MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL AVIATION UNIVERSITY

Department of Electronics, Robotics, Monitoring and

IoT Technologies

ADMIT TO THE DEFENSE

Head of the graduate department

_______________ Volodymyr Shutko

«_____» ____________ 20__.

DIPLOMA

(EXPLANATORY NOTE)

GRADUATE OF A BACHELOR'S DEGREE

SPECIALTY 171 «ELECTRONICS»

Subject: Remote temperature and humidity monitoring system.

Performer: Oleksandr Kazantsev.

Supervisor: sr. lecturer, Mykola Bidnyi.

Standard controller: _________ ___________________

Kyiv – 2023

NATIONAL AVIATION UNIVERSITY

Educational and scientific institute ________________________________.

Department of Electronics, Robotics, Monitoring and IoT Technologies.

Specialty 171 «Electronics».

APPROVE

Head of the graduate department

__________ (V. Shutko)

«____» __________ 20__.

OBJECTIVE

to defense a diploma work

Oleksandr Kazantsev

1. Subject «Remote temperature and humidity monitoring system» approved by the rector’s order dated 14.12.2017 № 594/st.

2. Deadline: from ________ to ________

3. Output data: temperature and humidity measurement data, microcontrollers.

4. Content of the explanatory note: analytical review of temperature and humidity measurement systems. Comparison of the relevance of temperature and humidity measurement systems. Comparison of microcontrollers. Modeling of an electronic temperature and humidity measurement system.

5. List of mandatory illustrative material: pictures, tables, code.

6. Calendar plan-schedule

№	Subject	Deadline	Supervisor Signature
1.	Gathering the necessary information on the topic of the thesis.	24.02.2023 – 03.03.2023	
2.	Select literature for the topic research.	03.03.2023 – 17.03.2023	
3.	Analysis of monitoring systems.	17.03.2023 – 31.03.2023	
4.	Analysis of microcontrollers.	31.03.2023 – 14.04.2023	
5.	Preparation and execution of the of the theoretical section of the thesis work.	14.04.2023 – 21.04.2023	
6.	Global Logic laboratory visit. Arduino platform investigation.	24.04.2023 – 05.05.2023	
7.	Hardware investigation. Preparing the hardware setup	08.05.2023 – 12.05.2023	
8.	Block diagram design	15.05.2023 – 19.05.2023	
9.	Developing a working prototype	22.05.2023 – 27.05.2023	

7. Consultation on specific section(s):

Section title	Consultant (position, full name)	Date, signature	
		Assigner	Accepter

8. Issue assignment date: «____» _________ 20 __ p.

Diploma supervisor (project): ________ ______________

The task was accepted: _____________ Oleksandr Kazantsev

Abstract

Explanatory note to the diploma " Remote temperature and humidity monitoring system ": 56 p., 13 pictures, 3 tables, 15 references.

Object of study: Microcontroller-based air humidity and temperature monitoring system.

Objective of the work: To develop an air humidity and temperature monitoring system.

Research methods: comparative analysis, processing of literature sources, prototype modeling.

AIR HUMIDITY, TEMPERATURE MONITORING, ELECTRONIC SYSTEM, MICROCONTROLLER, SENSORS, TIMERS

Content

Introduction

Remote temperature and humidity monitoring system is an emerging technology that is gaining widespread use in various industries. This system is designed to measure and monitor temperature and humidity levels in remote locations, and transmit the data to a central control unit. This technology has revolutionized the way businesses manage their operations, especially those that rely on consistent temperature and humidity levels, such as the pharmaceutical industry, food industry, and data centers. By implementing a remote temperature and humidity monitoring system, businesses can ensure that their products and services meet the required standards.

The demand for remote temperature and humidity monitoring systems has increased rapidly in recent years due to various factors. The first factor is the growing awareness among businesses of the importance of maintaining consistent temperature and humidity levels in their operations. The second factor is the increasing need for remote monitoring systems, which has been fueled by the COVID-19 pandemic. With many businesses operating remotely, it has become critical to have a reliable monitoring system in place to ensure that operations continue smoothly.

The benefits of using a remote temperature and humidity monitoring system are numerous. One of the primary benefits is the ability to monitor and control temperature and humidity levels in real-time, which helps prevent costly losses due to equipment malfunction, power outages, or other issues that could compromise product quality. Additionally, remote monitoring systems can help businesses save time and money by reducing the need for on-site personnel to monitor temperature and humidity levels manually.

CHAPTER 1. Monitoring systems.

1.1. The Benefits of Monitoring Systems for Operations Optimization and Safety Enhancement

Temperature and humidity are two critical environmental factors that need to be monitored and controlled in various settings such as homes, offices, industries, and hospitals. Accurate and reliable monitoring of temperature and humidity levels is essential for maintaining optimal living and working conditions, preventing the growth of harmful microorganisms, and preserving sensitive materials.

Monitoring systems are crucial tools that allow users to observe and track the performance of different components in real-time. These systems provide valuable insights into the behavior of systems, allowing users to detect potential issues before they escalate into significant problems. In this chapter, we will discuss the benefits of monitoring systems and their impact on operations, efficiency, and safety.

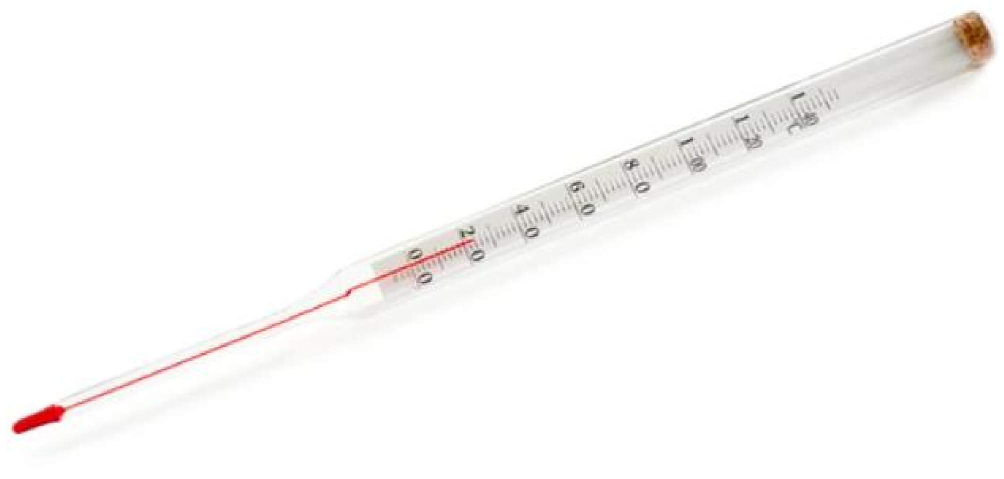
Optimizing Operations. One key benefit of monitoring systems is that they can help organizations optimize their operations. By constantly monitoring performance metrics, organizations can identify inefficiencies and areas where improvements can be made. For example, in a manufacturing plant, monitoring systems can track the performance of machinery and detect potential issues before they result in downtime. By addressing these issues proactively, manufacturers can minimize downtime, reduce maintenance costs, and improve overall efficiency.

Enhancing Safety. Another benefit of monitoring systems is that they can improve safety in various industries. Safety is a critical concern in many industries, such as energy production, transportation, and healthcare. Monitoring systems can track the behavior of various components and identify potential safety hazards. For example, in an oil refinery, monitoring systems can detect gas leaks and notify operators to take appropriate action to prevent accidents. By detecting potential safety hazards early, monitoring systems can help prevent accidents, injuries, and fatalities.

1.2. Thermometers

Thermometers are widely used to monitor temperature in various settings such as homes, hospitals, laboratories, and industries. In medical settings, thermometers are essential tools for diagnosing and monitoring fever, which is a common symptom of many diseases. In homes, thermometers are used to monitor room temperature and ensure a comfortable living environment. In industries and laboratories, thermometers are used to monitor process temperature and ensure that materials are processed under optimal conditions.

Liquid thermometers consist of a glass tube filled with a liquid, usually mercury or alcohol, that expands or contracts as it is exposed to changes in temperature. The expansion or contraction of the liquid is then measured on a calibrated scale, which indicates the temperature. Liquid thermometers are relatively inexpensive, easy to use, and provide accurate temperature readings. However, they are also prone to breakage, which can release toxic mercury or alcohol, and require regular calibration to ensure accurate readings.



Pic. 1.2.1. Liquid thermometer

Digital thermometers, on the other hand, use electronic sensors to measure temperature and display the readings on a digital screen. Digital thermometers are more expensive than liquid thermometers, but they provide faster and more accurate readings. They are also less prone to breakage and do not contain toxic materials, making them safer to use. Digital thermometers can be used to measure temperature orally, rectally, or under the arm, and some models can even measure temperature without making contact with the skin, which is particularly useful for measuring the temperature of infants.

