VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Samuel Silvola

Environmental Monitoring System For Greenhouses

Technology and Communication
2024

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Samuel Silvola
Title	Environmental Monitoring System for Greenhouses
Year	2024
Language	English
Pages	43
Name of Supervisor	Smail Menani

This project addresses the need for sustainable gardening by developing an integrated wireless environmental monitoring system for greenhouses. The aim is to optimize plant growth through real-time monitoring of temperature, humidity, and soil moisture levels.

The system incorporates a temperature and humidity sensor for measuring the air and a soil moisture sensor for plant hydration monitoring. The STM32 microcontroller board serves as the core component for connecting with the sensors. Additionally, the Raspberry Pi is utilized for data storage, visualization, and remote accessibility. The implementation uses USART for communication between the STM32 microcontroller and Raspberry Pi. Then the data is transmitted to a mobile device via Wi-Fi.

The results demonstrate the successful capture and transmission of real-time environmental data, providing a full view of greenhouse conditions. This integrated system offers a practical and efficient solution for informed decision making in greenhouse management.

Keywords Embedded C, Microcontroller, Raspberry Pi

CONTENTS

LIST OF ABBREVIATIONS

APB2ENR	Advanced Peripheral Bus 2 Enable Register
ADC	Analog-To-Digital Converter
ARR	Auto-Reload Register
CDT	C/C++ Development Toolkit
CR	Control Register
DR	Data Register
EOC	End of Conversion
GCC	GNU Compiler Collector
GDB	GNU Project Debugger
GND	Ground
GPIO	General Purpose Input/Output
IC	Integrated Circuit
IDE	Integrated Development Environment
IOT	Internet Of Things
MCU	Micro Controller Unit
MSB	Most Significant Bit
NTC	Negative Temperature Coefficient
OS	Operating System
PCB	Printed Circuit Board
RCC	Reset and Clock Control

RX	Receive
SWSTART	Software Start
TX	Transmit
TXE	Transmit Data Register Empty
UI	User Interface
USART	Universal Synchronous and Asynchronous Receiver-Transmitter

LIST OF FIGURES AND TABLES

1 INTRODUCTION

Environmental monitoring systems are essential in modern agriculture, particularly in greenhouses where precise control over environmental parameters is essential. This thesis aims to develop an automated greenhouse monitoring system to monitor key environmental factors such as soil moisture, air temperature, and air humidity, ensuring optimal plant growth.

By integrating sensors and developing the required software, the system will provide real-time data to improve greenhouse management. The goal is to enhance plant health and productivity by maintaining ideal environmental conditions. Manual monitoring of environmental parameters is labor-intensive and error-prone, leading to suboptimal growing conditions. This thesis addresses the problem by developing an automated system for continuous monitoring and access to accurate real-time data.

This project focuses strictly on monitoring environmental conditions within a greenhouse, without including automated control mechanisms like irrigation or ventilation systems. The scope is limited to sensor integration, data acquisition, and data communication.

2 THEORY AND BACKGROUND INFORMATION

In this chapter, the theory and background information are introduced.

2.1 Requirements

The greenhouse monitoring system must track the status of the greenhouse in real time, continuously collecting data on soil moisture, temperature, and humidity. It needs to accurately read sensor data from sensors and process this data for real-time monitoring. The system must also transmit the collected data to a central monitoring unit or display it on a user interface. A user-friendly interface is required for displaying the current environmental conditions.

The system must operate reliably under various environmental conditions, ensuring continuous monitoring without frequent failures. Sensor readings must be accurate and precise. The system must be easy to install, configure, and use, with minimal technical knowledge required. It should be scalable, allowing additional sensors or components to be added as needed. Additionally, the system must be power-efficient, suitable for long-term.

Input data for the system includes soil moisture levels from soil moisture sensors, temperature readings from temperature sensor, and humidity readings from humidity sensor. Output data includes the real-time display of soil moisture, temperature, and humidity on a user interface.

2.2 Hardware

The hardware design of the project consists of the following parts: the microcontroller board, a humidity and temperature sensor, a soil moisture sensor, a Raspberry Pi, and a 10K Ohm resistor to keep the data line high to enable the connection between the sensor and the board.

The STM32 microcontroller facilitated seamless interfacing with sensors, ensuring accurate data collection. The communication link between the microcontroller and Raspberry Pi enables real-time data transfer and remote accessibility. By establishing this connection, the system allowed monitoring the greenhouse environment from a mobile device, eliminating the need for physical presence in the garden.

2.3 Application Theory

This section includes theory of the measurements and the approach of obtaining them.

2.3.1 Measurements

The goal behind measurements is to collect relevant data from different sensors to analyze living conditions of the plants inside the greenhouse.

For measuring humidity, the DHT22 sensor uses a humidity sensing component which has two electrodes with moisture holding substrate between them. So, as the humidity changes, the conductivity of the substrate changes or the resistance between changes in these electrodes. This change in resistance is measured and processed by the IC which makes it ready to be read by the microcontroller. **/1/**

For measuring temperature, the DHT22 uses a NTC temperature sensor or a thermistor. A thermistor is a variable resistor that changes its resistance with the change of the temperature. The sensor is made by sintering of semiconductive materials such as ceramics or polymers to provide larger changes in the resistance with just small changes in temperature.

The SparkFun Soil Moisture Sensor is a simple breakout for measuring the moisture in soil and similar materials. It offers a reliable and cost-effective solution for monitoring soil moisture levels in this project.

2.3.3 Measurement Format

The DHT22 sensor provides temperature and humidity measurements in a digital format. Specifically, the sensor communicates with the microcontroller using a single-wire digital interface protocol. When MCU finishes sending the start signal, DHT22 will send response signal of 40-bit data that reflect the relative humidity and temperature information to MCU.

For temperature measurements, the DHT22 sensor outputs the temperature in degrees Celsius with a resolution of 0.1°C. The temperature data is represented as a 16-bit unsigned integer, where the MSB indicates the sign (positive or negative), and the remaining bits represent the temperature value.

For humidity measurements, the DHT22 sensor outputs the relative humidity as a percentage with a resolution of 0.1%. Like temperature, the humidity data is also represented as a 16-bit unsigned integer.

The SparkFun soil moisture sensor provides analog output based on the moisture level detected in the soil. The ADC converts the analog voltage into a digital value that can be processed and interpreted by the microcontroller.

The sensor measures the electrical conductivity or resistance of the soil, which varies with its moisture content. The output from the sensor is an analog voltage signal that corresponds to the soil moisture level. The voltage output varies proportionally with the moisture level in the soil: higher moisture levels result in higher analog voltage readings, while lower moisture levels outputs lower voltage readings.

Once the application is running, it begins measuring the output from the sensors. The data is stored inside the buffer structure of the STM32 microcontroller, and it can be viewed by the user through a terminal software running on the Raspberry Pi.

3 USED TECHNOLOGY

In this chapter, the software tools and the hardware components of the project are described.

3.1 Software Tools

The project was created using STM32CubeIDE. It is an all-in-one multi-OS development tool, which is part of the STM32Cube software eco-system. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors. It is based on the Eclipse®/CDT™ framework and GCC toolchain for the development, and GDB for the debugging. It allows the integration of the hundreds of existing plugins that complete the features of the Eclipse® IDE. /**2**/

Raspberry Pi OS was chosen as the operating system for the project for its versatility and for its user-friendly interface for the end user. It is a Debian-based Linux distribution tailored specifically for the Raspberry Pi hardware. Raspberry Pi OS serves as a versatile platform for a wide range of projects, including home automation, media centers, retro gaming consoles, IoT devices, and educational applications. It is the official operating system for the Raspberry Pi.

Screen was used as the terminal window software. With Screen temperature monitoring code can run in the background even if disconnected from the terminal. This ensures continuous monitoring of temperature levels without interruptions.

RealVNC Viewer was used in the project to check the data from the Raspberry Pi over the home network. RealVNC Viewer is a remote desktop software that allows users to access and control a computer from another device over a network connection.

3.2 Hardware Components

In this section of the thesis, the list of components with description will be presented. In selecting the hardware components for the project, careful consideration was given to factors such as functionality, compatibility, and reliability. Each component was chosen to contribute to the project's objectives. Key criteria included availability, sensor accuracy, communication protocols, and ease of interfacing with the microcontroller.

