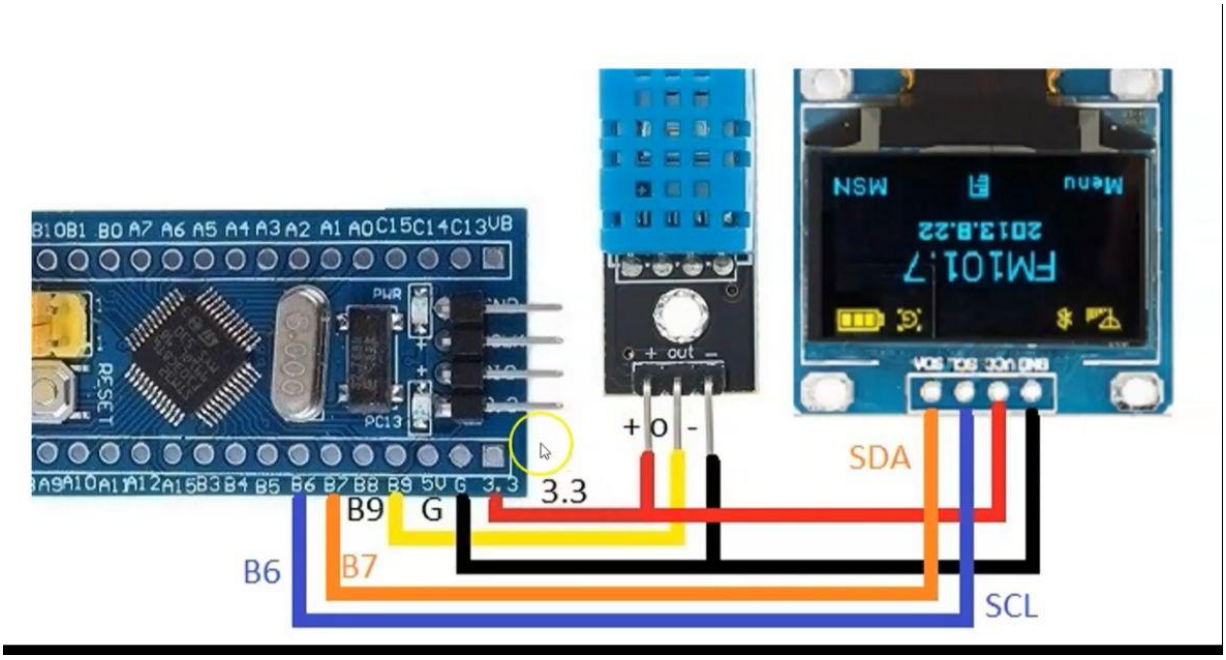

Figure 1: Circuit Diagram of the Project

B. PHYSICAL DESIGN:

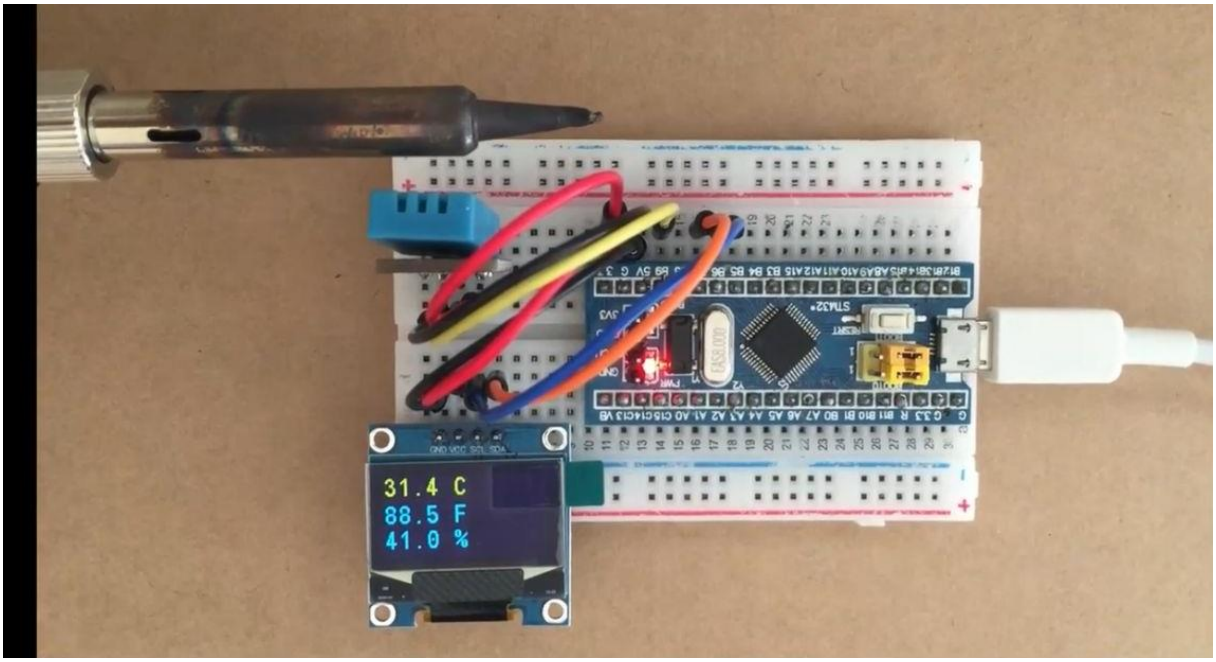

Figure 2: Physical Design of the Project.

SIMULATION RESULTS

The system was simulated using STMicroelectronics tools and validated with real-time data collecting. The DHT11 sensor gave accurate temperature and humidity data, which were shown on the 16x2 LCD. The system successfully logged the maximum and minimum values during its operation. The system could also recognize and respond to human input via the keypad.

Simulation Code

```
/* 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.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "fonts.h"
#include "ssd1306.h"
#include "stdio.h"
/* 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 ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;

TIM_HandleTypeDef htim1;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_TIM1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#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;
uint8_t TFI = 0;
uint8_t TFD = 0;
char strCopy[15];

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;
}
/* 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_I2C1_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start(&htim1);
  SSD1306_Init();
  /* USER CODE END 2 */

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


              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;
                RH = (float)RHI + (float)(RHD/10.0);
                // Can use tCelsius, tFahrenheit and RH for any purposes
                TFI = tFahrenheit;  // Fahrenheit integral
                TFD = tFahrenheit*10-TFI*10; // Fahrenheit decimal
                sprintf(strCopy,"%d.%d C   ", TCI, TCD);
                SSD1306_GotoXY (0, 0);
                SSD1306_Puts (strCopy, &Font_11x18, 1);
                sprintf(strCopy,"%d.%d F   ", TFI, TFD);
                SSD1306_GotoXY (0, 20);
                SSD1306_Puts (strCopy, &Font_11x18, 1);
                sprintf(strCopy,"%d.%d %%  ", RHI, RHD);
                SSD1306_GotoXY (0, 40);
                SSD1306_Puts (strCopy, &Font_11x18, 1);
                SSD1306_UpdateScreen();

                if(tCelsius>29.1){