Pic. 1.2.2. Digital thermometer

While thermometers are essential tools for monitoring temperature, they do have some limitations. For example, thermometers only measure temperature at a specific point, which may not reflect the overall temperature of a room or process. In industries and laboratories, where process temperature is critical, it is necessary to use multiple thermometers to monitor the temperature at various points. Additionally, thermometers may not be able to measure temperature accurately in extreme temperatures, such as in high-temperature furnaces or in liquid nitrogen.

1.3. Hygrometers

In homes, hygrometers are used to monitor indoor humidity levels to ensure a comfortable living environment and prevent problems such as mold growth, which can occur in areas with high humidity. In industries, hygrometers are used to monitor the humidity levels in manufacturing processes, particularly in industries such as food and pharmaceuticals where humidity control is critical. Humidity can affect the quality of products and can lead to spoilage or degradation if not monitored and controlled properly.

Electronic hygrometers use sensors to measure the humidity level in the air and provide accurate and reliable readings. They can measure humidity levels within a wide range and are capable of providing continuous monitoring of humidity levels, making them suitable for use in manufacturing processes. Electronic hygrometers can also provide additional features such as data logging and alerts when humidity levels exceed certain thresholds.

Pic. 1.3.1. Digital hygrometer

Mechanical hygrometers, on the other hand, use a hair or synthetic material that changes shape as it absorbs moisture to measure humidity levels. They are less accurate than electronic hygrometers and require regular calibration to ensure accurate readings. However, they are less expensive and do not require a power source, making them suitable for use in areas where power is not readily available. Mechanical hygrometers are also less prone to failure than electronic hygrometers, making them suitable for use in harsh environments.

Pic. 1.3.1. Mechanical hygrometer

1.4. Data loggers

Data loggers have many applications in various industries. They are used to monitor environmental conditions in food storage facilities, pharmaceutical storage areas, and manufacturing plants. In laboratories, data loggers are used to monitor temperature and humidity levels in experiments, while in museums, they are used to monitor temperature and humidity levels to prevent damage to artifacts. Data loggers are also commonly used in the transportation of perishable goods to ensure that the temperature and humidity levels remain within acceptable ranges during transport.

Digital data loggers offer many advantages over analog data loggers. They provide more accurate and reliable data, have a larger memory capacity, and are easier to use. Digital data loggers are capable of providing real-time data and can be programmed to send alerts when temperature or humidity levels exceed predetermined thresholds. They can also be connected to a computer or a network for remote monitoring and data analysis.

Pic. 1.4.1. Digital data logger

Analog data loggers, on the other hand, use a chart recorder to record temperature and humidity levels over time. They are less accurate and reliable than digital data loggers and require regular maintenance to ensure accurate readings. Analog data loggers also have limited memory capacity and do not provide real-time data or alerts when temperature or humidity levels exceed predetermined thresholds. However, they are less expensive than digital data loggers and do not require a power source, making them suitable for use in areas where power is not readily available.

Pic. 1.4.2. Analog data logger

1.5. Electronic monitoring system

With the advancements in technology, electronic temperature and humidity monitoring systems have become more popular. These systems use electronic sensors to measure temperature and humidity levels and provide accurate and reliable readings. They are capable of storing large amounts of data and can be connected to a computer or network for remote monitoring and analysis.

Electronic temperature and humidity monitoring systems offer many advantages over analog monitoring systems. They provide faster and more accurate readings, have a larger memory capacity, and can be programmed to send alerts when temperature or humidity levels exceed predetermined thresholds. They also do not require regular calibration like analog devices, which saves time and money. Electronic monitoring systems are widely used in various industries such as food storage, pharmaceuticals, and healthcare.

One of the most popular electronic monitoring systems is the Arduino-based temperature and humidity monitoring system. Arduino is an open-source hardware and software platform that can be used to develop electronic projects. The Arduino-based temperature and humidity monitoring system consists of an Arduino board, a temperature sensor, a humidity sensor, and an LCD display. The system is programmed to measure temperature and humidity levels at regular intervals and display the readings on the LCD display. It can also be programmed to send alerts when temperature or humidity levels exceed predetermined thresholds.

1.6. Challenges with Analog Monitoring Systems

Analog monitoring systems may not be as accurate as electronic systems. This is because analog systems are affected by factors such as environmental changes, calibration errors, and mechanical wear and tear. These limitations can lead to inaccuracies in temperature and humidity measurements, which can have serious consequences in industries such as food storage and pharmaceuticals.

Electronic monitoring systems offer several advantages over analog systems. They are more accurate, reliable, and efficient. Electronic systems use sensors that provide precise readings, and the data can be easily stored and analyzed using software. Electronic systems can also be programmed to send alerts when temperature or humidity levels exceed predetermined thresholds, which can prevent costly damages and losses.

One example of an electronic monitoring system is the Internet of Things (IoT) temperature and humidity monitoring system. The IoT system consists of a network of sensors that are connected to the internet, allowing for real-time monitoring and analysis. The sensors can be placed in various locations, such as warehouses, laboratories, and healthcare facilities, to monitor temperature and humidity levels. The system can be programmed to send alerts to users' mobile devices when readings exceed predetermined thresholds. This allows for quick action to be taken to prevent damages or losses.

Another advantage of electronic monitoring systems is their ability to integrate with other systems. For example, electronic monitoring systems can be integrated with automated temperature and humidity control systems, allowing for real-time adjustments to be made based on readings. This can save energy, reduce costs, and improve efficiency.

While analog monitoring systems have been used for many years, they have several limitations that make them less efficient and effective than electronic monitoring systems. Electronic monitoring systems offer several advantages, including accuracy, reliability, efficiency, and the ability to integrate with other systems. With the increasing demand for accurate and reliable temperature and humidity monitoring, electronic monitoring systems, such as IoT-based systems, are expected to become more popular in the future.

CHAPTER 2. Microcontrollers

2.1. Technical Features and Broad Applications

Microcontrollers are tiny computers that control and observe devices and systems. Industrial automation, consumer electronics, robotics and IoT devices, all these fall within their wide range of applications.

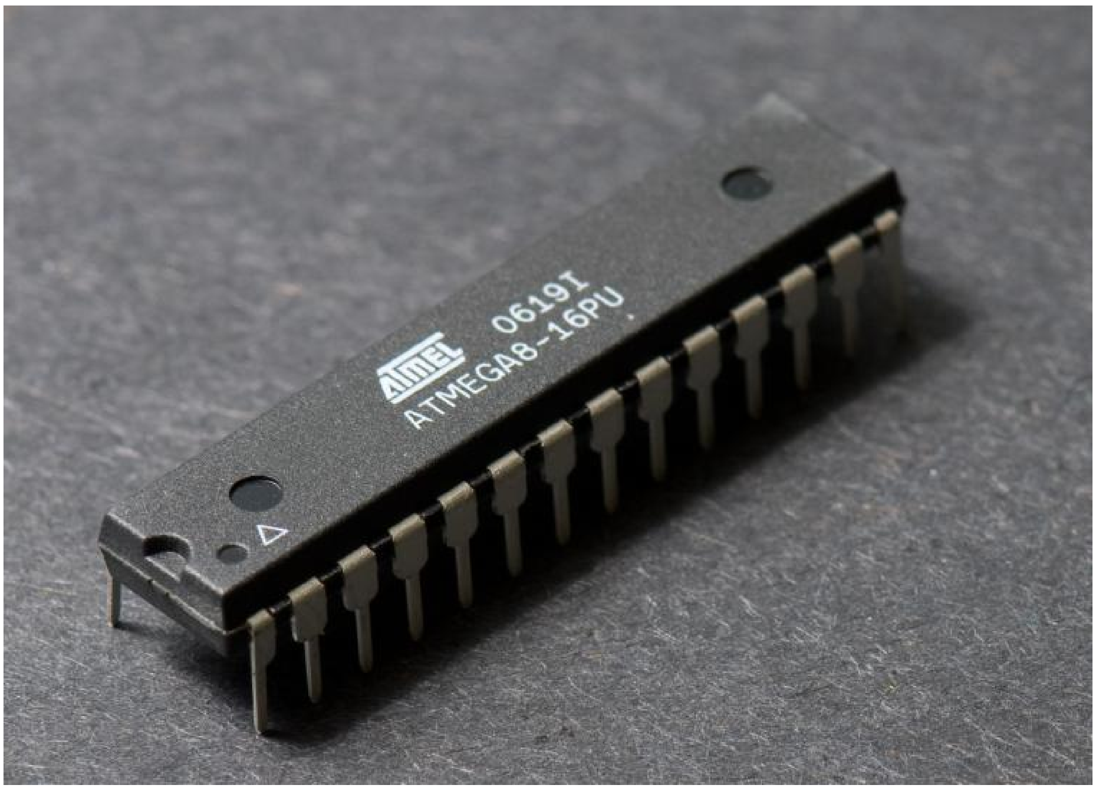
Features of Microcontrollers. One key feature of microcontrollers is their ability to operate with low power consumption, making them ideal for use in battery-powered devices. They also have real-time processing capabilities, allowing them to respond quickly to changes in input signals. Another feature of microcontrollers is their small form factor, which makes them easy to integrate into different devices and systems. Additionally, microcontrollers have a range of input and output capabilities, such as analog-to-digital conversion, PWM output, and serial communication, that make them versatile for use in various applications.