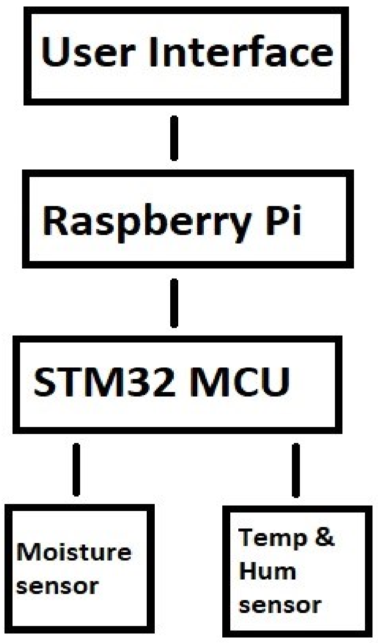


Figure 1. System Architecture Diagram

The STM32L152RE devices offer key features essential for low-power applications. They incorporate a high-performance Arm Cortex-M3 32-bit RISC core operating at 32 MHz, facilitating efficient processing. It includes a 12-bit ADC for analog-to-digital conversion, three USARTs for communication, and a variety of timers, including a general-purpose 32-bit timer and six 16-bit timers. They also feature a real-time clock for timekeeping. With Flash memory up to 512 Kbytes, these devices provide ample storage for program code and data. Operating within a voltage range of 1.8 to 3.6 V, they offer flexibility in power supply requirements and can operate in temperature ranges from -40 to +85 °C and -40 to +105 °C, making them suitable for diverse environmental conditions. /**3**/

STM32L152RE is the core of this thesis project.

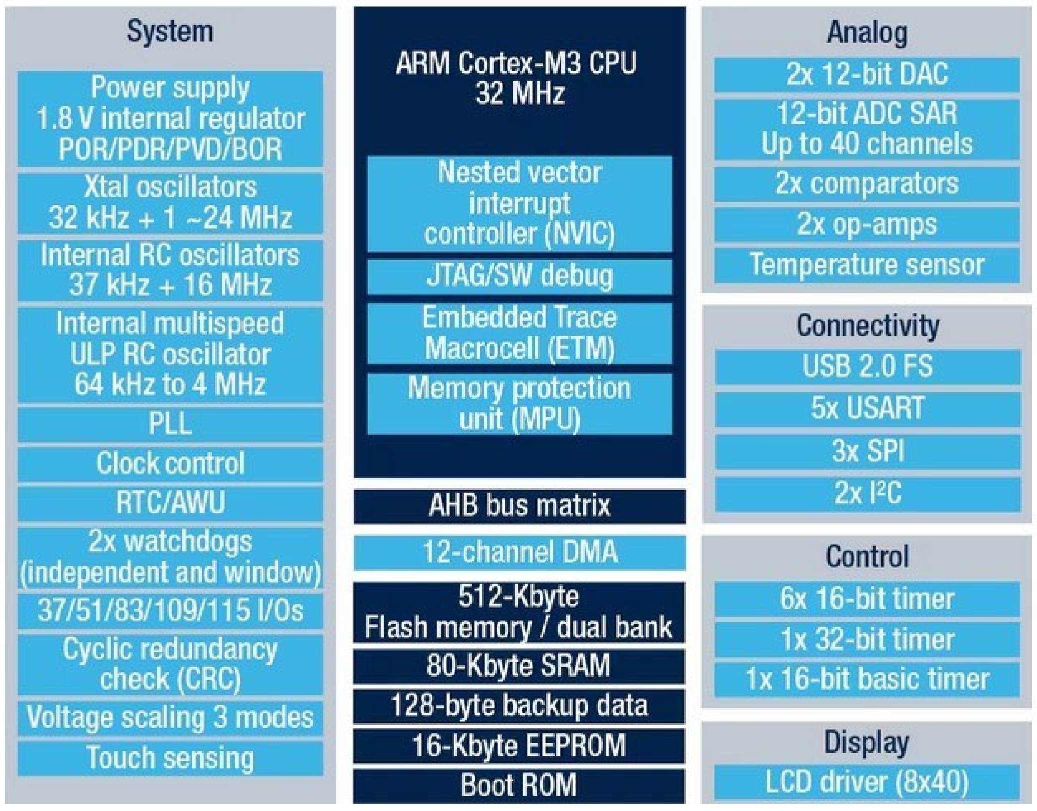

System
Power supply 1.8 V internal regulator POR/PDR/PVD/BOR
Xtal oscillators 32 kHz + 1 ~24 MHz
Internal RC oscillators 37 kHz + 16 MHz
Internal multispeed ULP RC oscillator 64 kHz to 4 MHz
PLL
Clock control
RTC/AWU
2x watchdogs (independent and window)
37/51/83/109/115 I/Os
Cyclic redundancy check (CRC)
Voltage scaling 3 modes
Touch sensing
ARM Cortex-M3 CPU 32 MHz
Nested vector interrupt controller (NVIC)
JTAG/SW debug
Embedded Trace Macrocell (ETM)
Memory protection unit (MPU)
AHB bus matrix
12-channel DMA
512-Kbyte Flash memory / dual bank
80-Kbyte SRAM
128-byte backup data
16-Kbyte EEPROM
Boot ROM
Analog
2x 12-bit DAC
12-bit ADC SAR Up to 40 channels
2x comparators
2x op-amps
Temperature sensor
Connectivity
USB 2.0 FS
5x USART
3x SPI
2x I²C
Control
6x 16-bit timer
1x 32-bit timer
1x 16-bit basic timer
Display
LCD driver (8x40)

Figure 2. STM32L152RE Circuit Diagram /**4**/

A crystal oscillator is an electronic oscillator circuit which is often used to keep track of time to provide a stable clock signal for digital integrated circuits and to stabilize frequencies. The most common type of piezoelectric resonator used is a quartz crystal, so oscillator circuits incorporating them became known as crystal oscillators. A crystal is a solid in which the constituent atoms, molecules, or ions are packed in a regularly ordered, repeating pattern extending in all three spatial dimensions. /**5**/

A 32 MHz crystal oscillator is embedded into the STM32-L152RE microcontroller board, which was used as the system clock in this thesis.

To measure the temperature and humidity of the air inside the greenhouse a DHT22 sensor was chosen for its combination of accuracy, affordability, reliability, and availability.

DHT22 output is calibrated digital signal. It utilizes digital-signal-collecting-technique and humidity sensing technology assuring its reliability and stability. Its sensing elements are connected with an 8-bit single-chip computer. /**6**/

The communication between the MCU and this device occurs with the help of USART.

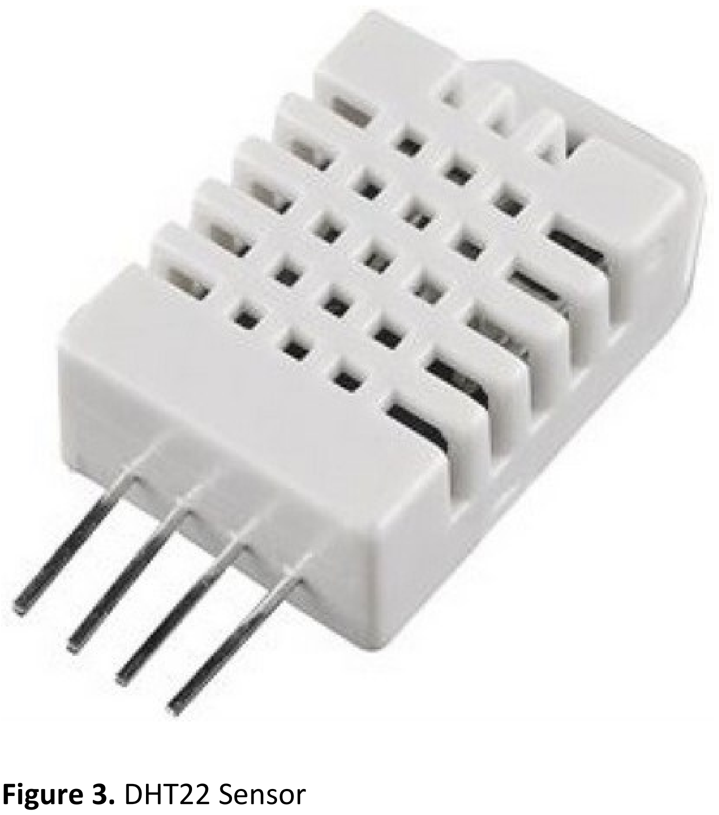
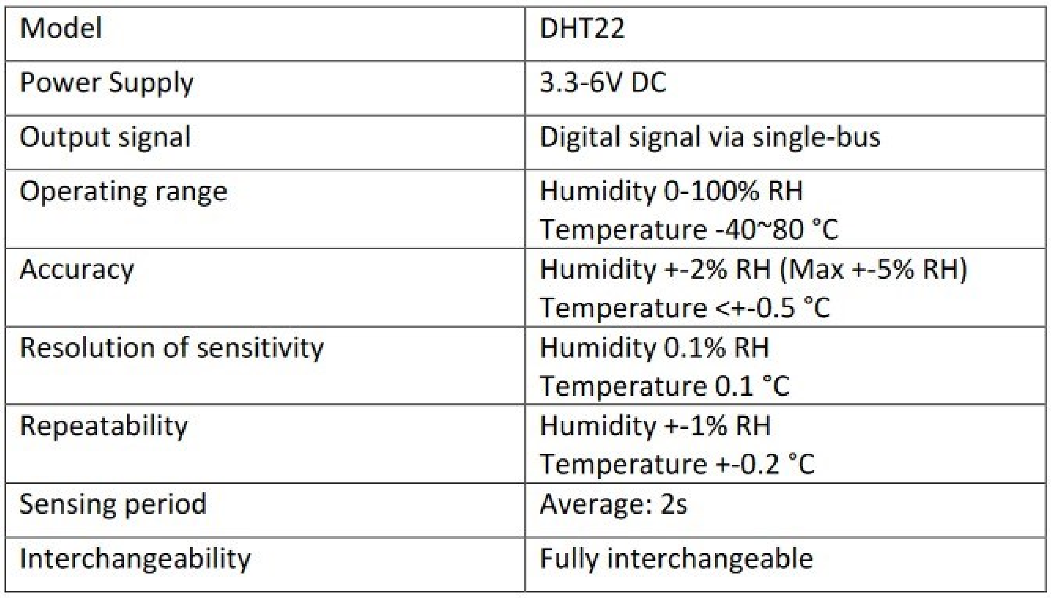


