**MEASURING TEMPERATURE AND HUMIDITY WITH HTU21D SENSOR AND STM32 NUCLEO STM32F410**

**INTRODUCTION**

This report outlines the process of interfacing an HTU21D sensor with an STM32 Nucleo STM32F410 development board to measure temperature and humidity. The acquired data will be displayed in Tera Term it is widely used serial terminal emulator.

**HTU21D SENSOR**

The HTU21D sensor is a digital humidity and temperature sensor that provides accurate and reliable environmental data in various applications. Below is some technical content about the HTU21D sensor:

**SENSOR SPECIFICATION**

**Measurement Range:**

* Humidity: 0% to 100% RH (Relative Humidity).
* Temperature: -40°C to 125°C.

**Resolution:**

* Humidity: 0.04% RH.
* Temperature: 0.01°C.

**Accuracy:**

* Humidity: ±2% RH.
* Temperature: ±0.3°C.

**Sensing Element:**

The HTU21D sensor employs a capacitive-type sensing element for measuring humidity and a band-gap temperature sensor for temperature measurements.

**Communication Protocol:**

The sensor communicates with microcontrollers and other digital devices using the I2C (Inter-Integrated Circuit) protocol. It has a 7-bit I2C address that can be configured to allow multiple sensors on the same bus.

**Operating Voltage:**

The HTU21D sensor typically operates at 3.3V, making it suitable for use with microcontrollers and systems operating at 3.3V logic levels.

**Power Consumption:**

The sensor is designed with low power consumption in mind, making it suitable for battery-powered or energy-efficient devices.

**Calibration:**

The HTU21D sensor is factory-calibrated, reducing the need for user calibration. This ensures accurate and reliable measurements.

**Response Time:**

The sensor offers a fast response time, allowing for real-time monitoring of humidity and temperature changes.

**Physical Package:**

The sensor is typically available in compact packages suitable for various applications. Common packages include surface-mount devices (SMD) with exposed sensor elements for accurate measurements.

**Data Output:**

The sensor provides digital output data, making it easy to interface with microcontrollers and digital systems. The humidity and temperature values are typically transmitted in a binary format.

**HARDWARE** **SETUP:**

**Components** **Used:**

* STM32 Nucleo STM32F410 development board
* HTU21D digital humidity and temperature sensor
* Jumper wires

**Hardware** **Connections:**

The HTU21D sensor was connected to the STM32 Nucleo board as follows:

VCC: The VCC pin of the HTU21D sensor was connected to a 3.3V power supply on the Nucleo board.

GND: The GND pin of the HTU21D sensor was connected to the ground (GND) on the Nucleo board.

SDA: The SDA (data) pin of the HTU21D sensor was connected to the I2C data pin on the Nucleo board.

SCL: The SCL (clock) pin of the HTU21D sensor was connected to the I2C clock pin on the Nucleo board.

**SOFTWARE DEVELOPMENT:**

**Development Environment:**

STM32CubeIDE was used for code development. This integrated development environment facilitates microcontroller configuration and code generation.

**Configuration in STM32CubeMX:**

The following steps were taken in STM32CubeMX to configure the Nucleo board for interfacing with the HTU21D sensor:

* A new project was created for the STM32F410 Nucleo board.
* The I2C communication peripheral was enabled, and the pins for SDA and SCL were properly configured.
* Clock settings for the I2C peripheral were adjusted to match the sensor's requirements.

**Code Implementation:**

The code was developed to read data from the HTU21D sensor using the I2C interface. Additionally, a UART (USART2) was configured to communicate with Tera Term for displaying the data. The code is as follows,

**Code:**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 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** "string.h"

**#include** "stdlib.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;

UART\_HandleTypeDef huart2;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_USART2\_UART\_Init**(**void**);

**static** **void** **MX\_I2C1\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* 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\_USART2\_UART\_Init();

MX\_I2C1\_Init();

/\* USER CODE BEGIN 2 \*/

uint8\_t dev\_addr = 64, return\_val, str[2] = "";

uint8\_t cmd = 0xE3;

uint8\_t buffer[2] = { 0 };

uint16\_t Stemp;

**float** Temperature;

return\_val = HAL\_I2C\_Master\_Transmit(&hi2c1, dev\_addr << 1, &cmd, 1, 1000);

**if** (return\_val == 0)

{

HAL\_I2C\_Master\_Receive(&hi2c1, dev\_addr << 1, buffer, 2, 1000);

// HAL\_I2C\_Master\_Receive(&hi2c1, dev\_addr<<1, buffer1, 2, 1000);

Stemp = ((buffer[0] << 8) | (buffer[1]));

Temperature = (-46.85 + 175.72 \* ((**float**) Stemp / 65536.0));

// Humidity = (-6 + 125 \* ((float)Srh / 65536.0));

**memset**(str, 0, **strlen**(str));

**sprintf**(str, "%f C\n", Temperature);

HAL\_UART\_Transmit(&huart2, str, **strlen**(str), 1000);

}

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* 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\_SCALE2);

/\*\* 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 = 16;

RCC\_OscInitStruct.PLL.PLLN = 336;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV4;

RCC\_OscInitStruct.PLL.PLLQ = 7;

**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 = 100000;

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 USART2 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USART2\_UART\_Init**(**void**)

{

/\* USER CODE BEGIN USART2\_Init 0 \*/

/\* USER CODE END USART2\_Init 0 \*/

/\* USER CODE BEGIN USART2\_Init 1 \*/

/\* USER CODE END USART2\_Init 1 \*/

huart2.Instance = USART2;

huart2.Init.BaudRate = 115200;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

huart2.Init.StopBits = UART\_STOPBITS\_1;

huart2.Init.Parity = UART\_PARITY\_NONE;

huart2.Init.Mode = UART\_MODE\_TX\_RX;

huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;

**if** (HAL\_UART\_Init(&huart2) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART2\_Init 2 \*/

/\* USER CODE END USART2\_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\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(LD2\_GPIO\_Port, LD2\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin : B1\_Pin \*/

GPIO\_InitStruct.Pin = B1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_FALLING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(B1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : LD2\_Pin \*/

GPIO\_InitStruct.Pin = LD2\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(LD2\_GPIO\_Port, &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 \*/

**TERA TERM CONFIGURATION:**

Tera Term was opened on a computer. A connection was established to the COM port to which the STM32 Nucleo board was connected. Baud rate, data bits, stop bits, and flow control settings were configured to match the UART settings in the code (e.g., 115200 baud rate).

**OBSERVATION:**

After flashing the code to the STM32 Nucleo board, temperature and humidity data were observed to be displayed using given formula in Tera Term in real-time. The data was read from the HTU21D sensor and sent over the UART connection to the computer.

**CONCLUSION:**

The successful interfacing of the HTU21D sensor with the STM32 Nucleo STM32F410 board, along with the real-time display of temperature and humidity data in Tera Term, demonstrates the effectiveness of this sensor integration. This setup provides a foundation for various applications, including environmental monitoring and data logging.