Applications of Microcontrollers. Remote monitoring systems for humidity and temperature often utilize microcontrollers to control and monitor different components. In the food industry, for example, microcontrollers are used in refrigeration systems to maintain optimal temperature and humidity levels for storing perishable goods. In greenhouses, microcontrollers are used to control heating, ventilation, and irrigation systems, ensuring that plants receive the ideal temperature and humidity conditions for growth. In medical research labs, microcontrollers are used in environmental chambers to precisely control temperature and humidity levels for testing and storing samples.

I'll focus on comparing and contrasting the Atmel 8-bit AVR microcontroller commonly used in Arduino and the RP2040 microcontroller used in Raspberry Pi in this section of the discussion. I will also compare them to the STM32, to provide a comprehensive overview of the features and capabilities of these devices.

2.2. Atmel 8-bit AVR microcontroller

The AVR microcontroller family has gained a substantial following among both hobbyists and developers since its inception in the early 1990s. One of its primary selling points is its user-friendly nature, making programming and understanding a breeze. This feature has made it a preferred choice for beginners just starting to explore the world of microcontrollers and electronics.



Pic. 2.2. Atmel 8-bit microcontroller

Furthermore, the AVR microcontroller's energy efficiency is impressive, making it an excellent option for battery-powered devices, including portable electronics, wearable gadgets, and remote sensors. Additionally, the microcontroller's low power consumption results in less heat production, ultimately prolonging the lifespan of the device.

Timers, interrupts, and ADCs are just a few of the many functions and abilities available on the AVR microcontroller[1]. Because of these characteristics, it is adaptable and able to carry out a variety of activities, including regulating motors and servos and measuring temperature and light levels. For applications that need analog input, such audio processing or environmental monitoring, the ADCs are particularly helpful..

The Arduino platform is one of the most popular applications for the AVR microprocessor. Arduino boards employ AVR microcontrollers as its primary processing unit, and the Arduino IDE offers developers with an easy-to-use programming environment. This has made the AVR microcontroller more accessible to a broader spectrum of users, including hobbyists and artists with no training in electronics or programming.

The AVR microcontroller, however, has several restrictions[1]. One of the most significant drawbacks is its 8-bit design, which means it has less memory and processing capability than 32-bit microcontrollers. The intricacy of the applications that may be built utilizing the AVR microcontroller may be limited as a result. Furthermore, the AVR microcontroller lacks the power of several other microcontrollers, such as the STM32 or the RP2040.

Despite these restrictions, the AVR microcontroller is still a popular choice for many applications. Its simplicity, low power consumption, and wide range of capabilities make it a handy and useful device for a variety of tasks.

2.2.1. Technical characteristics

According to the datasheet[1], technical characteristics are:

- **Architecture:** 8-bit RISC
- **Clock Speed:** Up to 32 MHz
- **Flash Memory:** Up to 256 KB
- **SRAM:** Up to 16 KB

- **Operating Voltage:** 1.8V to 5.5V
- **ADC Resolution:** Up to 10-bit
- **Digital I/O Pins:** Up to 86
- **Communication:** USART, SPI, I2C, USB
- **Development Tools:** Atmel Studio, Arduino IDE
- **PWM Channels:** Up to 6 channels with 8-bit resolution or 2 channels with 16-bit resolution. PWM (Pulse Width Modulation) is a method of controlling analog circuits using digital signals.
- **Timers:** Up to 3 16-bit timers with up to 3 output compare or input capture channels. Timers can be used for measuring time intervals, generating periodic signals, or controlling motor speed.
- **Watchdog Timer:** A built-in timer that can reset the microcontroller if the program gets stuck in an infinite loop or experiences other errors. This helps prevent the system from getting stuck in an undefined state.
- **Sleep Modes:** The AVR microcontroller can enter low-power sleep modes, reducing power consumption when the system is idle or not performing any tasks.
- **Temperature Range:** The AVR microcontroller can operate in a wide temperature range, from -40°C to 85°C, making it suitable for harsh environments.
- **Bootloader Support:** The AVR microcontroller can be programmed using a bootloader, which allows for firmware updates over a communication channel such as UART or USB. This can be useful in situations where the microcontroller is embedded in a device and difficult to access for programming.

2.2.2. Advantages of Atmel 8-bit AVR microcontroller

1. **Simple architecture and easy to program.** The architecture of the Arduino microcontroller is straightforward, making it simple to comprehend and program. Its C++-based programming language is easy to learn even for newcomers,

making it a popular option for students and hobbyists. Because the Arduino software is open-source, users may readily adapt it to meet their own requirements.

2. **Low power consumption.** Due to its low power requirements, the Arduino microcontroller is a great option for battery-powered applications. When not in use, the microcontroller may be put into sleep mode, which considerably lowers its power usage. This makes it a fantastic option for tasks requiring a lengthy battery life.

3. **Wide range of capabilities, including timers, interrupts, and ADCs[1].** The Arduino microcontroller has a wide range of functionalities, including timer, interrupt, and ADC operations. It is suitable for a range of applications as a result, from simple LED blinking projects to more challenging robotics and automation projects. While analog signals like temperature and light may be monitored using an ADC, time-sensitive actions can be performed utilizing timers and interrupts.

4. **Versatile and can be used in a wide range of applications.** Numerous uses for the flexible Arduino microcontroller are possible. Its capabilities may be increased by using shields, which are supplementary boards that can be placed on top of the Arduino board to provide greater functionality. This makes it possible to readily adapt the Arduino to operate with a range of applications.

5. **Low-cost.** Due to its low price, the Arduino microcontroller is available to everyone, including professionals, students, and enthusiasts. Due of the platform's inexpensive price, consumers may test it out without breaking the bank. Additionally, because the platform is open-source, users may make their own unique boards, which can sometimes be far less expensive than equivalents that are for sale.

2.2.3. Disadvantages of Atmel 8-bit AVR microcontroller

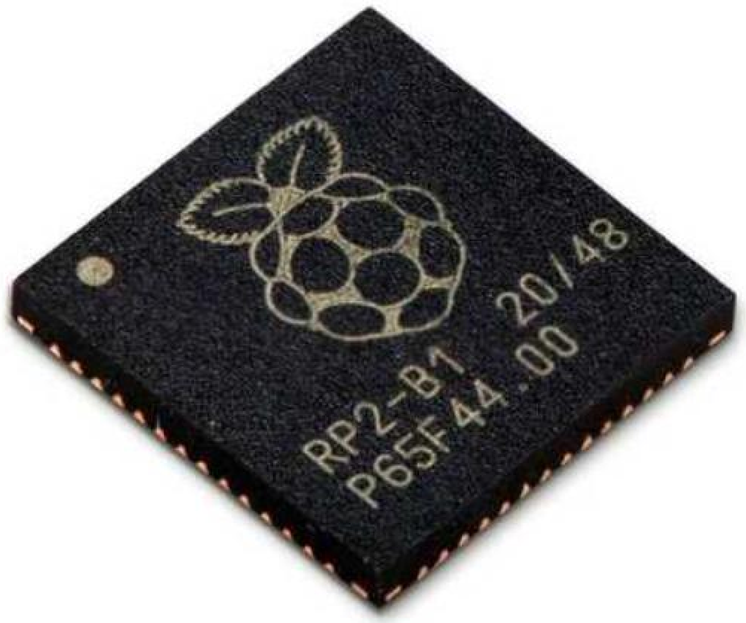
1. **Limited memory and processing power compared to more advanced microcontrollers.** While the Arduino platform is intuitive to use and simple to write, it may not have as much memory or processing capacity as advanced microcontrollers. This makes it difficult to use for applications that are more complex requiring greater

processing power or memory. However, the platform does provide a number of models, some with greater specifications than others, allowing customers to select a model that suits the particular requirements of a project…

2. **Limited support for advanced features, such as USB connectivity.** Another drawback of the Arduino platform is how little support it has for contemporary features like USB connections. While some versions come with a USB port, others might not, and such models might not have as sophisticated microcontrollers. As a result, it could be challenging to use the platform for tasks that need cutting-edge connectivity characteristics. The platform may optionally add cutting-edge features to a project by upgrading it with a number of shields and add-ons to broaden its range of capabilities.

2.3. RP2040 microcontroller

The Raspberry Pi RP2040 microprocessor, launched in early 2021, is a relatively fresh addition to the microcontroller industry. It is driven by a dual-core ARM Cortex-M0+ CPU and has USB, I2C, SPI, and UART interfaces among other things[2]. The RP2040 microcontroller has quickly gained popularity among developers and hobbyists because to its inexpensive cost, high performance, and ease of use.