Figure 3. DHT22 Sensor

For the DHT22 sensor a pull-up resistor is required on the data line to reduce signal noise. For this purpose, a 10k Ohm resistor was used in this thesis for measurement accuracy.

Table 1. DHT22 Technical Specifications /**6**/



Model	DHT22
Power Supply	3.3-6V DC
Output signal	Digital signal via single-bus
Operating range	Humidity 0-100% RH Temperature -40~80 °C
Accuracy	Humidity +-2% RH (Max +-5% RH) Temperature <+-0.5 °C
Resolution of sensitivity	Humidity 0.1% RH Temperature 0.1 °C
Repeatability	Humidity +-1% RH Temperature +-0.2 °C
Sensing period	Average: 2s
Interchangeability	Fully interchangeable

For soil moisture sensor the SparkFun soil moisture sensor was chosen. It offers a reliable and cost-effective solution for monitoring soil moisture.

The two large, exposed pads function as probes for the sensor, together acting as a variable resistor. The more water that is in the soil means the better the conductivity between the pads will be, resulting in a lower resistance and a higher signal output. /**7**/

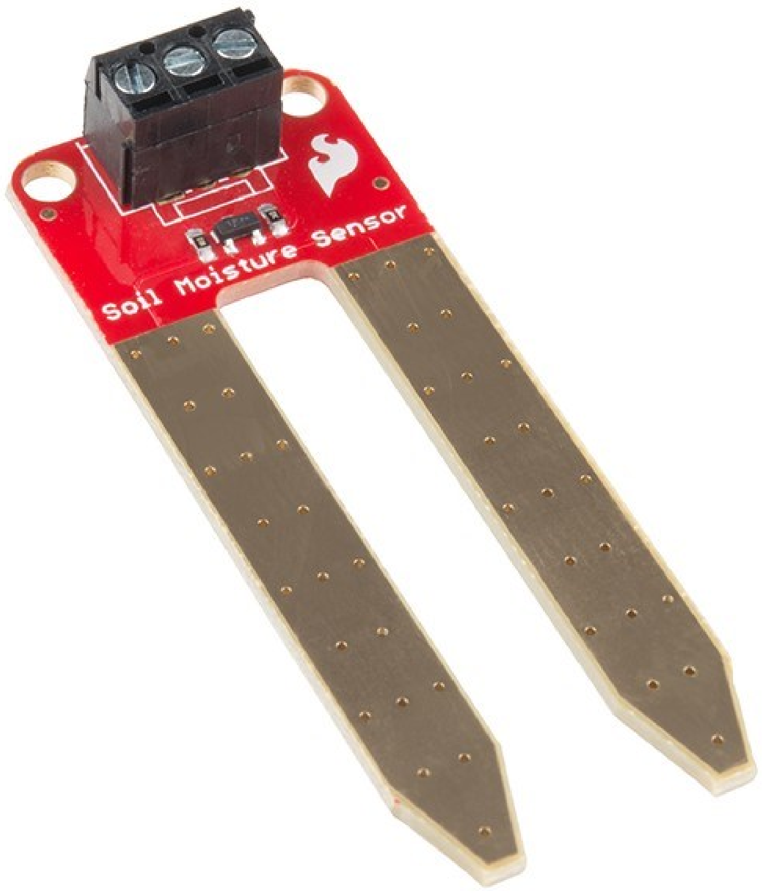


Figure 4. SparkFun Soil Moisture Sensor

For this project a UI was required for the user to be able to access the data effortlessly. Raspberry Pi can visualize sensor data, set thresholds, configure system settings, and view historical data making it smooth to monitor and manage the greenhouse environment. It also allows an easy way to connect to a Wi-Fi network and to power the STM32 making it a convenient solution for the project.

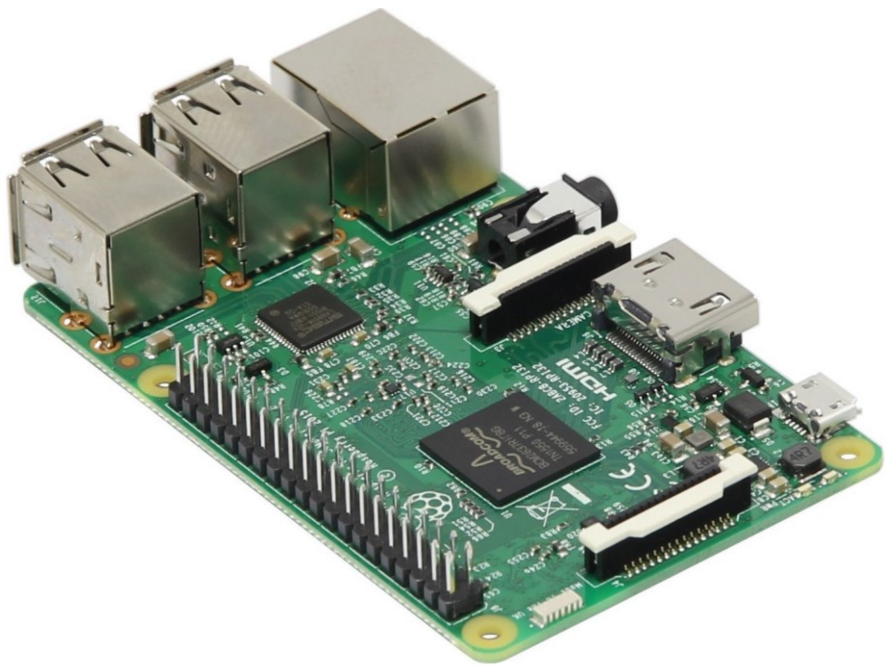


Figure 5. Raspberry Pi

4 SPECIFICATIONS AND DESIGN

The Specification and Design chapter provides a detailed and precise description of the requirements for the greenhouse monitoring system from the developer's perspective. This chapter translates the user-oriented requirements into technical specifications that guide the design and testing of the system. The specifications define the functional and non-functional requirements in measurable terms, ensuring that they can be designed to and tested for.

4.1 Functional and Non-Functional Specifications

The system must continuously monitor soil moisture, temperature, and humidity levels in the greenhouse. The sensors will provide real-time data input to the STM32 microcontroller. The STM32 microcontroller must process the raw data from the sensors and store it in an internal buffer for further analysis. The data processing includes filtering noise and converting analog signals to digital values. The system must provide a user interface that displays the monitored parameters in real-time. This interface should be accessible via USART to a connected device, such as a computer.

The system must operate continuously without failure, ensuring constant monitoring and data accuracy. The system architecture must support the addition of more sensors without significant changes to the existing design. The system must be energy-efficient to ensure long-term operation, particularly if powered by batteries or renewable energy sources. The sensors and hardware components must be robust and capable of withstanding the greenhouse environment, including humidity and temperature variations.

4.3 Detailed Design

The detailed design translates the specifications into a concrete implementation plan, outlining the hardware and software components, their interactions, and the overall system architecture.