                    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_SET);
                    HAL_Delay(500);
                    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_RESET);
                    HAL_Delay(500);
                }
              }
            }
            HAL_Delay(2000);
  /* 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};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  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_DIV1;

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

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */

  /* USER CODE END I2C1_Init 1 */
```

```
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 400000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_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_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_9, GPIO_PIN_RESET);

  /*Configure GPIO pins : PB3 PB9 */
  GPIO_InitStruct.Pin = GPIO_PIN_3|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(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 */
 __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 */
```


ACKNOWLEDGEMENT

I would like to thank Dr. Mohammad Abdul Qayum for his guidance and support during this project. His knowledge and assistance were vital in the effective execution of this project.

REFERENCES:

[1] Gu, X., Mo, H. and Liu, S. (2021) Design of Control System of Cleaning Machine for Exterior Wall of High-Rises Based on STM32. International Core Journal of Engineering, 7, 207-210.

[2] Yang, F. and Wei, Q. (2021) Design of Smart Home Control System of Internet of

Things Based on STM32. Journal of Physics: Conference Series, No. 1, 1-5.

[3] "Getting Started with STM32: Introduction to STM32CubeMX and STM32CubeIDE," Digi-Key Electronics, Jul. 16, 2020. [Online]. Available: https://www.digikey.com/en/articles/getting-started-with-stm32-introduction-to-stm32cubemx-and-stm32cubeide

[4] "I2C Protocol Overview and Use in Embedded Systems," NXP Semiconductors, Jun. 10, 2019. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN10216.pdf

[5] "Interfacing Temperature Sensors with ARM Microcontrollers," Texas Instruments, Dec. 12, 2019. [Online]. Available: https://www.ti.com/lit/an/sboa202a/sboa202a.pdf