Pic. 2.3. RP2040 microcontroller

One of the RP2040 microcontroller's key advantages is its dual-core architecture. This allows it to do many tasks at the same time, which is suitable for applications that require real-time processing or multi-tasking. The RP2040 microcontroller also features a wide range of peripherals, including USB, which makes it well-suited for applications that require connectivity, such as internet of things (IoT) devices or data loggers[2].

Another benefit is the high performance of the RP2040 microprocessor. It can handle complex applications and data processing duties because to its 133 MHz clock speed and 264 KB of RAM[2]. In addition, the RP2040 microcontroller incorporates a dedicated hardware-based PIO (programmable I/O) system for offloading processing tasks from the main CPU and giving additional processing capability.

The RP2040 microcontroller is incredibly simple to use since software development tools and libraries are widely accessible. To get started programming the RP2040 microcontroller, the Raspberry Pi Foundation provides a range of software development tools, including the C/C++ SDK. Additionally, there is a large community of developers and enthusiasts who have developed libraries and examples that can be used to quickly prototype and develop applications.

The RP2040 microcontroller has significant shortcomings despite its benefits. Its lack of connectivity with the Arduino ecosystem is one of its key drawbacks. The RP2040 microprocessor is not yet supported by the Arduino IDE, in contrast to the AVR microcontroller, which is commonly used in the Arduino platform. This might make learning how to program the RP2040 microcontroller more difficult for novices.

The RP2040 microcontroller's power consumption is another drawback[2]. It uses more power than the AVR microcontroller despite not being as power-hungry as some other microcontrollers, making it less suitable for battery-powered applications. The RP2040 microprocessor is also not as readily accessible as other microcontrollers, which might make acquiring and integrating it into projects more challenging.

2.3.1. Technical characteristics

According to the datasheet[2], technical characteristics:

- **Architecture:** Dual-core ARM Cortex-M0+
- **Clock Speed:** Up to 133 MHz
- **Flash Memory:** Up to 16 MB
- **SRAM:** Up to 264 KB
- **Operating Voltage:** 1.8V to 3.3V
- **ADC Resolution:** Up to 12-bit
- **Digital I/O Pins:** Up to 30
- **Communication:** UART, SPI, I2C, USB
- **Development Tools:** Raspberry Pi Pico C/C++ SDK
- **PWM Channels:** Up to 6 channels with 16-bit resolution. PWM (Pulse Width Modulation) is a method of controlling analog circuits using digital signals.

- **Timers:** Up to 2 16-bit timers with up to 4 output compare or input capture channels. Timers can be used for measuring time intervals, generating periodic signals, or controlling motor speed.
- **Watchdog Timer:** A built-in timer that can reset the microcontroller if the program gets stuck in an infinite loop or experiences other errors. This helps prevent the system from getting stuck in an undefined state.
- **Sleep Modes:** The RP2040 microcontroller can enter low-power sleep modes, reducing power consumption when the system is idle or not performing any tasks.
- **Temperature Range:** The RP2040 microcontroller can operate in a wide temperature range, from -20°C to 85°C, making it suitable for a range of environments.
- **Bootloader Support:** The RP2040 microcontroller can be programmed using a bootloader, which allows for firmware updates over a communication channel such as UART or USB. This can be useful in situations where the microcontroller is embedded in a device and difficult to access for programming.

2.3.2. Advantages of RP2040 microcontroller

1. **Powerful dual-core processor.** The Raspberry Pi platform is powered by a potent dual-core CPU that offers tremendous computing capacity for a number of applications. The Raspberry Pi is a great choice for difficult tasks that call for a lot of processing power due to its fast clock speed and remarkable performance capabilities.
2. **Wide range of capabilities, including ADCs, timers, and communication interfaces.** Analog-to-digital converters (ADCs), timers, and numerous communication interfaces are just a few of the features that the Raspberry Pi is capable of. Because of its adaptability, it may be used for a variety of tasks, including data recording and sensing applications as well as robotics and automation.
3. **Easy to program.** The Raspberry Pi is simple to program, and both novice and experienced users have access to a wide range of programming languages and

development tools. It is available to a wide variety of users because to its interoperability with well-known programming languages like Python and C.

4. **Suitable for a wide range of applications.** From home automation and security systems to industrial automation and Internet of Things devices, the Raspberry Pi is appropriate for a wide range of applications. Both novice and experienced users favor it because of its adaptability, cost, and simplicity of use.

5. **Low-cost.** The Raspberry Pi is a cost-effective platform that enables amateurs and students to experiment and study electronics and computing. The Raspberry Pi is a well-liked option for a variety of applications, from educational tools to media centers and gaming systems, thanks to its inexpensive cost.

2.3.3. Disadvantages of RP2040 microcontroller

1. **Limited support for advanced features, such as USB connectivity.** One potential drawback of the Raspberry Pi platform is its limited support for certain advanced features, such as USB connectivity. While the Raspberry Pi does have USB ports, these ports may not be sufficient for certain applications that require multiple USB devices or high-speed USB connections. Additionally, the Raspberry Pi's USB implementation may not be as reliable or robust as that of other platforms, which can cause issues for developers who rely on USB connectivity for their projects.

2. **Limited memory compared to more advanced microcontrollers.** While the Raspberry Pi is a powerful platform with many capabilities, it does have some limitations when it comes to memory. Compared to more advanced microcontrollers, the Raspberry Pi may have limited memory available for certain applications, which can be a significant limitation for projects that require a large amount of data storage or processing power. This can also be a challenge for developers who are working on complex projects that require a high degree of memory management.

3. **Limited availability of compatible development boards.** Another potential drawback of the Raspberry Pi platform is the limited availability of compatible

development boards. While there are many Raspberry Pi boards available, not all of them are compatible with all of the Raspberry Pi software and hardware, which can make it difficult for developers to find the right board for their project. Additionally, some Raspberry Pi boards may not be as reliable or well-supported as others, which can lead to issues with stability and performance over time.

2.4. STM32 microcontroller

The STM32 microcontroller is a well-known and widely-used microcontroller in the industry, being a 32-bit microcontroller based on the Arm Cortex-M processor architecture[3]. Its versatility and powerful features make it suitable for a wide range of applications, including industrial automation, robotics, and consumer electronics.

Pic. 2.4. STM32 microcontroller

The STM32 microcontroller family includes several different models with various specifications and capabilities, catering to different application needs. These microcontrollers are designed to be robust, reliable, and high-performing, ensuring that they can handle even the most demanding applications with ease.

One of the key advantages of the STM32 microcontroller is its compatibility with a range of software development tools and programming languages. This makes it easy to integrate into existing systems and work with different software stacks, simplifying the development process and reducing time to market.

The STM32 microcontroller also features a rich set of peripherals, including timers, interrupt controllers, analog-to-digital converters (ADCs), and digital-to-analog converters (DACs)[3]. These peripherals enable the microcontroller to handle complex tasks and interact with various sensors and actuators, providing a high degree of flexibility in application design.

Furthermore, the STM32 microcontroller has a low power consumption mode, making it a suitable choice for battery-powered applications. Its low power consumption mode allows it to conserve energy and extend battery life, making it ideal for applications in the Internet of Things (IoT) space.

Despite its advantages, there are some limitations to the STM32 microcontroller. One of the main limitations is its relatively high cost compared to some other microcontrollers. This may make it less attractive for hobbyist and low-cost projects.

Another limitation of the STM32 microcontroller is its complexity. Its advanced features and powerful capabilities require a higher level of expertise and experience to work with effectively, making it more challenging for beginners to get started with programming and development.

2.4.1. Technical characteristics

According to the datasheet[3], technical characteristics:

- **Architecture:** ARM Cortex-M0/M3/M4/M7
- **Clock Speed:** Up to 480 MHz
- **Flash Memory:** Up to 2 MB
- **SRAM:** Up to 640 KB
- **Operating Voltage:** 1.7V to 3.6V
- **ADC Resolution:** Up to 16-bit
- **Digital I/O Pins:** Up to 168
- **Communication:** USART, SPI, I2C, USB, Ethernet
- **Development Tools:** STM32CubeIDE, Keil uVision
- **PWM Channels:** Up to 18 channels with up to 16-bit resolution. PWM (Pulse Width Modulation) is a method of controlling analog circuits using digital signals.
- **Timers:** Up to 17 16-bit timers with up to 4 output compare or input capture channels. Timers can be used for measuring time intervals, generating periodic signals, or controlling motor speed.
- **Watchdog Timer:** A built-in timer that can reset the microcontroller if the program gets stuck in an infinite loop or experiences other errors. This helps prevent the system from getting stuck in an undefined state.
- **Sleep Modes:** The STM32 microcontroller can enter low-power sleep modes, reducing power consumption when the system is idle or not performing any tasks.
- **Temperature Range:** The STM32 microcontroller can operate in a wide temperature range, from -40°C to 125°C, making it suitable for harsh environments.
- **Bootloader Support:** The STM32 microcontroller can be programmed using a bootloader, which allows for firmware updates over a communication channel such as UART or USB. This can be useful in situations where the microcontroller is embedded in a device and difficult to access for programming.