The system consists of the STM32 microcontroller at the core, responsible for data acquisition, processing, and communication. The soil moisture sensor measures the volumetric content of water in the soil, outputting an analog signal proportional to soil moisture levels, which is then read by the STM32 ADC. The DHT22 Temperature and Humidity Sensor provides digital output for both temperature and humidity, directly interfacing with the STM32 GPIO pins. The USART Communication Module enables serial communication between the STM32 MCU and the user interface, ensuring data is reliably transmitted to a connected device for real-time monitoring.

The software components include sensor drivers, which are software modules to interface with the soil moisture and DHT22 sensors, handling data acquisition and initial processing. Data Processing Algorithms implemented on the STM32 filter noise and convert raw sensor data into meaningful values, including analog-to-digital conversion and temperature and humidity calculations. User Interface Software runs on a connected Raspberry Pi that communicates with the STM32 via USART, displaying real-time data.

The system's operation starts with initialization, where the STM32 peripherals, including ADC, GPIO, and USART, are configured. The system then continuously reads data from the soil moisture sensor (analog) and DHT22 sensor (digital), storing raw data in the internal buffer. This raw sensor data is filtered and processed, converting analog signals to digital values and applying necessary calculations for temperature and humidity. The processed data is then sent to the user interface via USART.

4.4 Preliminary Design

Before implementing the greenhouse monitoring system, a preliminary design outlines how the specifications will be fulfilled. This includes a list of circuits and methods to be used, major inputs, a list of major functions, their relations, and the steps to develop the greenhouse solution.

The major circuits include the soil moisture sensor circuit, the DHT22 temperature and humidity sensor circuit, the STM32 microcontroller circuit, and the USART communication circuit. The methods involve data acquisition from sensors, data processing algorithms, and USART communication protocols.

The major inputs are the analog signal from the soil moisture sensor and the digital signals from the DHT22 sensor. The major functions include initializing the system, reading sensor data, processing data, and transmitting data to the user interface. These functions are in interrelationship, where the sensor data is read and processed before being transmitted to the user interface.

The development steps include designing and testing each circuit, implementing the sensor drivers and data processing algorithms, developing the user interface software, integrating all components, and testing the entire system for reliability and accuracy.

5 THE IMPLEMENTATION OF THE PROJECT

In this chapter the implementation of the hardware and software parts of the project are explained.

5.1 Hardware Design Implementation

This chapter describes the hardware implementation of the prototype of this project. The prototype is built of several components: STM32-L152RE MCU board, DHT22 humidity and temperature sensor, SparkFun soil moisture sensor, resistor, and Raspberry Pi.

STM32-L152RE board is the core of the project. It can be programmed to control all the connected modules, which are essential for this project, as well as to store data obtained from the sensors.

The used features of the STM32-L152RE board are:

• USART

• ADC

• GPIO

• Timers

• RCC

DHT22 was chosen as the temperature and humidity sensor for this project. It is a essential element of the greenhouse monitoring system, used to measure both temperature and humidity within the greenhouse. This sensor offers reliable and precise data on environmental conditions, ensuring that the greenhouse maintains the optimal climate for plant growth. The useful features for this project are:

• Digital output signal via single bus

• Temperature range from -40 to +80 Celsius

• Power supply range from 3.3 V to 6 V

• Resolution of 0.1% RH and 0.1 ℃

• Small size

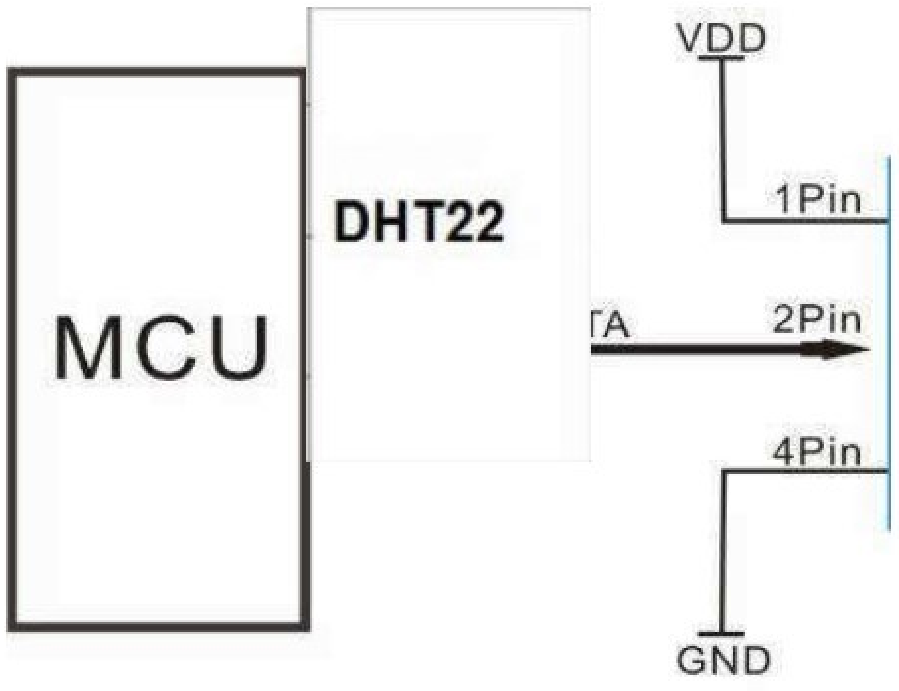
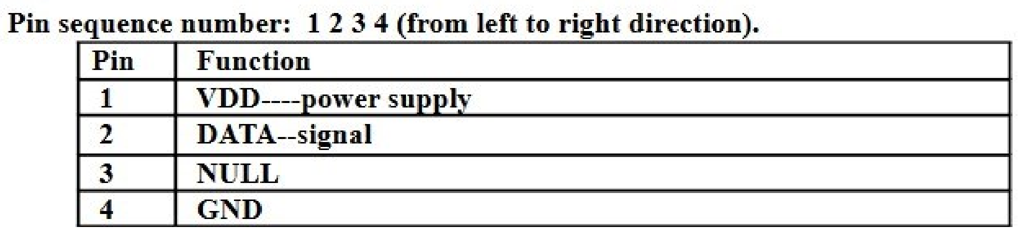


Figure 6. DHT22 connection schematic



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

Table 2. DHT pin function definition

SparkFun soil moisture sensor was chosen to measure the soil moisture of the plants. It is a key component of the greenhouse monitoring system, designed to measure soil moisture levels accurately. This sensor detects the volumetric water content in the soil, providing essential data for maintaining optimal plant hydration. Its notable features include:

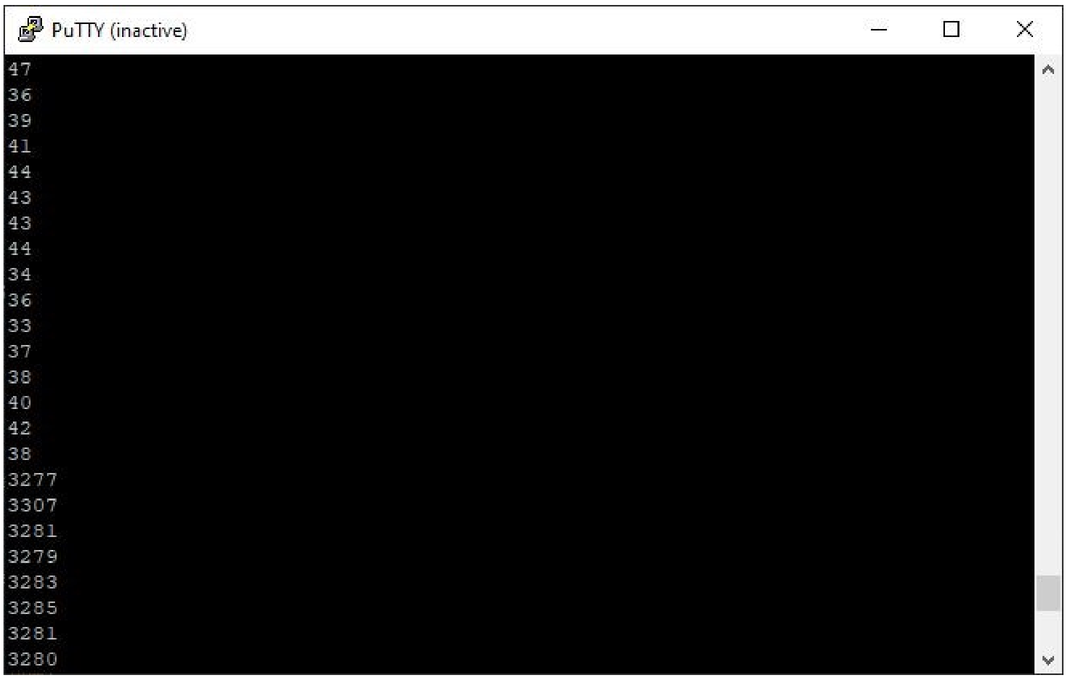
• Analog output

• PCB coated in Gold Finishing for corrosion resistance

• Power supply range from 3.3 V to 5 V

• Affordability

Different types of soil can affect the sensor, and the resulting readings can be different from one composition to the next. To measure the soil moisture the sensor must first be calibrated to get useful data for the user. The values when the sensor is completely dry and when it is submerged in water must be acclaimed for calibration.



```
47
36
39
41
44
43
43
44
34
36
33
37
38
40
42
38
3277
3307
3281
3279
3283
3285
3281
3280
```

Figure 7. Soil moisture calibration

To convert the analog value from the sensor to a percentage of moisture in the soil the following formula was used. First the sensor was held above water and then it was sunk underneath to calibrate the system.

$$Soil\ moisture\ \% = 100 * \frac{\text{Analog value}}{3200}$$

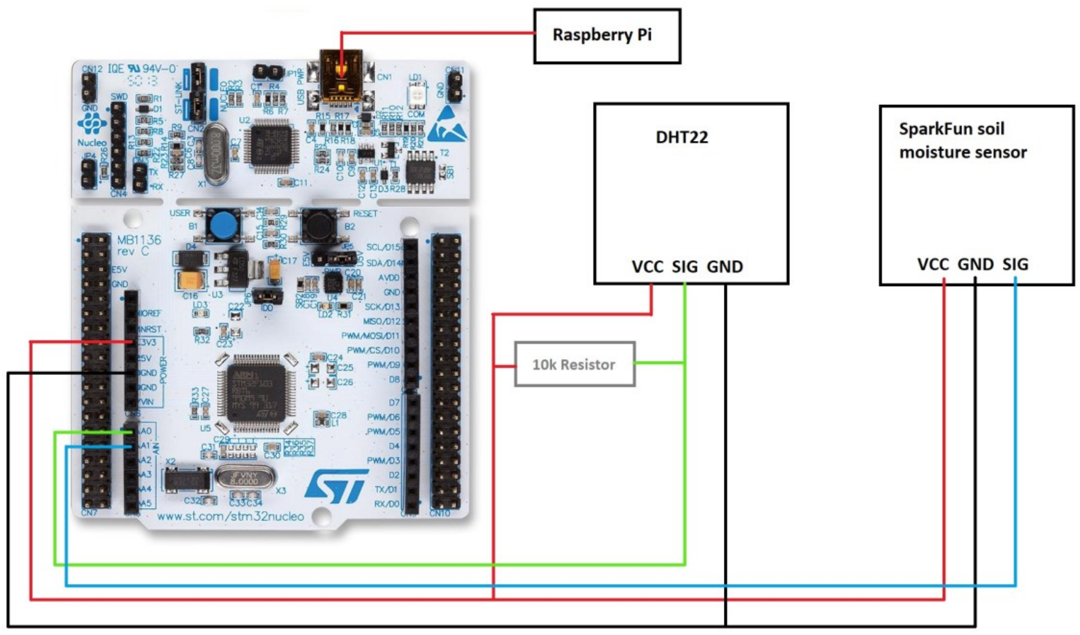


Figure 8. Device prototype schematic

The device prototype schematic provides a comprehensive visual representation of the hardware configuration and interconnections within the greenhouse monitoring system, detailing the integration of sensors, the STM32 microcontroller, and additional components.

5.2 Application Implementation

This chapter represents the implementation of the system application.

The application was built using STM32CubeIDE. First, all the predefined pinouts were cleared. After the project was built, most of the source code was deleted. Only the system_stm32l1xx.c file was left for the project. Inside the include folder the main.h and stm32l1xx_it.h files, were also deleted since own new code was written for the project without using the HAL libraries provided by the CubeIDE.

The main functionality of the application consists of several different parts, which are:

• System initialization

• Serial communication

• Sensor applications

• Continuous operation

The application consists of a main function, which initializes hardware and starts the system by launching the task with system initialization. Afterwards the sensor subfunctions are initiated. These will be described later in this work.

5.2.1 Main Function

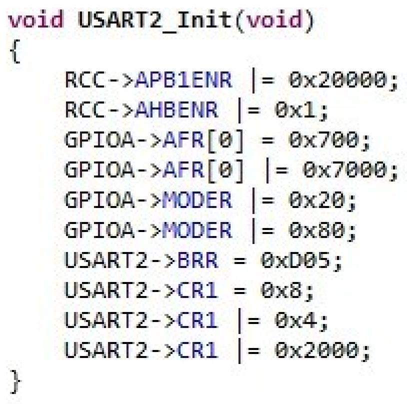
The main function serves as the entry point and central control loop for the application. Its purpose is to initialize the system, configure peripherals, and manage the overall behavior of the system.

The main function initializes the system clock and updates the system core clock to ensure proper timing and operation of the microcontroller. It then calls functions to initialize USART communication USART2_Init() and the ADC for reading soil sensor data ADC_Init(). These initializations configure the hardware peripherals.

Inside the main loop sensor data is read periodically, formatted, and transmitted over USART for display. The main function includes a delay using the usDelay() function to control the frequency of the user display output. This ensures that sensor readings occur at regular intervals preventing data overload. Then the main function calls functions to read soil moisture read_soil() and DHT22 temperature and humidity read_dht22() sensor data. Once the data is obtained, it is formatted into strings and transmitted over USART to the Raspberry Pi using the USART2_write() function.

5.2.2 USART

The provided function, USART2_Init, encapsulates the initialization process for USART2 on an STM32 microcontroller. By executing this function, USART2 is configured for communication utilizing pins PA2 and PA3 for TX and RX, respectively.



```
void USART2_Init(void)
{
    RCC->APB1ENR |= 0x20000;
    RCC->AHBENR |= 0x1;
    GPIOA->AFR[0] = 0x700;
    GPIOA->AFR[0] |= 0x7000;
    GPIOA->MODER |= 0x20;
    GPIOA->MODER |= 0x80;
    USART2->BRR = 0xD05;
    USART2->CR1 = 0x8;
    USART2->CR1 |= 0x4;
    USART2->CR1 |= 0x2000;
}
```

Figure 9. USART initialization function

This initialization routine prepares the microcontroller for USART2 communication.

Before configuring any peripheral, it is essential to ensure its clock is enabled. In this function, the first two lines enable the clocks required for USART2 and GPIO Port A. The RCC peripheral manages clock configuration in STM32 microcontrollers.

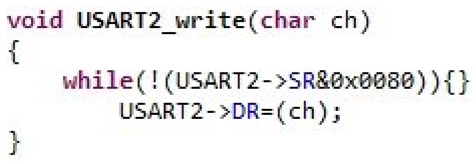
USART2 requires GPIO pins for communication. The next few lines configure pins PA2 and PA3 to function as USART2's TX and RX pins, respectively. Alternate function mode is set for these pins, allowing them to function as USART2 interface pins.

The line USART2->BRR = 0xD05; sets the Baud Rate Register of USART2 to achieve a baud rate of 9600 bits per second. Baud rate determines the speed of data transmission.

Configuring USART CR1: USART control registers control various aspects of communication, including enabling the transmitter, receiver, and USART itself.

- USART2->CR1 = 0x8; enables the transmitter.
- USART2->CR1 |= 0x4; enables the receiver.
- USART2->CR1 |= 0x2000; enables USART2.

The function USART2_write covers the process of transmitting a single character via USART2. This function ensures that the transmission occurs only when the USART is ready to accept new data.



```
void USART2_write(char ch)
{
    while(!(USART2->SR&0x0080)){}
        USART2->DR=(ch);
}
```

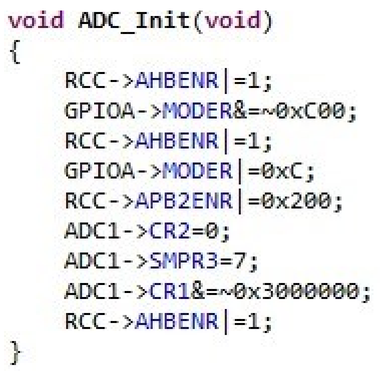
Figure 10. USART write function

Before sending data, it is crucial to ensure that the USART is ready to transmit. The code implements a busy-wait loop that continuously checks the TXE flag in the status register (USART2->SR). This loop waits until the TXE flag is set, indicating that the transmit DR is empty and ready to receive new data for transmission.