2.4.2. Advantages of STM32 microcontroller

1. **Powerful 32-bit processor.** The STM32 has a powerful 32-bit processor that allows it to handle complex tasks quickly and efficiently. This makes it an excellent choice for applications that require high-speed processing, such as real-time control and signal processing.

2. **Wide range of capabilities, including high-speed communication interfaces, advanced timers, and analog peripherals.** These features make it a versatile option for a variety of applications, from robotics and automation to medical devices and industrial control systems.

3. **Wide range of compatible development tools and programming languages.** The STM32 has a wide range of compatible development tools and programming languages, which makes it easier for developers to get started and customize their applications. Additionally, the STM32 has a large library of open-source code and a well-established ecosystem, which can speed up development and reduce time-to-market.

4. **High level of integration and compatibility with existing systems[3].** This makes it a popular choice for upgrading legacy systems or building new systems that need to work with existing hardware and software. Additionally, the STM32 has a variety of communication protocols, such as Ethernet and CAN, which make it easy to integrate with other devices and systems.

5. **Large community of developers and resources available.** This community provides a wealth of knowledge, support, and resources that can help developers solve problems and create better applications. From forums and tutorials to online courses and documentation, there are plenty of resources available to help developers get the most out of the STM32 platform.

2.4.3. Disadvantages of STM32 microcontroller

1. **Higher cost compared to 8-bit microcontrollers.** One of the biggest drawbacks of STM32 microcontrollers is that they are generally more expensive than 8-

bit microcontrollers, which can make them less accessible for hobbyists and low-budget projects.

2. **Higher power consumption compared to 8-bit microcontrollers.** STM32 microcontrollers typically have higher power consumption compared to 8-bit microcontrollers, which can be a concern for battery-powered applications. However, this is often offset by their higher processing power and more advanced features.

3. **Steep learning curve for beginners.** Due to their advanced capabilities and more complex architecture, STM32 microcontrollers can have a steep learning curve for beginners. They require a deeper understanding of programming and electronics compared to simpler microcontrollers like the Arduino, which can make them a more challenging option for beginners. However, there are a wealth of resources available online to help developers learn and get started with the STM32 platform.

2.4.3. Comparison of microcontrollers

Microcontrollers are crucial components in a wide range of devices and systems, from consumer electronics to industrial automation. When choosing a microcontroller, it's essential to consider factors such as processing power, memory, power consumption, and development tools. According to the table 1, I will consider each microcontroller:

Table 1.

Microcontroller technical characteristics[1,2,3]

Microcontroller	**Atmel 8-bit AVR**	**RP2040**	**STM32**
Architecture	8-bit/32-bit	Dual-core ARM Cortex-M0+	ARM Cortex-M0/M3/M4/M7
Clock Speed	Up to 32 MHz	Up to 133 MHz	Up to 480 MHz
Flash Memory	Up to 256 KB	Up to 16 MB	Up to 2 MB
SRAM	Up to 16 KB	Up to 264 KB	Up to 640 KB
Operating Voltage	1.8V to 5.5V	1.8V to 3.3V	1.7V to 3.6V
ADC Resolution	Up to 10-bit	Up to 12-bit	Up to 16-bit

Digital I/O Pins	Up to 86	Up to 30	Up to 168
Communication	USART, SPI, I2C, USB	UART, SPI, I2C, USB	USART, SPI, I2C, USB, Ethernet
Development Tools	Atmel Studio, Arduino IDE	Raspberry Pi Pico C/C++ SDK	STM32CubeIDE, Keil uVision
PWM Channels	6x8bit or 2x16bit	6x16bit	17x16bit
Timers	Up to 3x16-bit timers with up to 3x output/input channels	Up to 2x16-bit timers with up to 4x output/input channels	Up to 14x16-bit timers with up to 4x output/input channels
Watchdog Timer	Yes	Yes	Yes
Sleep Modes	Yes	Yes	Yes
Temperature Range	-40°C to 85°C	-20°C to 85°C	-40°C to 85°C
Bootloader Support	Yes	Yes	Yes

The Atmel AVR microcontroller is an 8-bit device that is simple to program and consumes less power, making it well-suited for low-power and battery-powered applications. Although it has a smaller memory and processing power compared to the STM32, the Atmel AVR is a versatile device with a wide range of capabilities that make it a popular choice for hobbyists and beginners.

The RP2040 microcontroller is a recent addition to the market that has gained popularity due to its low cost and impressive performance. It features a dual-core processor and a range of peripherals, making it suitable for hobbyists and low-cost applications. However, it has some limitations in terms of memory and advanced features such as USB connectivity.

The STM32 microcontroller is a 32-bit device based on the Arm Cortex-M processor architecture, offering powerful processing capabilities and a rich set of features that make it suitable for a wide range of applications, including industrial automation and robotics. It is a more expensive and complex device than the Atmel AVR and RP2040, but its advanced capabilities and wide range of compatible development tools and programming languages make it a popular choice for professionals.

Moreover, the STM32 microcontroller family includes a range of models, each with different specifications and capabilities. The STM32F7, for example, is a high-performance device with a floating-point unit, advanced graphics capabilities, and a range of connectivity options, making it ideal for demanding applications such as medical devices and aerospace systems.

Choice. Selecting the right microcontroller is a critical decision that can have a significant impact on the performance and functionality of a device or system. The Atmel AVR, RP2040, and STM32 microcontrollers are all popular choices with their own strengths and limitations. The Atmel AVR is a low-power and simple-to-program device that is suitable for beginners and low-power applications, while the RP2040 offers impressive performance at a low cost, making it an excellent choice for hobbyists and low-cost applications. The STM32 is a powerful and versatile device that is well-suited for professional applications, but it has a steeper learning curve and is more expensive than the other two options.

One additional consideration when choosing a microcontroller is the availability of support and resources. All three microcontrollers have a large user base and a range of development tools and resources available online. However, the STM32 has a particularly extensive ecosystem with a broad range of resources, including official documentation, forums, and development tools. This extensive ecosystem can be particularly helpful for developers who are new to the platform or working on complex projects. Ultimately, the choice of microcontroller depends on the specific requirements and goals of a project, and developers should carefully consider their options and choose the microcontroller that best suits their needs.

CHAPTER 3. System design

3.1. Hardware

The hardware section of the remote temperature and humidity monitoring system is a critical aspect of the project. In this section, we will outline the various components used to build the system, including the microcontroller board, sensors, power supply, and other necessary hardware.

The microcontroller board is an essential part of the system, providing the computational power necessary to perform temperature and humidity measurements and to transmit data wirelessly to a remote location. The chosen microcontroller board should be compatible with a range of sensors and modules, be easy to use, and have an affordable price point.

Sensors play a crucial role in the remote temperature and humidity monitoring system. The sensors used in the system should be capable of accurately measuring temperature and humidity levels and have a wide measurement range to account for a range of different environments. Additionally, sensors should be compatible with the chosen microcontroller board.

To power the system, we can use a variety of sources, including batteries or DC adapters. The selected power supply should be compatible with the microcontroller board and provide a stable voltage output to prevent damage to the system components.

The final hardware design of the remote temperature and humidity monitoring system will also require additional components such as breadboards, jumper wires, and USB cables for prototyping and programming. These components will be assembled together to form the final hardware design of the remote temperature and humidity monitoring system.

3.1.1. Microcontroller board

After researching different microcontroller platforms, I determined that Arduino Uno was the most suitable option for my project due to its versatility, ease of use, and affordability[4].

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- PWM Digital I/O Pins: 6
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB (ATmega328P) of which 0.5 KB used by bootloader
- SRAM: 2 KB (ATmega328P)
- EEPROM: 1 KB (ATmega328P)
- Clock Speed: 16 MHz

Arduino Uno operates on a 5V power supply and can accept input voltages between 6 and 20V. The board has 14 digital I/O pins and 6 analog input pins, which will allow me to interface with the humidity and temperature sensor as well as other components such as LEDs, LCD displays, and wireless modules. Each I/O pin can provide up to 20 mA of current, while the 3.3V pin can provide up to 50 mA[4].

The board also features a 16 MHz quartz crystal oscillator and a USB connection for programming and power. It has 32 KB of flash memory for storing code (of which 0.5 KB is used by the bootloader), 2 KB of SRAM for storing variables, and 1 KB of EEPROM for storing data that needs to be retained even when the power is turned off[4].

3.1.2. Sensor

As a crucial component of the system, sensors capture environmental data and are responsible for measuring temperature and humidity accurately. By choosing the right sensors, I can design a robust and reliable monitoring system that can collect and transmit data wirelessly for analysis and interpretation.

Table 2.