Once the USART is ready for transmission, the next line writes the character (ch) to the data register (USART2->DR) of USART2. The character is then transmitted serially via the USART communication interface.

5.2.3 ADC

The function ADC_Init initializes the process for configuring the ADC1 module on the STM32 microcontroller. The module is configured to accept analog input signal, convert it to digital value with a resolution of 12 bits, and sample it at a predefined rate.



```
void ADC_Init(void)
{
    RCC->AHBENR|=1;
    GPIOA->MODER&=~0xC00;
    RCC->AHBENR|=1;
    GPIOA->MODER|=0xC;
    RCC->APB2ENR|=0x200;
    ADC1->CR2=0;
    ADC1->SMPR3=7;
    ADC1->CR1&=~0x3000000;
    RCC->AHBENR|=1;
}
```

Figure 11. ADC initialization function

Before configuring peripherals, the clock is enabled. The first and last lines of the function enable the clock for the GPIO Port A and ADC1 module respectively. The RCC peripheral is responsible for managing clock configurations.

ADC1 requires GPIO pins to be properly configured for analog input. The following lines adjust the GPIO pins connected to the ADC input channel.

- GPIOA->MODER &= ~0xC00 clears bits 11 and 12 in the GPIOA mode register, ensuring that the corresponding pins are in input mode.
- GPIOA->MODER |= 0xC sets bits 1 and 0 in the GPIOA mode register, configuring the pins for analog mode.

Enabling ADC Peripheral Clock and Configuration.

- RCC->APB2ENR |= 0x200; enables the clock for ADC1 by setting the corresponding bit in the APB2ENR.

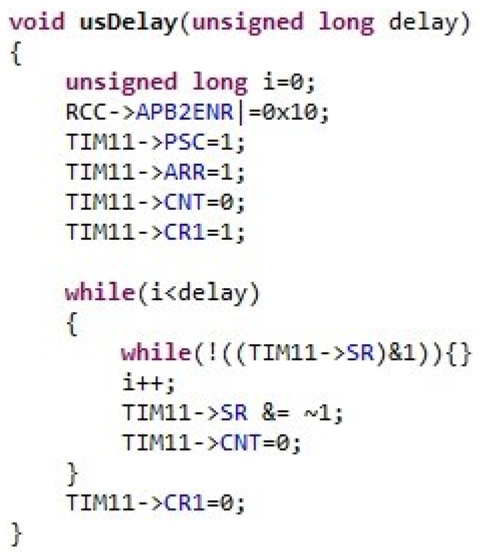
- ADC1->CR2 = 0; clears the control register 2 of ADC1, ensuring that the ADC is in its default state.

Configuring Sampling Rate and Resolution; ADC1->SMPR3 = 7 sets the sampling time for the ADC1 channel. The specific value of 7 indicates a sampling time of 384 ADC clock cycles.

Setting ADC Control Register 1; ADC1->CR1 &= ~0x3000000 clears bits 24 and 25 in the control register of ADC1. That sets the resolution of the ADC to 12 bits.

5.2.4 Delay

The following function usDelay implements a microsecond delay using Timer 11 (TIM11) on the microcontroller.



```c
void usDelay(unsigned long delay)
{
    unsigned long i=0;
    RCC->APB2ENR|=0x10;
    TIM11->PSC=1;
    TIM11->ARR=1;
    TIM11->CNT=0;
    TIM11->CR1=1;

    while(i<delay)
    {
        while(!((TIM11->SR)&1)){}
        i++;
        TIM11->SR &= ~1;
        TIM11->CNT=0;
    }
    TIM11->CR1=0;
}
```

Figure 12. Delay function

Before utilizing Timer 11, its clock must be enabled. The line RCC->APB2ENR |= 0x10; enables the clock for Timer 11 by setting the corresponding bit in the APB2 peripheral clock enable register (RCC->APB2ENR).

Timer 11 is configured to operate in a basic mode to create a microsecond delay.

- TIM11->PSC = 1: The prescaler is set to 1, meaning the timer will increment every clock cycle of its input clock.
- TIM11->ARR = 1: The ARR is set to 1, configuring Timer 11 to generate an interrupt after 1 count.
- TIM11->CNT = 0: The counter register is initialized to 0.
- TIM11->CR1 = 1: Timer 11 is enabled by setting the counter enable bit in the control register.

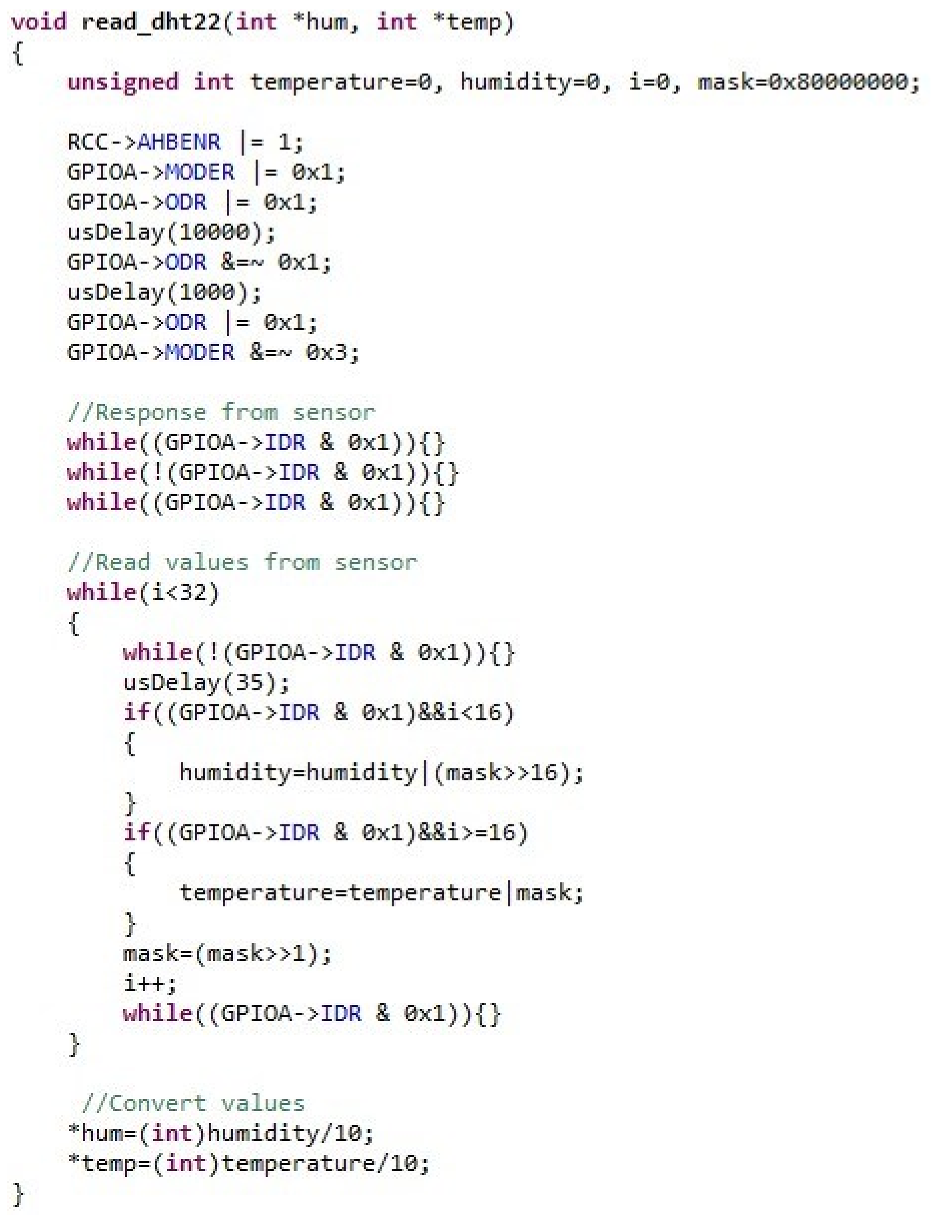
The function enters a while loop that creates a delay of approximately 1 microsecond.

- while(i<delay): This loop ensures that the total delay matches the specified duration.
- while(!((TIM11->SR) & 1)){}: Within each loop, the code waits until the update event flag of Timer 11 is set, indicating that the timer has reached its maximum value.
- TIM11->SR &= ~1: Upon reaching the maximum value, the update event flag is cleared by writing 0 to the corresponding bit in the status register (TIM11->SR).
- TIM11->CNT = 0: The counter is reset to 0 to prepare for the next repetition.

After the delay loop completes, Timer 11 is disabled to conserve power and resources by resetting the counter enable bit in the control register.

5.2.5 DHT22

The function read_dht22 reads the data from the temperature and humidity sensor connected to the controller.



```
void read_dht22(int *hum, int *temp)
{
    unsigned int temperature=0, humidity=0, i=0, mask=0x80000000;

    RCC->AHBENR |= 1;
    GPIOA->MODER |= 0x1;
    GPIOA->ODR |= 0x1;
    usDelay(10000);
    GPIOA->ODR &=~ 0x1;
    usDelay(1000);
    GPIOA->ODR |= 0x1;
    GPIOA->MODER &=~ 0x3;

    //Response from sensor
    while((GPIOA->IDR & 0x1)){}
    while(!(GPIOA->IDR & 0x1)){}
    while((GPIOA->IDR & 0x1)){}

    //Read values from sensor
    while(i<32)
    {
        while(!(GPIOA->IDR & 0x1)){}
        usDelay(35);
        if((GPIOA->IDR & 0x1)&&i<16)
        {
            humidity=humidity|(mask>>16);
        }
        if((GPIOA->IDR & 0x1)&&i>=16)
        {
            temperature=temperature|mask;
        }
        mask=(mask>>1);
        i++;
        while((GPIOA->IDR & 0x1)){}
    }

     //Convert values
    *hum=(int)humidity/10;
    *temp=(int)temperature/10;
}
```

Figure 13. Read DHT22 function

Configuring GPIO Pins:

- RCC->AHBENR |= 1; Enables the clock for GPIO Port A.
- GPIOA->MODER |= 0x1; Configures pin PA0 as output. It connects to DHT22.
- GPIOA->ODR |= 0x1; Sets pin PA0 high.

- usDelay(10000); Delays for 10 milliseconds to ensure the sensor stabilizes.
- GPIOA->ODR &=~ 0x1; Sets pin PA0 low to initiate communication.
- usDelay(1000); Delays for 1 millisecond.
- GPIOA->ODR |= 0x1; Sets pin PA0 high again.
- GPIOA->MODER &=~ 0x3; Configures pin PA0 as input.

Sensor Response**:**

- The code waits for the DHT22 sensor to respond. It expects a response signal from the sensor by monitoring the state of pin PA0.
- The loop waits for the pin to go low, then high, and finally low again, indicating the start of data transmission from the sensor.

Reading Data:

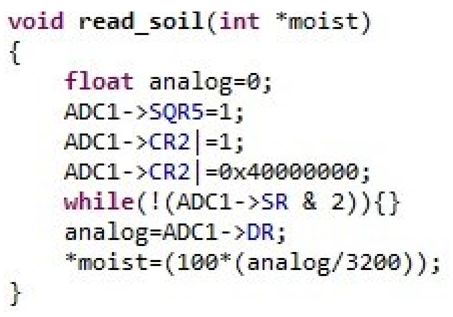
- The function reads 32 bits of data from the sensor, which consists of both temperature and humidity values.
- The mask variable is a shifting bitmask used to selectively set or clear individual bits in the temperature and humidity variables based on the sensor's output, simplifying the process of constructing 32-bit values from the received binary data stream.
- It goes through each bit and reads its value by observing the changes on pin PA0.
- For each bit, it waits for the pin to go high, then delays for 35 microseconds, and checks the pin's state again to determine the bit's value.
- The received bits are combined to form the temperature and humidity values.

Converting Values:

- The received temperature and humidity data are stored in temperature and humidity variables, respectively.
- To obtain the actual temperature and humidity values, the received data is divided by ten and assigned to the variables pointed to by the hum and temp pointers.

5.2.6 Soil Moisture Sensor

The function read_soil is designed to retrieve data from the SparkFun soil moisture sensor connected to the microcontroller. It facilitates the acquisition of soil moisture data by utilizing the ADC of the microcontroller and converts the analog sensor output into a meaningful percentage value.



```
void read_soil(int *moist)
{
    float analog=0;
    ADC1->SQR5=1;
    ADC1->CR2|=1;
    ADC1->CR2|=0x40000000;
    while(!(ADC1->SR & 2)){}
    analog=ADC1->DR;
    *moist=(100*(analog/3200));
}
```

Figure 14. Read soil function

Setting up ADC:

- ADC1->SQR5 = 1; Configures the ADC to sample from channel 1.
- ADC1->CR2 |= 1; Enables the ADC to start the conversion process.
- ADC1->CR2 |= 0x40000000; Starts the conversion process by setting the SWSTART bit in the control register.

The function enters a while loop to wait for the conversion to be complete. while(!(ADC1->SR & 2)){}; Waits until the EOC flag is set in the status register, indicating that the conversion process has finished.

Upon completion of the conversion, the analog value representing soil moisture is retrieved from the data register (analog = ADC1->DR;).

The analog value is converted to an user friendly percentage value with *moist = (100 * (analog / 3200)); by dividing the analog value by the maximum value and scaling it to a percentage scale.

6 RESULTS AND TESTING

This chapter describes the testing phase of the project. The point of this chapter is to show how the testing was done as well as show the results.

6.1 Preparations

First the necessary equipment was prepared for testing. The equipment used for this were:

- STM32-L152RE microcontroller board with a micro-USB cable connected to the PC
- PC with STM32CubeIDE and Putty
- DHT22 sensor
- SparkFun soil moisture sensor
- A breadboard
- Resistor, wires

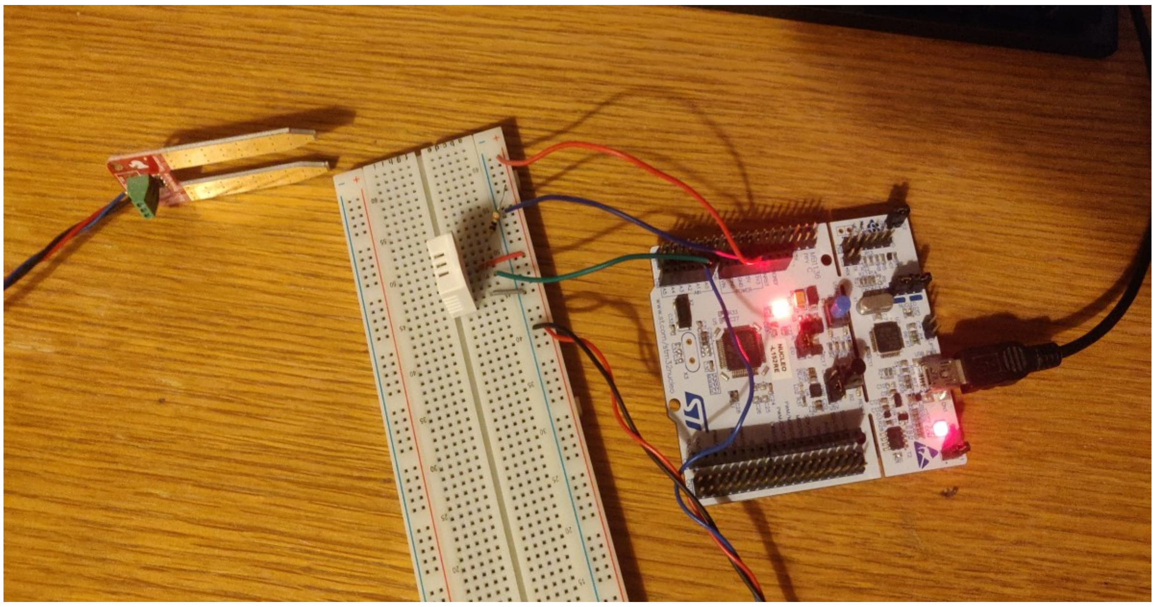
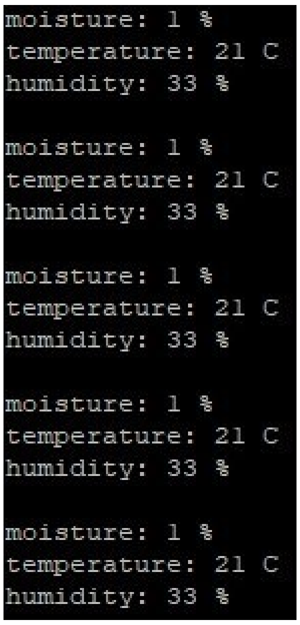


Figure 15. Prototype testing configuration

The components were connected to the microcontroller board as seen in the picture. The temperature and humidity sensor were connected to pin PA0 and the soil moisture sensor was connected to pin PA1. Each of the sensors were connected to the common ground and to the common voltage source (GND and 3.3V board pins). For the DHT22 sensor a pull-up resistor was required on the data line, so a 10k Ohm resistor was connected there for voltage stability.