Sensor's characteristics[5,6,7,8]

Sensor	**DHT11**	**DHT22**	**DS18B20**	**BME280**
Temperature Range	0-50°C	-40-125°C	-55-125°C	-40-85°C
Humidity Range	20-80% RH	0-100% RH	N/A	0-100% RH
Accuracy	±2°C	±0.5°C	±0.5°C	±1°C
Interface	Digital	Digital	Digital	I2C/SPI
Power Consumption	1-5 mA	1-2.5 mA	1-1.5 mA	2.7 μA (standby)
Price	Low	Medium	Low	High

I will use DHT22 sensor, since it offers the widest range of temperature measurement (-40-125°C) and a high accuracy of ±0.5°C compared to the other sensors[6].

It also offers a wide range of humidity measurement (0-100% RH) and has a low power consumption of 1-2.5 mA, which is a good choice for remote monitoring applications where battery life is crucial.

3.1.3. Resistor

I have chosen to use a 10kΩ resistor for the DHT22 sensor's data line. The reason for this choice is that the DHT22 sensor requires a pull-up resistor to maintain a constant voltage level at the data line.

The pull-up resistor helps to ensure that the signal transmitted by the sensor remains stable and reliable, even in noisy environments. Additionally, by using a 10K ohm resistor, we can limit the amount of current flowing through the sensor's data line, protecting it from damage due to excess current flow.

Furthermore, the 10kΩ ohm resistor is a common value that is readily available and widely used in various electronic circuits, making it an easy, convenient and extendable choice for the system.

I can ensure reliable data transmission and protect the sensor from damage, contributing to the overall performance and reliability of the remote temperature and humidity monitoring system.

3.1.4. Wireless communication module

For my project that requires wireless connectivity, a Wi-Fi module is a good choice. There are many different Wi-Fi modules available for the Arduino platform, ranging from simple and affordable to more complex and feature-rich. I will consider the most used.

Table 3.

Wi-Fi modules characteristics[9,10]

Feature	**ESP8266**	**ESP32**
Microcontroller	Tensilica L106 32-bit	Xtensa dual-core 32-bit LX6

	RISC	
Clock Speed	80 MHz	Up to 240 MHz
Wi-Fi	802.11 b/g/n	802.11 b/g/n and Bluetooth 4.2
GPIO Pins	Up to 17	Up to 36
Analog Inputs	1	Up to 18
ADC Resolution	10-bit	12-bit
SPI	2	4
I2C	1	2
UART	1	3
Memory	64 KB instruction RAM, 96 KB data RAM	520 KB SRAM, 4 MB flash
Active Power Consumption	Around 80 mA	Range from around 70 mA to 240 mA

I have selected the ESP8266 wireless communication module for my system design, because it offers a cost-effective and simple solution that is compatible with the Arduino platform. Additionally, the ESP8266 has a relatively small form factor, which is important for the space limitations of my project. The module also offers lower power consumption, which is desirable for my project's requirements. Finally, my project does not require the higher processing power or memory of the ESP32, so the ESP8266 is a practical and effective choice for my needs.

3.1.5. Power supply

I selected a 5V DC power supply, because the Arduino Uno can be powered through the USB port or the DC jack with a voltage range of 7-12V, and the ESP8266 and DHT22 can be powered directly from the 5V pin on the Arduino.

By using a 5V DC power supply, I can ensure that all components in the setup receive the required voltage for their operation. It's also important to ensure that the power supply can provide sufficient current for all the components to operate properly.

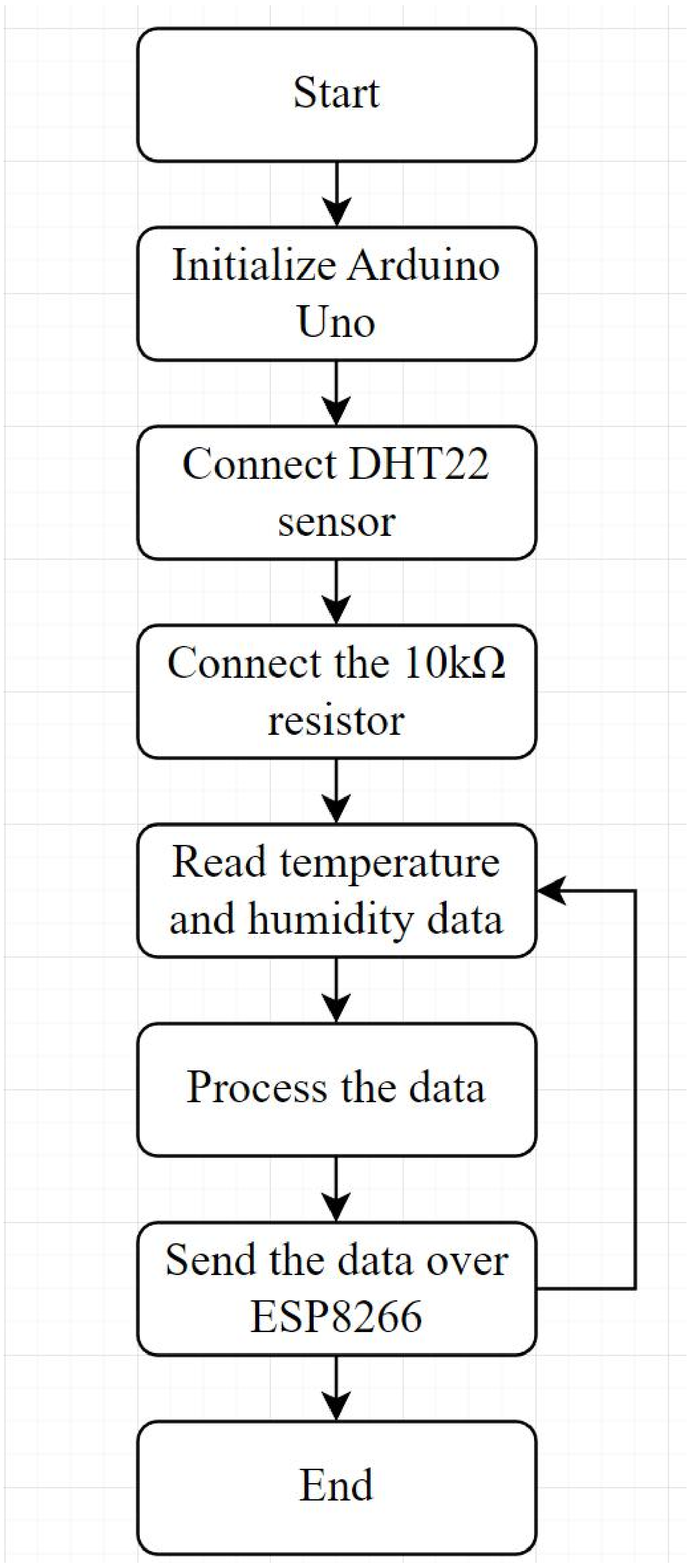
3.2. Prototyping

3.2.1. Block Schema

To better visualize and plan my project, I have decided to create a block schema using a tool Tinkercad.

In my block schema (Pic. 3.2.1.), I will start with the initialization of the microcontroller, followed by the connection of the DHT22 sensor to the breadboard. From there, I will proceed to read and process the data, which will then be saved and displayed on a screen. To ensure that the data is displayed accurately, I will include a recursive loop that continuously updates the display as new data is collected.

By creating a block schema, I can more easily visualize the flow of my project and identify any potential issues or areas that may require further attention. It will also serve as a helpful reference throughout the project, enabling me to keep track of my progress and ensure that all necessary components are properly connected and functioning.


Start
Initialize Arduino Uno
Connect DHT22 sensor
Connect the 10kΩ resistor
Read temperature and humidity data
Process the data
Send the data over ESP8266
End

Pic. 3.2.1. Block-schema

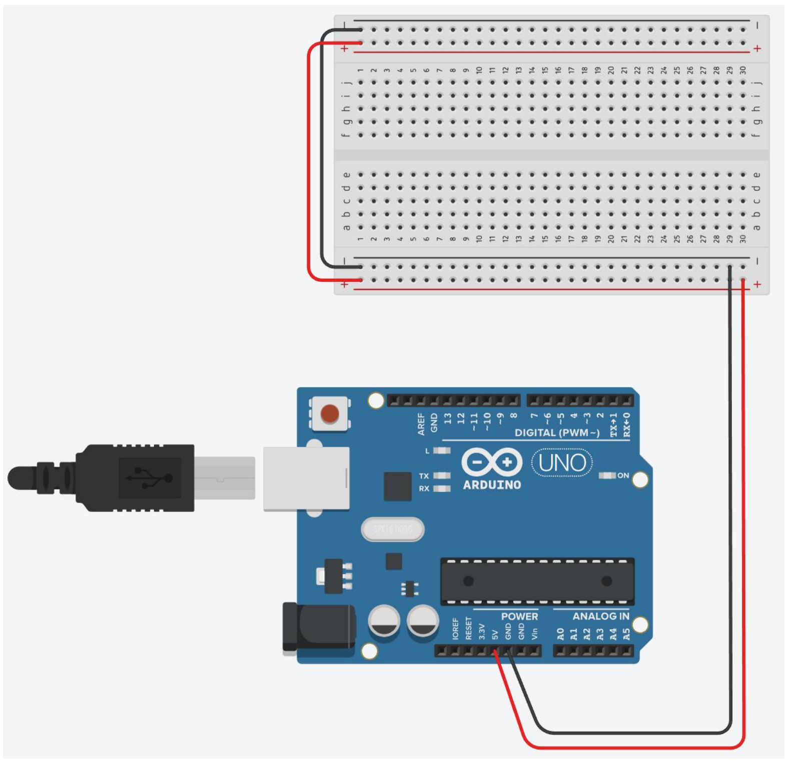
3.2.2. Modeling

The next step after creating a block-schema is to start the modeling process. Modeling is the process of creating a virtual representation of a physical system or process.

For modeling, I chose Tinkercad as my modelling tool because it is a free and easy-to-use online platform for designing and simulating circuits. It has a wide range of components and tools that make it suitable for both beginners and advanced users. Additionally, Tinkercad has a 3D design feature that allows me to create and visualize my circuits in a more realistic way.

First of all, I will set up a basic breadboard connection (Pic. 3.2.2.)[16].

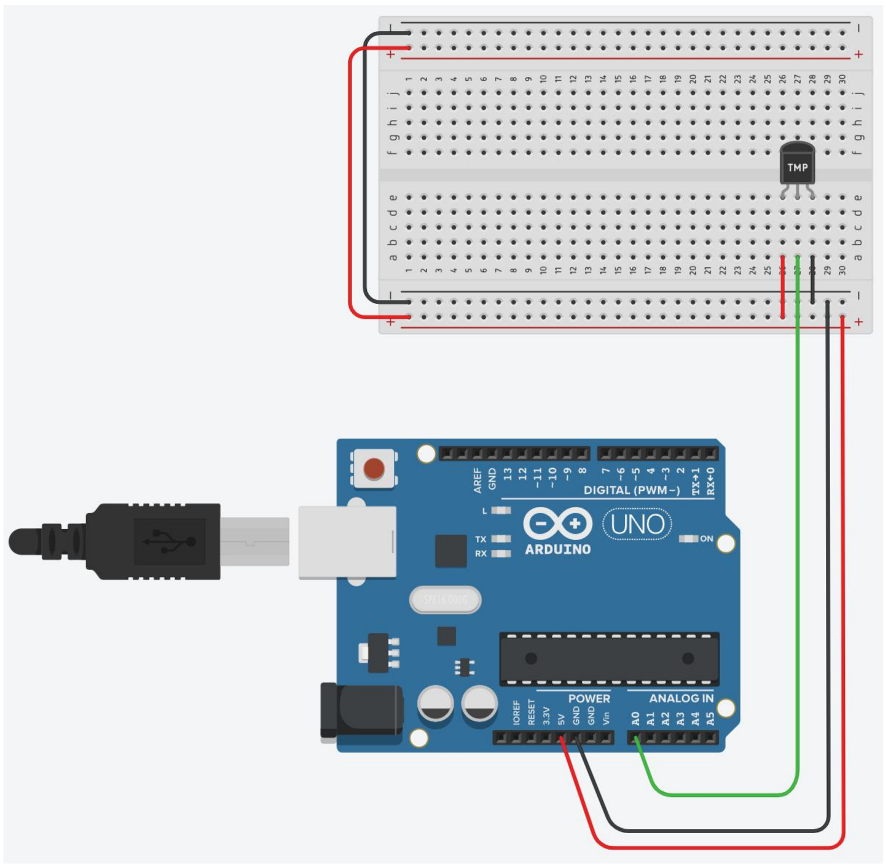
I will connect the power and ground rails of the breadboard to the 5V and GND pins of the Arduino Uno. To do this, I will connect the 5V pin of the Arduino Uno to the positive (+) rail of the breadboard using a red jumper wire, and I will connect the GND pin of the Arduino Uno to the negative (-) rail of the breadboard using black jumper wire. This will allow me to power all of my components from the same power source.



Pic 3.2.2. Connect the power and ground to Arduino Uno.

After connecting the power and ground rails of the breadboard to the Arduino Uno, I will connect the DHT22 sensor to the breadboard (Pic. 3.2.3.) [16].

The DHT22 sensor has three pins: VCC, GND, and Data. I will connect the VCC pin to the positive (+) using a red jumper wire, and I will connect the GND pin to the negative (-) rail using black jumper wire. Finally, I will connect the Data pin of the DHT22 sensor to a digital pin on the Arduino Uno using a green jumper wire. This will allow the Arduino Uno to read the temperature and humidity data from the DHT22 sensor.

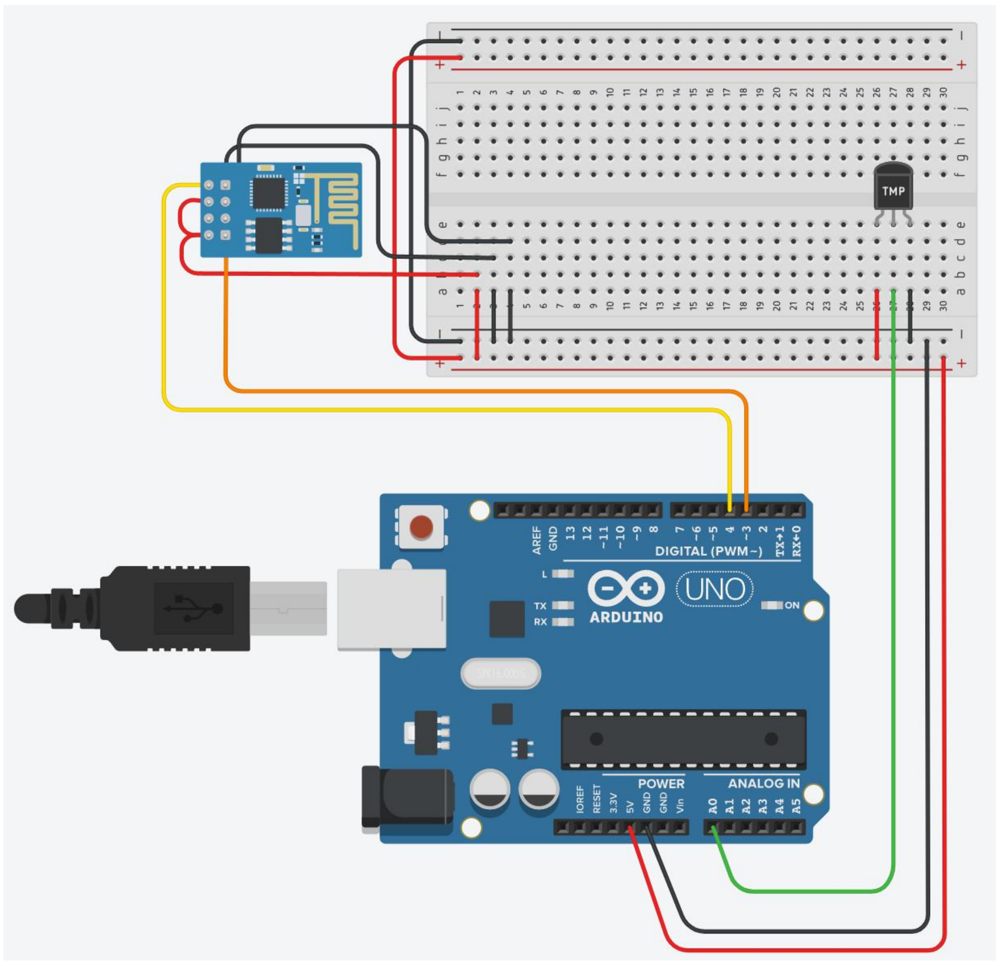


Pic 3.2.3. Connect DHT22 sensor to Arduino Uno.

Also, I will connect the ESP8266 Wi-Fi module (Pic. 3.2.4.).

The ESP8266 module has six pins[9,15]: VCC, GND, CH_PD, RX, TX, and GPIO0.I will connect the VCC pin to the positive (+) rail red jumper wire, the GND pin to the negative (-) rail using black jumper wire.

The RX pin is connected to D3 pin on the Arduino Uno using orange jumper wire, and the TX pin to D4 using yellow jumper wire. Finally, I will connect the GPIO0 pin of the ESP8266 module to the negative (-) rail using black jumper wire.



Pic 3.2.4. Connect ESP8266 Wi-Fi module.

Having completed the block-schema and the modeling in Tinkercad, I have a clear understanding of how the different components of my project will be connected and work together. And I can now proceed with writing the code, using the information and connections established in the block-schema. This will allow me to effectively and efficiently translate the design of my system into a functional and practical implementation.

3.2.3. Programming

I will write the code for the remote temperature and humidity monitoring system using C++. I have chosen to use C++ because it is a powerful and widely used programming language that is well-suited for embedded systems and real-time applications. Additionally, C++ supports object-oriented programming (OOP) and the model-view-controller (MVC) design pattern, which will allow me to write clean and modular code[11,12,13].

Object-oriented programming (OOP)[12] is a programming paradigm that revolves around the concept of "objects", which are instances of classes that encapsulate data and methods. In OOP, everything is an object, and the data and methods of an object are tightly bound together. OOP is a powerful tool for designing complex systems, as it allows developers to organize their code into reusable, modular components.

One of the key features of OOP is inheritance, which allows classes to inherit properties and methods from other classes. This makes it easy to create new classes that are based on existing ones, and helps to avoid duplicating code. Another important feature of OOP is polymorphism, which allows objects of different classes to be treated as if they were of the same class. This makes it possible to write more generic code that can work with different types of objects.

The Model-View-Controller (MVC) pattern[14] is a popular architectural design pattern used in software engineering that separates an application into three interconnected components: the model, the view, and the controller. The model represents the data and the logic of the application, the view presents the data to the user, and the controller receives user input and updates the model and the view accordingly. This separation of concerns allows for better organization, maintainability, and testability of the codebase. In addition, it facilitates the development of user interfaces that can be easily customized or swapped out without affecting the underlying data or business logic.

The Model component in the MVC pattern is responsible for managing the data and logic of the application[14]. This includes retrieving and storing data, processing data, and performing calculations. The Model component should be decoupled from the

View and the Controller components, meaning that it should not be aware of how the data is being presented to the user or how the user interacts with the application. This ensures that the Model component can be reused or modified without affecting the other components of the application. Overall, the MVC pattern promotes separation of concerns, modularity, and maintainability in software development.

I will start from creating model.h amd model.cpp files.

In the context of MVC, the model represents the data and logic of the application. The model.h file contain the class definitions and declarations for the data structures and functions that handle the data. The model would interact with the controller to receive input and update the view to reflect the changes in the data.

model.h

```
#ifndef MODEL_H
#define MODEL_H

#include <DHT.h>

class Model {
  public:
    Model();
    float getTemperature();
    float getHumidity();
    void updateData();
  private:
```

```
  DHT dht;

  float temperature;

  float humidity;

};


#endif
```

model.cpp

```
#include "Model.h"


#define DHTPIN 3        // Digital pin connected to the DHT sensor

#define DHTTYPE DHT22     // DHT 22  (AM2302), AM2321


Model::Model(): dht(DHTPIN, DHTTYPE) {

 dht.begin();

}


void Model::updateData() {

 temperature = dht.readTemperature();

 humidity = dht.readHumidity();

}
```

```
float Model::getTemperature() {
  return temperature;
}


float Model::getHumidity() {
  return humidity;
}
```

Next are controller.h and controller.cpp files.

controller.h is a header file that contains the declaration of the Controller class. It includes the headers for the Model, View, ESP8266WiFi, and WiFiClient libraries. It also defines the constants for the network credentials and server details. The Controller class has two private members: an instance of the Model class and an instance of the View class, and a private member function connectToWiFi() and sendDataToServer(). The public member function is run(), which is responsible for running the main loop of the program.

controller.cpp is the implementation file for the Controller class. It includes the header file for the Controller class and defines the constructor for the Controller class. It also defines the private member functions connectToWiFi() and sendDataToServer(), which are responsible for connecting to WiFi and sending data to the server, respectively. The run() member function is responsible for running the main loop of the program. It calls the updateData() method of the Model object to update the temperature and humidity data. It then gets the temperature and humidity data from the Model object and displays it on the serial monitor using the View object. Finally, it sends the temperature and humidity data to the server using the sendDataToServer() method and waits for 5 seconds before repeating the loop.

controller.h

```
#ifndef CONTROLLER_H
#define CONTROLLER_H

#include "Model.h"
#include "View.h"
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

// Replace with your network credentials
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

// Server details
const char* serverName = "your_SERVER_NAME";
const int serverPort = 80;

class Controller {
  public:
    Controller();
    void run();
  private:
    Model model;
    View view;
    WiFiClient client;
    void connectToWiFi();
    void sendDataToServer(float temperature, float humidity);
};

#endif
```

controller.cpp

```
#include "Controller.h"

Controller::Controller() {}

void Controller::connectToWiFi() {
  // Connect to WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");
}

void Controller::sendDataToServer(float temperature, float humidity) {
  if (client.connect(serverName, serverPort)) {
    // Format data as HTTP GET request
    String httpRequest = "GET /addData?temp=";
    httpRequest += String(temperature);
    httpRequest += "&humidity=";
    httpRequest += String(humidity);
    httpRequest += " HTTP/1.1\r\n";
    httpRequest += "Host: ";
    httpRequest += serverName;
    httpRequest += "\r\n";
    httpRequest += "Connection: close\r\n\r\n";

    // Send HTTP GET request to server
    client.print(httpRequest);
    Serial.println("Data sent to server");
  } else {
    Serial.println("Failed to connect to server");
  }
}

void Controller::run() {
  connectToWiFi();
  while(true) {
    model.updateData();
    float temperature = model.getTemperature();
    float humidity = model.getHumidity();
    view.displayData(temperature, humidity);
    sendDataToServer(temperature, humidity);
    delay(5000);
  }
}
```

Next is entrypoint of the system, main.cpp.

The main.cpp file simply includes the header files for the Controller and Model classes, and instantiates a Controller object. In the setup() function, the Serial object is initialized, and the run() method of the Controller object is called. The loop() function is left empty, as there is no need to do anything else in the main program loop.

main.cpp

```
#include "Controller.h"
#include "Model.h"



// Instantiate the Controller object
Controller controller;

void setup() {
  Serial.begin(115200);
  // Call the run method of the Controller object
  controller.run();
}

void loop() {
  // Nothing to do in the main loop
}
```

Conclusion

The development of a remote temperature and humidity monitoring system is a significant contribution to ensuring the stable operation of various equipment in industries such as agriculture, food processing, and storage facilities. The system provides continuous and real-time data on temperature and humidity, which can be accessed remotely, allowing for timely intervention and problem-solving in case of any deviation from the desired values.

The system is designed to be reliable, accurate, and easy to use, and it consists of several components, including a sensor, a microcontroller, a Wi-Fi module. The sensor measures temperature and humidity, and the microcontroller processes the data and sends it to the cloud platform through the Wi-Fi module.

Project uses the newest technical stack, Arduino UNO, DHT22 with ESP8266. Programming of this module is possible in the Arduino IDE program, in which a user-friendly interface and speed are an advantage of this program.

The implementation of the remote temperature and humidity monitoring system can significantly reduce the risk of equipment damage, product loss, and operational downtime, leading to increased efficiency, cost savings, and improved product quality. Furthermore, the system can provide valuable insights into the performance of the equipment and the environment, enabling data-driven decision-making and continuous improvement.

In conclusion, the remote temperature and humidity monitoring system is a vital tool for various industries, providing real-time monitoring, alerts, and insights into the performance of equipment and environment. The system is easy to use, reliable, and accurate, and it can significantly reduce the risk of equipment damage, product loss, and operational downtime, leading to increased efficiency, cost savings, and improved product quality.

References

1. ATmega328P microcontroller datasheet. Digital version: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
2. RP2040 microcontroller datasheet. Digital version: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf
3. STM32 microcontroller datasheet. Digital version: https://www.st.com/resource/en/datasheet/stm32f205re.pdf
4. Arduino UNO datasheet. Digital version: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf
5. DHT11 Humidity & Temperature Sensor datasheet. Digital version: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf
6. DHT22 Humidity & Temperature Sensor datasheet. Digital version: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf
7. DS18B20 Humidity & Temperature Sensor datasheet. Digital version: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf
8. BME280 Humidity & Temperature Sensor datasheet. Digital version: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf
9. ESP8266 Wi-Fi module datasheet. Digital version: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf
10. ESP32 Wi-Fi module datasheet. Digital version: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
11. Eric Freeman, Bert Bates, Kathy, Elisabeth Robson. Head First Design Patterns. Publisher: O'Reilly Media; 1st edition (October 1, 2004).
12. Matt Weisfeld. Object-Oriented Thought Process. Publisher: Addison-Wesley Professional; 4th edition (March 13, 2013).

13. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. Publisher: Pearson; 1st edition (August 1, 2008).

14. Adam Freeman. Pro ASP.NET Core MVC. Publisher: Apress; 6th ed. edition (September 16, 2016).

15. Bala Divya. ESP8266: Step by Step Book for ESP8266 IOT, Arduino Nodemcu Dev Kit. Kindle Edition.

16. Massimo Banzi, Michael Shiloh. Getting Started With Arduino: The Open Source Electronics Prototyping Platform. Publisher: Make Community, LLC; 4th edition (March 22, 2022).