Using STM32CubeIDE the program was loaded on the microprocessor. After all the connections were made and the program was loaded on board, the program was launched. Then the program was checked on the working PC using Putty to see if it runs correctly.

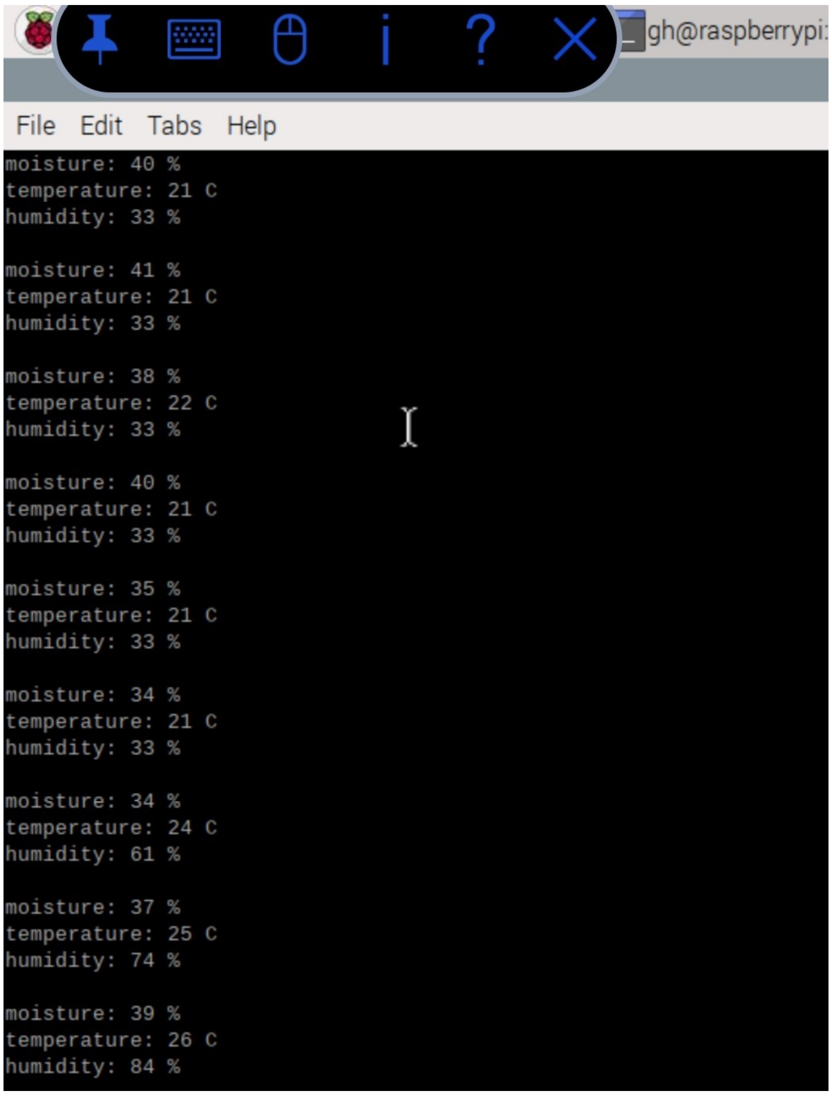


```
moisture: 1 %
temperature: 21 C
humidity: 33 %

moisture: 1 %
temperature: 21 C
humidity: 33 %

moisture: 1 %
temperature: 21 C
humidity: 33 %

moisture: 1 %
temperature: 21 C
humidity: 33 %

moisture: 1 %
temperature: 21 C
humidity: 33 %
```

Figure 16. Values on a PC using Putty

6.2 Testing phase

Once the application was receiving data from the sensors on the Raspberry Pi, using screen the RVNC application was turned on. Then it was possible to connect to the Raspberry remotely using a mobile phone. Next, the temperature, humidity, and moisture level were changed around the sensors to see if they were functioning properly. The values on the application seemed to follow changes made during testing quite rationally.



```
moisture: 40 %
temperature: 21 C
humidity: 33 %

moisture: 41 %
temperature: 21 C
humidity: 33 %

moisture: 38 %
temperature: 22 C
humidity: 33 %

moisture: 40 %
temperature: 21 C
humidity: 33 %

moisture: 35 %
temperature: 21 C
humidity: 33 %

moisture: 34 %
temperature: 21 C
humidity: 33 %

moisture: 34 %
temperature: 24 C
humidity: 61 %

moisture: 37 %
temperature: 25 C
humidity: 74 %

moisture: 39 %
temperature: 26 C
humidity: 84 %
```

Figure 17. Screen data during final testing

The sensors started measuring necessary data, and the data was visible through screen application using RVNC viewer on a mobile phone through home Wi-Fi. The prototype works as intended on the Raspberry Pi. The testing went successfully, and the project was made to operate as desired without flaws: the sensor data was gathered, stored, and sent over as designed.

6.3 Reliability and Usability

The system’s reliability was demonstrated through continuous operation without failures over an extended testing period. The sensors-maintained accuracy, and the mic

The system interface was evaluated for its ease of use and accessibility. The lack of complexity in the interface design allows quick data visualization and comprehension. The simplicity and clarity of the system contributed to its overall usability, allowing it to easily monitor greenhouse conditions.

6.4 Aims reached

The testing phase confirmed that the developed solution achieved its primary aims. The system provided reliable and accurate monitoring of soil moisture, temperature, and humidity levels. It also offered a user-friendly interface for real-time data visualization.

The greenhouse monitoring system developed for this thesis project demonstrated high reliability and usability. The system met all specified requirements and achieved the project's objectives, providing an effective solution for monitoring and maintaining environmental conditions in a greenhouse setting.

7 CONCLUSION AND FUTURE WORK

The focus of this thesis project was the development of a greenhouse monitoring system aimed at improving agricultural practices and greenhouse management. The objective was to create a comprehensive system capable of monitoring critical environmental parameters such as temperature, humidity, and soil moisture levels in real-time. The system was constructed using the STM32 microcontroller board, which interfaced with sensors including the DHT22 for temperature and humidity measurement within the greenhouse, and the SparkFun soil moisture sensor for detecting soil moisture levels in plants. Additionally, the Raspberry Pi facilitated visualization and remote accessibility.

The primary goal of the system is to provide a reliable solution for greenhouse monitoring, empowering users to monitor environmental conditions, identify needs, and make informed decisions regarding greenhouse management practices.

There is significant potential for the greenhouse monitoring system, driven by technological advancements. It can serve as a valuable tool for farmers, researchers, and greenhouse hobbyists, offering insights into environmental conditions and enabling proactive strategies. Furthermore, the scalability of the system allows for the integration of additional sensors and functionalities, such as multiple soil moisture sensors for monitoring multiple plants simultaneously or automating plant watering based on sensor readings.

As with any project, there are areas for further study and improvement. Future research could focus on enhancing the capabilities of the system by incorporating additional sensors to monitor additional environmental parameters. Additionally, exploring advanced data analytics techniques could provide deeper insights into environmental trends and further optimize greenhouse management practices.

REFERENCES

/1/ Dejan. DHT11 & DHT22 Sensors Temperature and Humidity Tutorial using Arduino. Accessed 13.3.2024. https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/

/2/ ST life augmented. Integrated Development Environment for STM32. Accessed 13.3.2024. https://www.st.com/en/development-tools/stm32cubeide.html

/3/ - STM32-L152RE. Accessed 14.3.2024. https://www.st.com/en/microcontrollers-microprocessors/stm32l152re.html

/4/ - RM0038 Reference manual. 2023. Accessed 15.3.2024. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiP9IK4iM6EAxWBKBAIHUQQBTAQFnoECBEQAQ&url=https%3A%2F%2Fwww.st.com%2Fresource%2Fen%2Freference_manual%2Fcd00240193-stm32l100xx-stm32l151xx-stm32l152xx-and-stm32l162xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf&usg=AOvVaw2oSqP_jDnBPl2cKI8cPC4h&opi=89978449

/5/ Crystal oscillator. Wikipedia. Accessed 14.3.2024. https://en.wikipedia.org/wiki/Crystal_oscillator

/6/ Spark Fun. SparkFun Soil Moisture Sensor. Accessed 13.3.2024. https://www.sparkfun.com/products/13637

/7/ Liu, Thomas. Digital-output relative humidity & temperature sensor/module DHT22. Accessed 15.3.2024. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf