TECHNICAL UNIVERSITY OF CRETE

DIPLOMA THESIS

Design and Implementation of an Embedded System for Remote Access to a Microcontroller for University Laboratory Exercises.

Author:

Athanasios KALLIONTZIS

Thesis Committee:

Prof. Apostolos DOLLAS, THESIS SUPERVISOR

Assoc. Prof. Sotirios IOANNIDIS

Dr. Evripidis SOTIRIADIS

A thesis submitted in fulfillment of the requirements for the diploma of Electrical and Computer Engineer in the

School of Electrical and Computer Engineering

Microprocessor and Hardware Laboratory

September 9, 2023

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

Design and Implementation of an Embedded System for Remote Access to a Microcontroller for University Laboratory Exercises.

by Athanasios KALLIONTZIS

This thesis addresses the issue of accessibility and practical learning in embedded systems education, particularly in the context of the COVID-19 pandemic. The growing use of online education platforms has highlighted the importance of students being able to work with real hardware and code, rather than relying solely on simulations. To address this challenge, a custom remote lab system was developed specifically for embedded systems course at the Technical University of Crete. This system, based on the AVR microcontroller, enables students and faculty to remotely access equipment via the internet, facilitating hands-on experiments and laboratory tasks. By leveraging internet connectivity and the AVR microcontroller, the system provides an immersive and authentic learning experience that transcends traditional simulations. The successful implementation of this remote lab system at the university contributes to the improvement of accessibility and the quality of embedded systems education. It empowers students to develop practical skills even in remote learning environments, supporting their proficiency in embedded systems. This work aims to bridge the gap between theoretical knowledge and practical application, enhancing the overall learning experience for students at the Technical University of Crete.

Acknowledgements

First of all, I would like to express my deep appreciation to Professor Apostolos Dollas, who served as the supervisor of this thesis, for his invaluable help, guidance, and support. I am grateful for his unwavering assistance throughout this process and the unique opportunity to work on this subject under his supervision. I would also like to thank Mr. Sotirios Ioannidis and Mr. Evripidis Sotiriadis for serving as members of the examination committee. I would like to extend my gratitude to Mr. Markos Kimionis, responsible for the Microprocessors and Materials Laboratory, for his support in providing the necessary materials. Additionally, I would like to acknowledge all my friends who have been by my side and supported me throughout these years. Above all, I would like to express my heartfelt gratitude to my family for their unwavering support and guidance in every decision I have made. Thank you all.

Contents

Abstract iii

Acknowledgements v

Contents vii

List of Figures ix

List of Tables xi

List of Algorithms xi

List of Abbreviations xiii

1 Introduction 1
- 1.1 Thesis Outline . . . 2

2 Theoretical Background 3
- 2.1 Methods and Associated Technology for Remote Access to a Microcontroller of a Laboratory . . . 3
- 2.2 Programming and Control of the Laboratory Microcontroller . 10

3 Related Work 13
- 3.1 Previous Implementations . . . 14
  - 3.1.1 Design and Implementation of an Integrated System for Logic Design exercises . . . 14
  - 3.1.2 FPGA e-Lab . . . 17
  - 3.1.3 Wireless ATMEL AVR In-Circuit Serial Programmer based on Wi-Fi and ZigBee . . . 20
  - 3.1.4 Development and Application of Remote Laboratory for Embedded Systems Design . . . 25
  - 3.1.5 Remote Programming and Reconfiguration System for Embedded Devices . . . 26

3.2 Thesis Approach 28

4 Design and Implementation of the Platform **29**
4.1 Component Selection and Integration in Implementation . . . 32
4.1.1 Microcontroller 32
4.1.2 Wifi Module 35
4.1.3 I2C EXPANDER 16BIT I/O - MCP23017 38
4.1.4 Target-Microcontroller 40
4.2 Remote Access to the Laboratory of the University 43
4.3 Programming and Control of the Laboratory Microcontroller 47

5 System Verification and Evaluation **53**

6 Conclusions and Future Work **59**
6.1 Conclusions 59
6.2 Future Work 59

References **62**

A User Manual **67**

B Javascript code **69**

C Block Diagram **71**

List of Figures

2.1 The process diagram of Remote Desktop 5
2.2 Block diagram of an IoT module 7

3.1 The physical Interface "MHL Development Board" 15
3.2 The graphical interface "MHL Development Board Control" . 16
3.3 The flow of communication between the computer(GUI) and the "MHL Development Board" 16
3.4 FPGA e-Lab GUI 17
3.5 FPGA e-Lab schematic diagram 19
3.6 FPGA e-Lab Video Capture 20
3.7 System Block Diagram of Wireless Programmer. 21
3.8 PCB Design Schematic of Wireless Programmer. 22
3.9 Wireless Programmer with Wifi. 23
3.10 Wireless Programmer with ZigBee. 24
3.11 An overview of RELDES. 26
3.12 RPD architecture (a) and RPD abstraction layers diagram (b). . 27
3.13 Communication flow of RPD system. 27

4.1 The flow diagram of the main process of the microcontroller. . 30
4.2 Schematic diagram of the Platform. 31
4.3 Schematic diagram of ATmega32A-PU pins. 35
4.4 Schematic diagram of ESP-12E Pins. 36
4.5 Connection between ESP-12E and ATmega32A-PU. 38
4.6 The MCP23017 integrated circuit. 39
4.7 Connection between I/O Expander and ATmega32A-PU. . . . 40
4.8 Schematic diagram of ATmega328p pins. 42
4.9 Connection between ATmega328p and ATmega32A-PU. . . . 43
4.10 Login web-page Developed in this Thesis. 45
4.11 Main Web Page Developed in this Thesis. 46
4.12 SPI interface. 48

5.1 Visual representation of programmer process logs. 55

5.2 Real-time Programming Image. 56
5.3 Webpage Interface during Programming Process. 57

C.1 Block diagram of the Platform. 71

List of Tables

4.1 ISP commands(0) 49
4.2 ISP commands(1) 50
4.3 ISP commands(2) 50

List of Abbreviations

ALU	**A**rithmetic **L**ogic **U**nit
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
BRAM	**B**lock **R**andom **A**ccess **M**emory
CPU	**C**entral **P**rocessor **U**nit
CS	**C**omputer **S**cience
DDR4	**D**ouble **D**ata **R**ate type texbf4 memory
DRAM	**D**ynamic **R**andom **A**ccess **M**emory
DSP	**D**igital **S**ignal **P**rocessor
FF	**F**lip **F**lops
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
GDDR6	**G**raphics **D**ouble **D**ata **R**ate type **6** memory
GPU	**G**raphic **P**rocessor **U**nit
HBM	**H**igh **B**andwidth **M**emory
HDL	**H**ardware **D**escription **L**anguage
HLS	**H**igh **L**evel **S**ynthesis
HPC	**H**ight **P**erformance **C**omputing
LUT	**L**ook **U**p **T**able
MPSoC	**M**ulti **P**rocessor **S**ystem **o**n **C**hip
PL	**P**rogrammable **L**ogic
PS	**P**rocessing **S**ystem
RAM	**R**andom **A**ccess **M**emory
SDK	**S**oftware **D**evelopment **K**it
SIMD	**S**ingle **I**nstruction **M**ultiple **D**ata
SSE	**S**treaming **S**IMD **E**xtensions
SSD	**S**olid **S**tate **D**rive
TDP	**T**hermal **D**esign **P**ower
URAM	**U**ltra **R**andom **A**ccess **M**emory
USD	**U**nited **S**tates **D**ollar

Chapter 1

Introduction

A problem that has become increasingly prevalent in laboratories, particularly during the pandemic, is the accessibility of education platforms. To effectively learn, students must be able to access and work with actual code that is downloaded onto real hardware, and not just simply by simulation. In light of this, a system was developed as part of this thesis to be used in embedded systems courses. While there are already existing academic and commercial systems available, the goal of this thesis is to develop our own system, ensuring that the necessary expertise is available at the university. It is worth noting that these types of systems are often referred to as Remote Labs. Precisely, this thesis was based on the thesis of Emmanouil Lykos, who implemented an embedded system for Logical Design exercises through a graphical interface via computer, with the aim to implement a platform that will enable an efficient integration in the laboratory of the Technical University of Crete and will utilize more efficiently the remote access of the platform. The key alteration entailed integrating a WiFi module to facilitate communication between the platform and the graphical interface, replacing the earlier USB connection. This change significantly enhanced remote platform access and fostered unrestricted future adaptations as well as its integration and utilization within the laboratory.

1.1 Thesis Outline

The thesis consists of the following chapters:

- **Chapter 2: Theoretical Background,** which refers to current technology and various approaches we can choose according to them.
- **Chapter 3: Related Work,** in which are presented similar implementations that have been carried out in the past.
- **Chapter 4: Design and Implementation of the Platform,** in which the implementation of this thesis is described.
- **Chapter 5: Results,** which presents the verification of the correct functioning of the implementation of this thesis.
- **Chapter 6: Conclusions and Future Work,** which analyses whether the initial objectives have been achieved as well as future extensions of the implementation that can be made.

Chapter 2

Theoretical Background

In this chapter, we're going to explore the basic ideas that support the technology allowing us to access labs from far away. Being able to control labs remotely is really important for modern research. It lets us control things, collect data, and keep an eye on things, even if we're not physically there. Thanks to technology, we have many ways to do this, each with its own pros and cons. Deciding on the best way to remotely access a lab involves thinking about things like what the lab needs, how much it costs, how secure it is, if it's easy to fix when things go wrong, and how well it works.

2.1 Methods and Associated Technology for Remote Access to a Microcontroller of a Laboratory

Remote access to a laboratory is a crucial aspect in modern research and experimentation. It allows for remote monitoring, control, and data collection from a laboratory setting, regardless of the physical location of the researcher. However, with the advancement of technology, there are now several methods available for achieving remote access to a laboratory. These methods have varying degrees of effectiveness and efficiency, and it is important to choose the appropriate method based on the specific needs and requirements of the laboratory.

The following factors should be considered when choosing a method for remote access to a laboratory:

- The way a laboratory is conducted and the requirements it has. This includes the tools that will be used, the time that each student will need to practice, and the amount of time that the lab will be accessible. It is also important to consider the level of technical expertise required for

the method, as well as the compatibility with existing equipment and infrastructure.

- The total cost, as a system in addition to its performance also depends on how affordable it is in a lab budget. Therefore the combination of efficiency and low cost is optimal. It is also important to consider the long-term costs, such as maintenance and upgrades, as well as the potential for cost savings through automation and streamlined processes.
- System security. Whether it is the physical space of the laboratory or the protection of the system from unauthorized persons, security is an important factor in evaluating a tool. This includes both the protection of sensitive data and equipment, as well as the ability to detect and respond to security breaches.
- Debug support. The ability of the engineer to debug the system is important. In the event of a subsystem failure, locating and repairing it is a necessity as otherwise the entire system will need to be replaced, which may not be possible or will affect the laboratory's expenses. Also, the ability to modify or extend the functionality is one of the features that all systems must have. This includes the availability of diagnostic and troubleshooting tools, as well as the level of technical support provided by the vendor.
- System performance. Finally, system performance is one of the main factors. When a real-time control is an initial goal, the speed with which the communication with the laboratory takes place must be at the desired limits as well as the control of the laboratory's equipment. This includes factors such as latency, bandwidth, and reliability, as well as the ability to handle high volumes of data and concurrent users.

The main tools and techniques used to remotely access a laboratory are analyzed below:

PC + Remote Desktop: Remote access to a laboratory can be achieved using a combination of PC and remote desktop software. This technique allows users to remotely control school, university, or district computers and virtual machines (VMs) in real-time from any device. The remote desktop software provides a simple and secure way for users to access the screen of the remote computer on their own device. This, combined with an application that communicates with the development board controlling the laboratory equipment, offers a solution for remote access to a lab. There are many remote desktop applications[1][2] available online to choose from, depending on the specific needs of the implementation. The process of how remote desktop access is performed is illustrated in Figure 2.1 below. It is important to note that remote desktop software is particularly useful when dealing with educational institutions where it is not possible to transfer specific hardware and software tools to the cloud. Additionally, the remote desktop software provides a high level of security and allows for real-time control and access to the laboratory equipment.

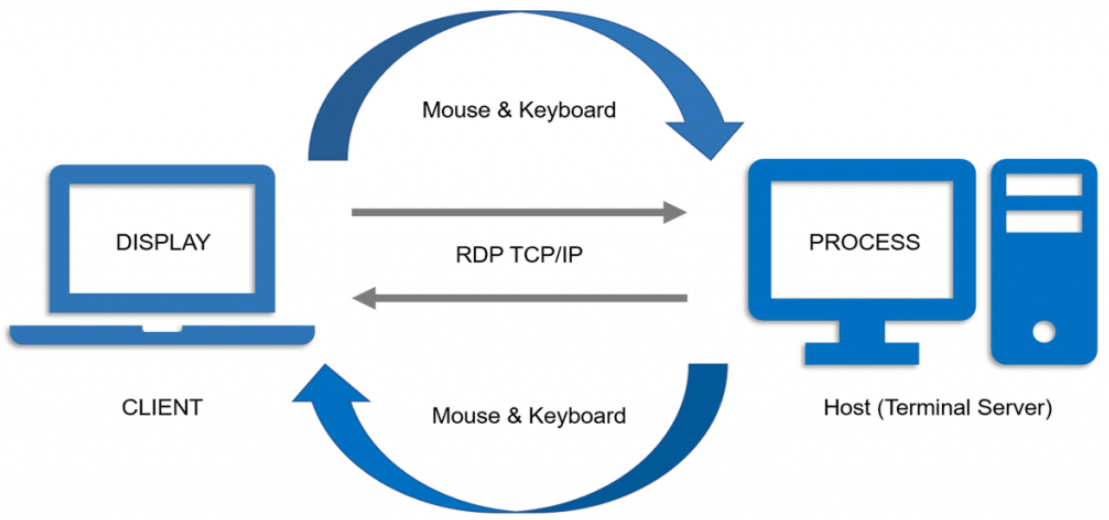

FIGURE 2.1: The process diagram of Remote Desktop.[Source: CyberlinkASP. "Remote Desktop Work." CyberlinkASP Insights, [3]]

PC + VCP: A virtual serial port (VCP) can be used to send and receive serial data over a network such as a LAN or the Internet. When using a serial port emulator, data generated from a serial application or device is converted from serial data to information that can be transferred over a network. A virtual serial port provides all of the functionality of a physical COM interface. Data is converted in both directions to allow network transmission and then turned back to serial signals for interaction with devices and programs. A network serial port server solves the problem of attaching to serial peripheral equipment that is not within the range of a direct connection. Employing serial over IP network technology lets users access your serial devices no matter where they are without being physically connected to them. There are many software development platforms that take advantage of virtual ports to achieve remote access to the laboratory. Implementations based on this technology have been developed[4] [5] [6].

Wifi Module and IoT: Internet of Things (IoT) is a rapidly growing technology that enables devices to be connected to the internet and exchange data with each other. In the field of Remote Labs, IoT technology is being used to create a connected environment of embedded systems that allows for remote access to laboratory equipment over the web.

A traditional remote laboratory system typically involves a full-scale computer system along with associated interfacing and web hosting technologies. However, the initial commissioning and subsequent maintenance of a remote laboratory system can be costly and time-consuming. To address this issue, the implementation of a remote laboratory facility using an embedded processor that is enabled to access wifi via IoT has been an area of focus.

The IoT embedded processor performs the functions of both the computer and server used in traditional remote laboratories. This allows for a more cost-effective and efficient solution for remote laboratory access. Each IoT module consists of a processing module that hosts the CPU and RAM, a network module that hosts a wireless transceiver (Tx/Rx circuit and a Tx power amplifier), and a function module that provides interfaces to a set of supported sensors and actuators. The module also includes a power management unit, which controls the power supply and charging of the module.

One of the key advantages of using an IoT module for remote laboratory access is its ability to support a wide range of sensors and actuators.

This allows for a wide range of experiments to be performed remotely, including those that involve temperature, humidity, pressure, and other types of data. Additionally, the use of an IoT module allows for remote access to laboratory equipment from any location, as long as there is an internet connection.

In conclusion, the use of an IoT module for remote laboratory access provides a cost-effective and efficient solution for remote laboratory access. It allows for a wide range of experiments to be performed remotely and provides remote access to laboratory equipment from any location, as long as there is an internet connection and actuators. There are several papers that evaluate such technology [7], [8] , [9].

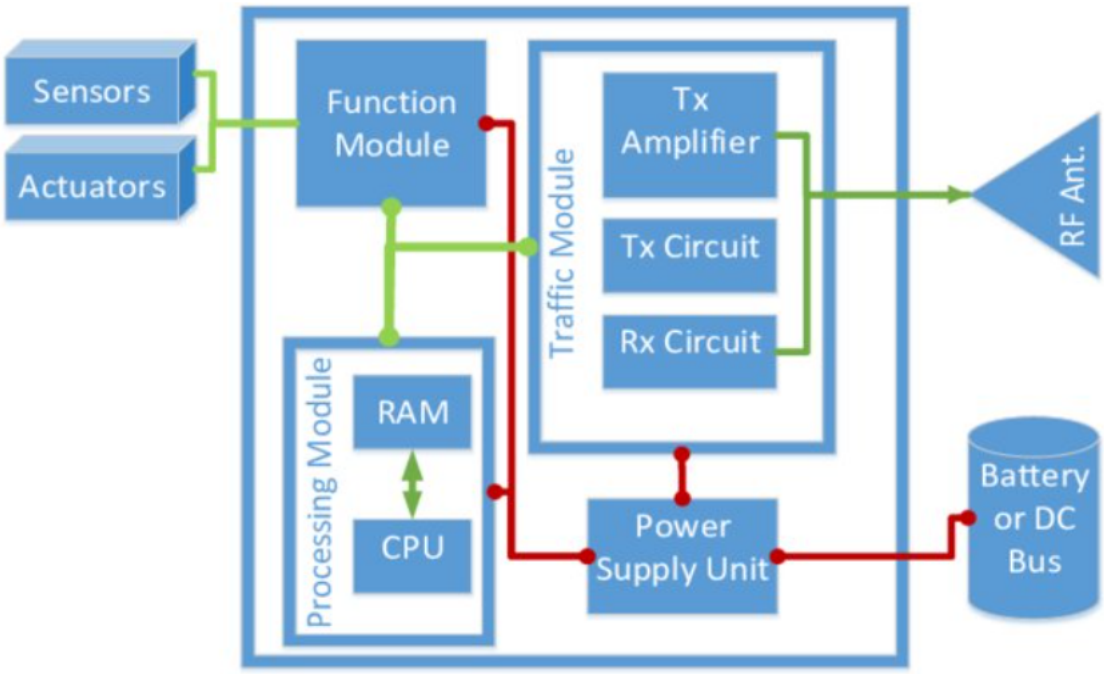

FIGURE 2.2: Block diagram of an IoT module. [Source: Energy efficient service embedding in IoT networks [10]]

There are two main categories of WiFi modules for IoT: A "single" solution where the MCU runs the WiFi stack and the host application in one chip and a "host processor + WiFi module" solution where the wireless connectivity solution contains the WiFi stack, and a separate processor runs the host application. What you will need to consider to select the right wifi module is shown below:

- IoT architecture:An important decision you must make is whether the solution should be a “single” or “host processor + WiFi module” solution. In a “single” solution, the IoT MCU runs the WiFi stack and the host application on a single chip. This configuration is an ideal fit for embedded devices where physical layout size is a main priority as it eliminates several external components that are now all integrated. Since the components can be shared in the wireless MCU configuration, this leads to fewer components, which helps to simplify PCB routing and layout considerations. However, this type of solution may not be as flexible as a "host processor + WiFi module" solution in terms of upgradability and future expansion. On the other hand, a “host processor + WiFi module” solution is where the wireless connectivity solution contains the WiFi stack, and a separate processor runs the host application. This architecture is ideal for well-defined and fully matured WiFi technology that does not change frequently. It allows for minimal WiFi overhead on the MCU and simplifies the code development effort since developers can rely on the packaged WiFi stack. The separation of the WiFi stack and the application layer allows the MCU to invest maximum hardware resources and bandwidth in the IoT application. Additionally, this type of solution allows for easier upgradability and expansion in the future as the WiFi module can be replaced or upgraded separately from the host processor.
- WiFi Protocol Support: The IEEE 802.11 represents the family of wireless LAN standards that operate mostly in the unlicensed frequency bands. Today, there are many standards like 802.11 a/b/g/j/n/p/ac/ad/ah in use, and each standard has different specification requirements. There are three key factors to trade-off when selecting these protocols: data rate, range, and power requirements. Protocols like 802.11n, 802.11ac and 802.11ah have the advantage of a higher data rate for IoT multimedia applications such as streaming video and audio. Alternatively, protocols like 802.11b/g have the advantage in power requirements and are more suitable for battery-powered IoT devices. Therefore, as per the project requirements, the best IoT WiFI module for you depends on the demands and uses of your IoT application. It’s also worth noting that newer standards like 802.11ax (Wi-Fi 6) and 802.11ay (Wi-Fi 6E) are becoming available and offer even higher data rates and improved performance, but these may not be as widely supported by devices or networks yet.

- Operating Frequency: IEEE 802.11 standards operate in different specified frequency ranges and are divided into a multitude of channels. Countries have their own rules and regulations to determine the allowable channels, users, and maximum power levels to use within these frequency bands. These days, many WiFi modules come up with dual-band support in 5GHz and 2.4GHz and provide flexibility in how mobile IoT devices are deployed and managed. The 5GHz band generally provides a higher data rate and less interference, but the 2.4GHz band may have a longer range and better penetration through walls and other obstacles. When selecting a WiFi module, it's important to consider the operating frequency and regulations for the country or region where the device will be used.
- Secure WiFi Support: Today, the security of data transmission over the internet is a major challenge. Therefore, before allowing IoT devices to connect to a network using WiFi, it is important to make sure the WiFi supports the required security standards. All the best IoT WiFi modules support at least one of the various WiFi security standards like WPA, WPA2, WPA3, WPS, or others. Each security standard has its advantages and disadvantages, so it's important to select the module that best suits your security needs.
- Hardware Interfaces: Usually, WiFi modules are bundled with many I/Os and peripheral interface support to suit different needs. The USB, SPI, or SDIO interfaces are preferred to support high data throughput applications. Otherwise, the typical interface is through a UART, I2C, I2S and others.
- Countries have their own regulatory certifications and for IoT devices to enter those markets, they need to comply with those regulations. This can include certifications such as FCC (Federal Communications Commission) in the United States, CE (Conformité Européene) in Europe, and others. It is important to research the certifications required for the specific market that the IoT device will be sold in and ensure that the WiFi module used in the device meets those requirements.

Finally, there are numerous WiFi IoT modules from various manufacturers available on the market. Every module has different specifications and features. Therefore, it's important to go through the datasheet of each module before finalizing your decision.

RAL(Remote Access Laboratories) Systems: They are online environments for operating instruments and collecting measurement data over the Internet. Such systems are often deployed by Universities to support undergraduate students and generally follow the client-server paradigm. Different technologies and frameworks such as LabVIEW[11], Java, Asynchronous JavaScript and XML (AJAX), LAN extension for Instrumentation (LXI), etc., have been used to connect users with the RAL systems. All of these follow a client–server centric model. These systems aim to accommodate a higher number of students with limited resources and remove the time and space-related constraints. Such systems are widely used for teaching at universities, including automation, electronics, and control systems.

2.2 Programming and Control of the Laboratory Microcontroller

Once we have access to the laboratory, a development board is required to control the application located on the user's remote computer. The development board acts as the interface between the remote computer and the target microcontroller. Depending on the specific implementation, the user can choose to use either web design or GUI design for the application. Both web design and GUI design have their own benefits and drawbacks, but the main focus should be on achieving a functional result. The choice of application design should be based on what best fits the overall implementation.

The development board plays a crucial role in programming and controlling the target microcontroller. It allows the user to program the microcontroller, control its functions, and receive real-time feedback from the microcontroller. In essence, the development board acts as both a programmer and a debugger for the target microcontroller. Therefore, it is important to select an appropriate microcontroller that meets the specific needs of the implementation.

There is a wide range of microcontrollers available on the market, each with its own set of features and capabilities. When choosing a microcontroller, there are several factors to consider:

The development board has the function of programming the target microcontroller, its complete control as well as the feedback it will send in

real time to the application. In essence, the development board has the function of a programmer and debugger for the target microcontroller. Therefore, the appropriate microcontroller on which the development board will be based must be selected. There is a huge range of microcontrollers on the market that can meet the needs of any implementation. Factors to consider when choosing a microcontroller are the following:

- Hardware and Software architecture: The architecture of the microcontroller can impact its performance and capabilities. It is important to choose a microcontroller that has the appropriate architecture for the specific implementation.
- Processing power: The processing power of the microcontroller determines how quickly it can execute instructions. A microcontroller with higher processing power will be able to handle more complex tasks.
- Memory: The amount of memory available on the microcontroller can affect its ability to store and access data. It is important to choose a microcontroller that has enough memory to meet the needs of the implementation.
- Cost: The cost of the microcontroller can vary widely depending on its features and capabilities. It is important to choose a microcontroller that fits within the budget for the implementation.
- Hardware interface: The microcontroller should have the appropriate hardware interface to connect to the other devices and sensors in the implementation.
- Security: The microcontroller should have built-in security features to protect against unauthorized access and data breaches.
- Temperature tolerance: The microcontroller should be able to operate within a wide range of temperatures to ensure reliable performance in different environments.
- Power efficiency: The microcontroller should be designed to be energy-efficient, to minimize power consumption and prolong battery life.

When selecting a microcontroller for a laboratory implementation, it is important to take into consideration factors such as the hardware and software architecture, processing power, memory, cost, hardware interface, security, temperature tolerance and power efficiency. By carefully evaluating

each of these factors and determining which ones are most critical to the success of the implementation, you can select a microcontroller that will work effectively with the development board and meet the specific needs of the laboratory environment. Additionally, by taking the time to research and compare different microcontroller options, you can ensure that you are choosing the most appropriate one for your specific implementation and budget, and that the development board will be able to effectively program and control it.

Chapter 3

Related Work

The field of remote access to laboratory environments has been an active area of research for several years, as technology continues to advance at a rapid pace. Researchers have been working to leverage these developments to create a powerful tool for education, particularly in light of the challenges posed by the COVID-19 pandemic. The goal of these efforts is to provide students with the ability to remotely access laboratory equipment, in order to overcome the limitations of time and space that traditional laboratory settings impose.

In order to achieve this goal, researchers have had to adapt their methods to the new technological tools that are being developed, and carefully evaluate the usefulness of each tool in different contexts. This chapter presents an overview of past work in this area, highlighting the various approaches that have been taken and the challenges that have been encountered. Additionally, it provides a perspective on the current state of the field and the direction in which future research is likely to go, as well as how this thesis is related to the current research.

3.1 Previous Implementations

3.1.1 Design and Implementation of an Integrated System for Logic Design exercises

The M.Eng. thesis by Emmanouil Lykos at School of ECE, Technical University of Crete [6] presents the implementation of an embedded system to facilitate students in the preparation of experimental exercises of the "Logic Design" course laboratory. It was based on the way the experiments were conducted as well as on a previous approach[5]. The platform consists of two main parts: the physical interface “MHL Development Board "for the connection with the physical circuits of the laboratory and the graphical interface "MHL Development Board Control" on the lab-PC, to control the physical interface and debugging the laboratory circuits.

The physical interface presented in figure 3.1 is "responsible" for controlling the physical circuits of the laboratory and setting the clock of the synchronous circuits as well as their power supply. The platform with which development, control, and verification of the system took place is STM32F072B-DISCO of STMicroelectronic STM32F0 series which offers more than enough features at a low cost. The choice of the platform was made due to its STM32F0-72RB ARM microprocessor that provides USB Full-speed capability and contains all the necessary peripherals for communication. Also, buffers - line drivers were added to protect the system and I/O Expanders so that the number of I/O meets the needs of the system.

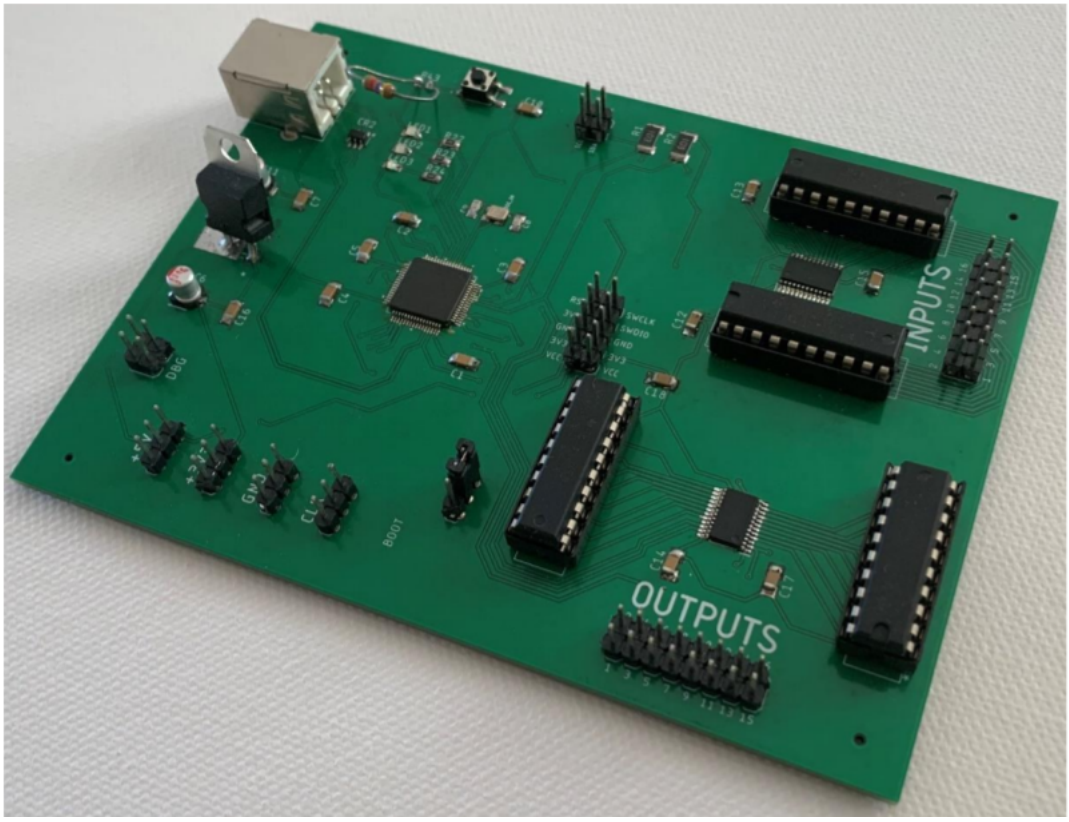

FIGURE 3.1: Implementation of the development board based on STM32F072RB microprocessor. [Source: Thesis by Emmanouil Lykos [6]]

The graphical interface presented in figure 3.2 provides the I/O indications of the system and allows the user to control them in order to verify the correct operation or to debug any errors. It was developed in C# programming language. For the communication with the physical interface on the computer, a VCP(virtual Com Port) has been implemented while on the physical interface the STM USB Stack for the peripheral USB functionality of the microcontroller. With the use of SerialPort Class of .NET Framework, which was selected for its compatibility with the operating system as well as for facilitating the access of the ports considering the use of drivers, the graphical interface has access to the platform as shown in figure 3.3. Finally, a communication protocol has been created between the GUI and the microcontroller and it's about an asynchronous bidirectional communication while a real-time control is achieved.

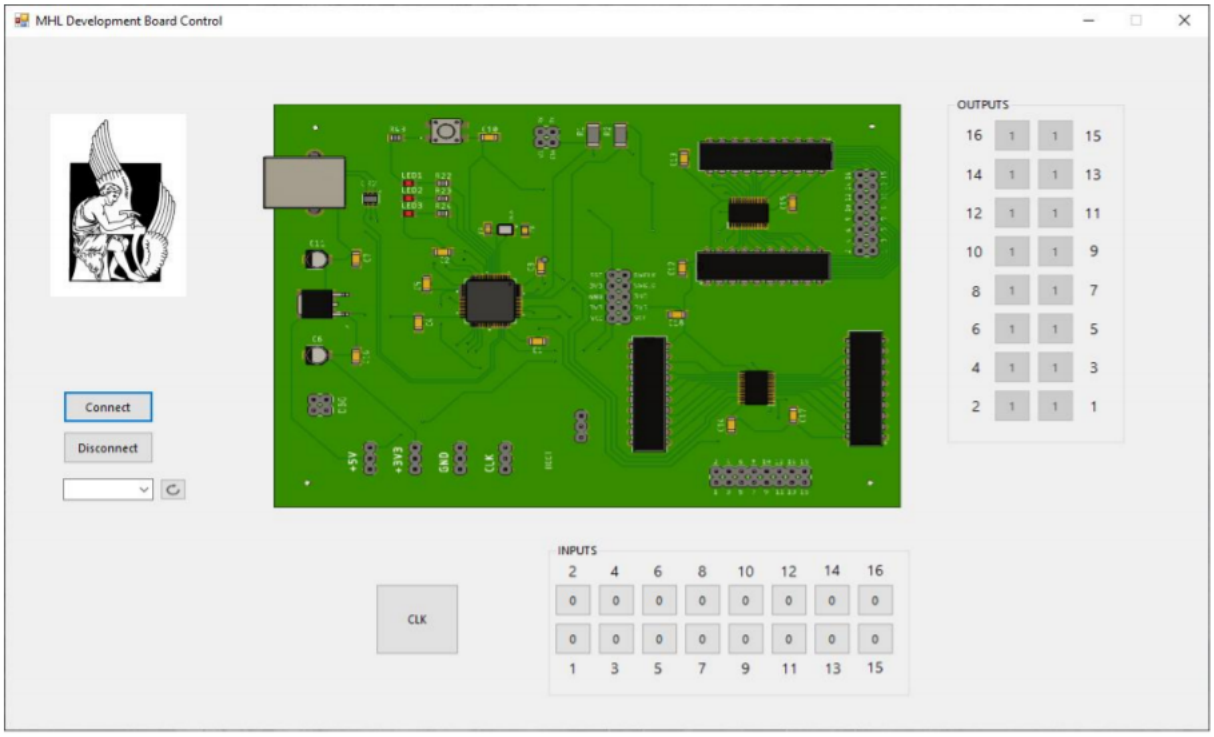

FIGURE 3.2: The graphical interface "MHL Development Board Control". [Source: Thesis by Emmanouil Lykos [6]]

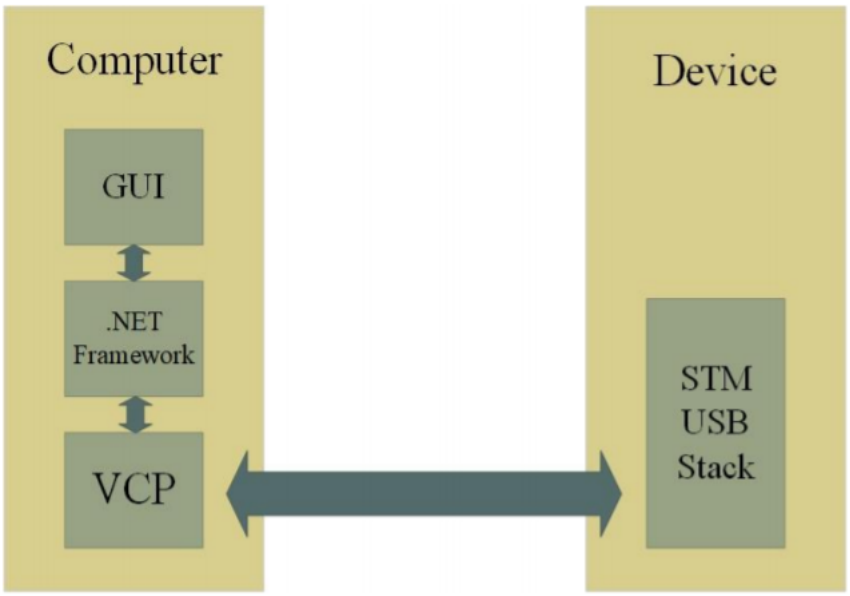

FIGURE 3.3: The flow of communication between the computer(GUI) and the "MHL Development Board" [Source: Thesis by Emmanouil Lykos [6]]

3.1.2 FPGA e-Lab

This paper on Remote Lab [12] presents a technique to remote Access a laboratory to design and test. It was created for educational purposes and the target is a FPGA.

Access to the laboratory is been achieved through the Microsoft XP Remote Desktop which gives you complete control of the laboratory PC. In the laboratory, there is a FPGA development board whose hardware is interfaced through a GUI, an acquisition hardware as well as RS-232 serial and USB ports. Figure 3.4 below shows the development board, the GUI and the functions it provides.

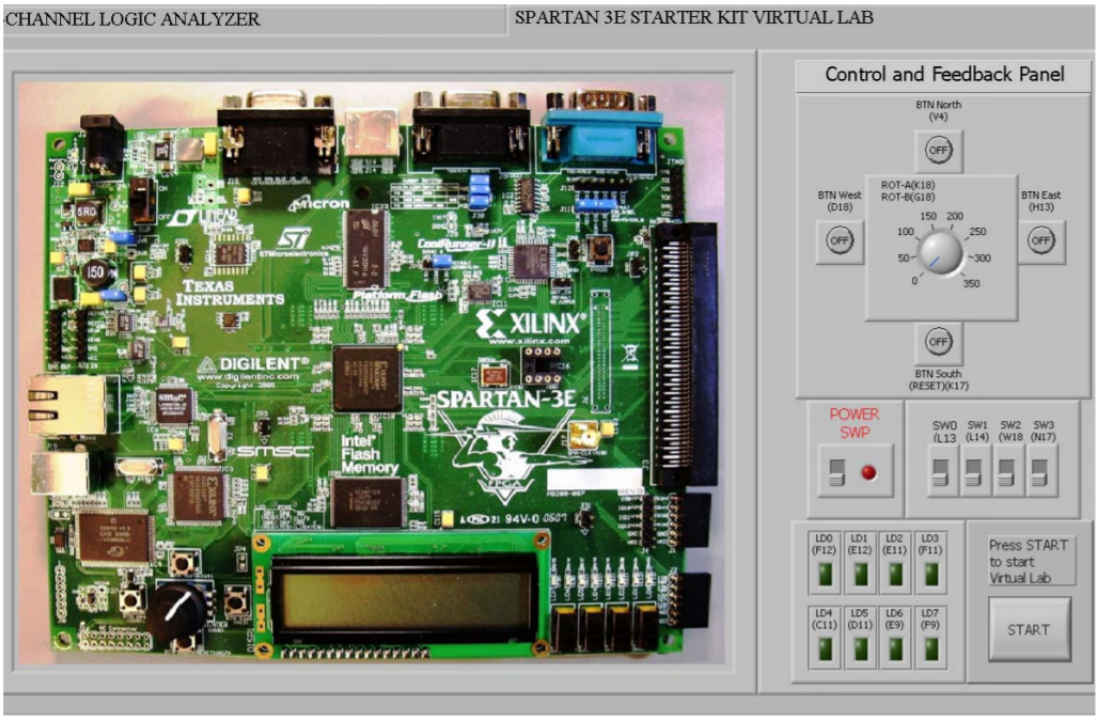

FIGURE 3.4: The GUI showing the FPGA development board and the I/O control and feedback panel. [Source: Reza Hashemian et al. [12]]

The FPGA development board that has been chosen is the Xilinx Spartan-3E Starter Kit. The Xilinx Spartan-3E Starter Kit is controlled using custom control hardware that is interfaced to the NI PCI-6025E/CB-100 data acquisition hardware and the LabView GUI. Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a system-design platform and development environment for a visual programming language from National Instruments while NI PCI-6025E/CB-100 data acquisition hardware is involved in aggregating signals that can be sent to the lab-PC. The student experiments consist of 7 stages of which the first 5 should be performed before entering the FPGA e-Lab environment and relate to design synthesis as well as all the preparation of the students. For stages 6 and 7, students will be able to access the FPGA e-Lab environment to program the FPGA development board, experimentally verify their designs and make modifications, resynthesize and retest if needed. Figure 3.5 shows the process of communication between the components of the system. Hewlett-Packard 5452 Oscilloscope and an Agilent 33250A which are connected directly to the Starter Kit constitute the Lab equipment.

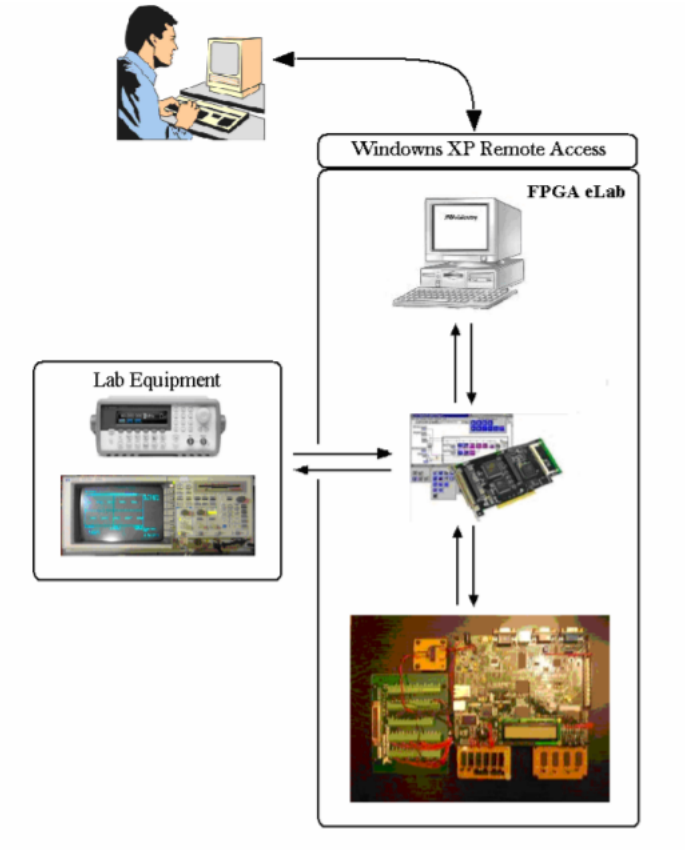

FIGURE 3.5: Schematic diagram of a remote interface system. [Source: Reza Hashemian et al. [12]]

For the verification and reliability of the system, a web cam has been added which gives a good view of the Spartan-3E Starter Kit as we see in figure 3.6.

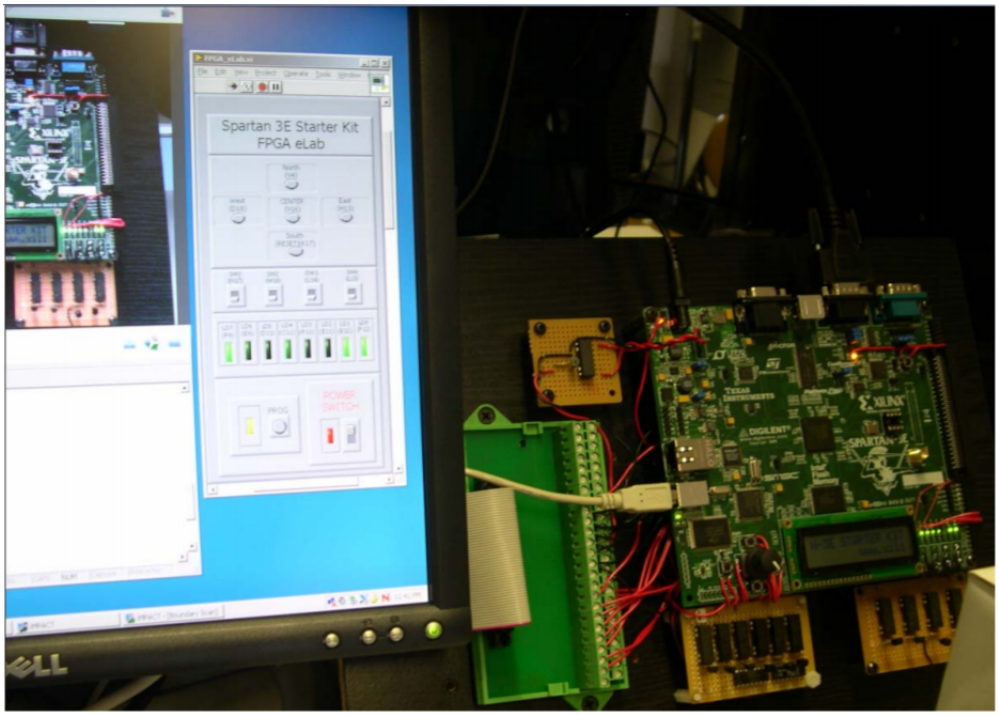

FIGURE 3.6: An e-Lab with video Capture of the FPGA board.
[Source: Reza Hashemian et al. [12]]

3.1.3 Wireless ATMEL AVR In-Circuit Serial Programmer based on Wi-Fi and ZigBee

The paper by Ahmed, Ahmed [13] presented at the International Computer Engineering Conference (ICENCO) presents the need for a modern programmer to facilitate factories as modernized systems require wireless communications. Wireless communications can be achieved with Wi-Fi and ZigBee, techniques which are used and analyzed below. The disadvantages of traditional programming methods are explained in the beginning, such as the need for the programmer to be close to a computer and a new method FOTA (Firmware Over-the-Air) is introduced.

The FOTA is one of the most modern embedded devices used by manufacturers in the automotive, consumer electronics, healthcare industries, and generally in the field of embedded systems. Its functionality enables manufacturers to fix bugs in software components of the existing system and to install updates remotely. In this way, devices are always up-to-date, even if new functions and features are only introduced after the purchase of a device. According to this, an ICSP programmer is presented,

based on an AVR microcontroller, which with the addition of a wifi device or a ZigbBee can program a microcontroller remotely from anywhere. ZigBee is based on the IEEE's 802.15.4 personal-area network standard. Zigbee is widely considered an alternative to Wi-Fi and Bluetooth for some applications including low-powered devices that don't require a lot of bandwidth and can provide low-power wireless programming. The use of both chips with the programmer will be presented and analyzed below. This wireless programmer functionality is writing, reading, and verifying flash and EEPROM memories as well as modifying fuse bits. Also, programmer focuses on low cost and portability which are achieved with the small size of the embedded system and the ability to be powered from an external power source or a battery. This system consists of three components, a PC / Phone, a Wi-Fi programmer and a target microcontroller as shown below in figure 3.7.

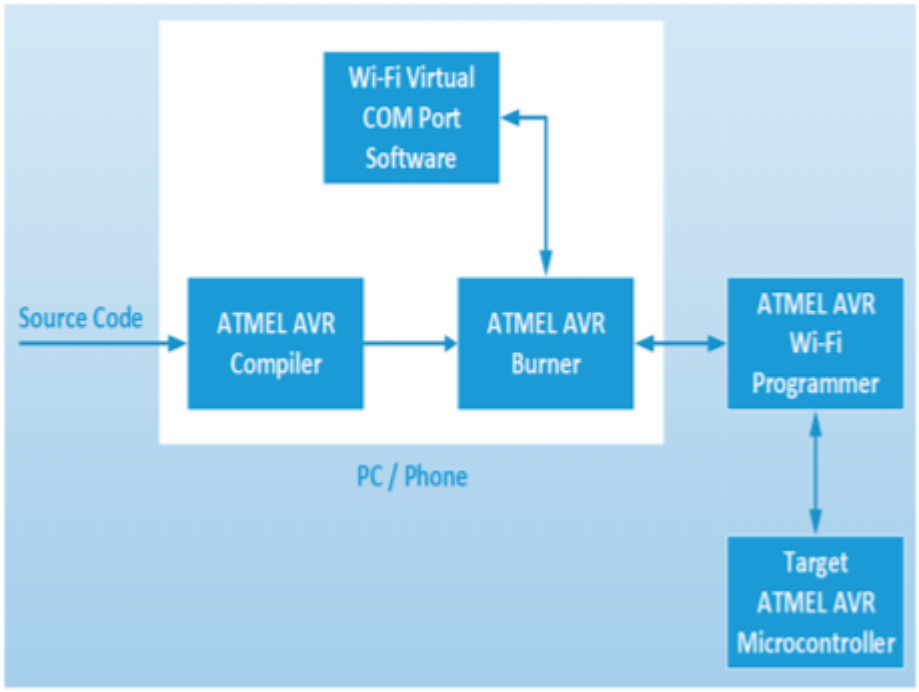

FIGURE 3.7: System Block Diagram of Wireless Programmer. [Source: Ahmed A. et al. [13]]

As is shown in the diagram the PC / Phone consists of three subsystems, the compiler software, the burner software, and a Wi-Fi virtual COM port converter software. The user uploads the hex file to an ATMEL AVR burner which converts the COM ports to virtual Wi-Fi communication through the Wi-Fi Virtual COM Port Converter software. The converter requires an IP address, a port number, and a communication protocol which is

TCP or UDP and defined by the user. The burner software remotely sends instructions to the programmer through the enabled communication channel and respectively the remote programmer will execute the sent commands to the target microcontroller and it will send feedback for all the activity statuses back to the burner software.

In terms of hardware figure 3.8, shows the PCB design schematic which is composed of the power unit, the microcontroller unit, the Wi-Fi / ZigBee module unit, and the ICSP unit.

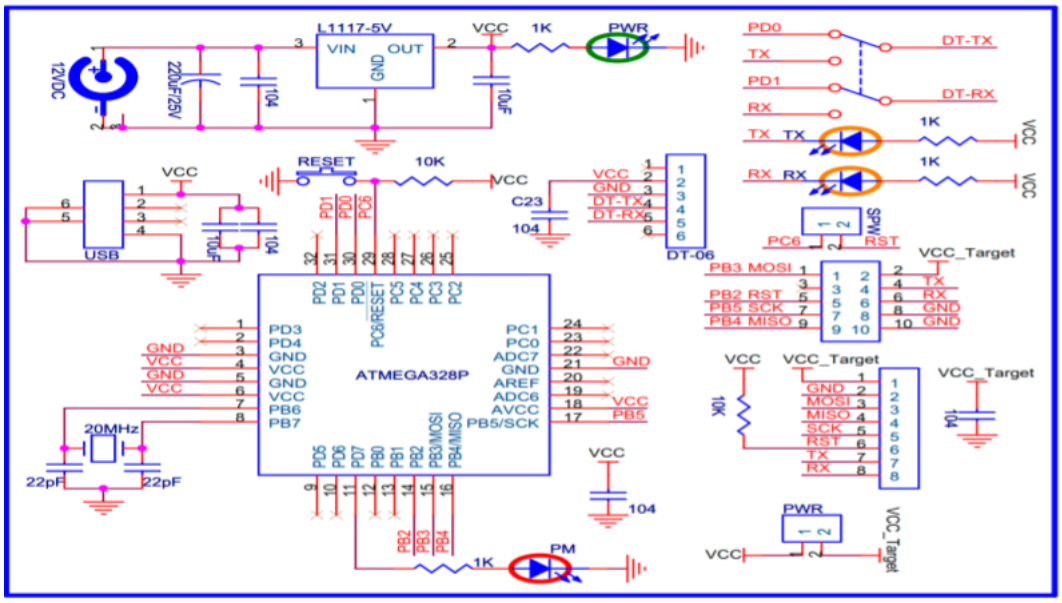

FIGURE 3.8: PCB Design Schematic of Wireless Programmer. [Source: Ahmed A. et al. [13]]

- The Power Unit provides the necessary power to the programmer via a USB port or a DC socket.
- The Microcontroller Unit is based on ATMEL AVR ATmega328P, which is loaded with the firmware that executes the ATMEL AVR programming AVRISP STK500 protocol.
- The Wi-Fi / ZigBee Module Unit function is to handle all the wireless communication between the PC and the remote programmer. It is based on the ESP8285 Wi-Fi chip of Espressif Systems family and the ZigBee (DIGI S2C 6.5mW ZigBee XBee) Wireless RF module respectively. This module is connected to both of the microcontroller serial communication pins for providing wireless programming and to the

Wi-Fi transparent serial (Tx, Rx) pins for wireless target board debugging.

- The ICSP Unit has an 8-pin header for wireless programming and debugging and a two-pin header is included to control the power source to the target board.

Finally, the last section of the paper provides two experimental works, one with the wifi technology and one with the ZigBee technology to test and verify those two designs. Wi-Fi Remote Programming: Figure 3.9 shows the embedded system with the use of the wifi module. This method uses the NB Virtual Comm Port Software as the Wi-Fi virtual COM port converter software and the BASCOM-AVR Burner Software as the burner software.

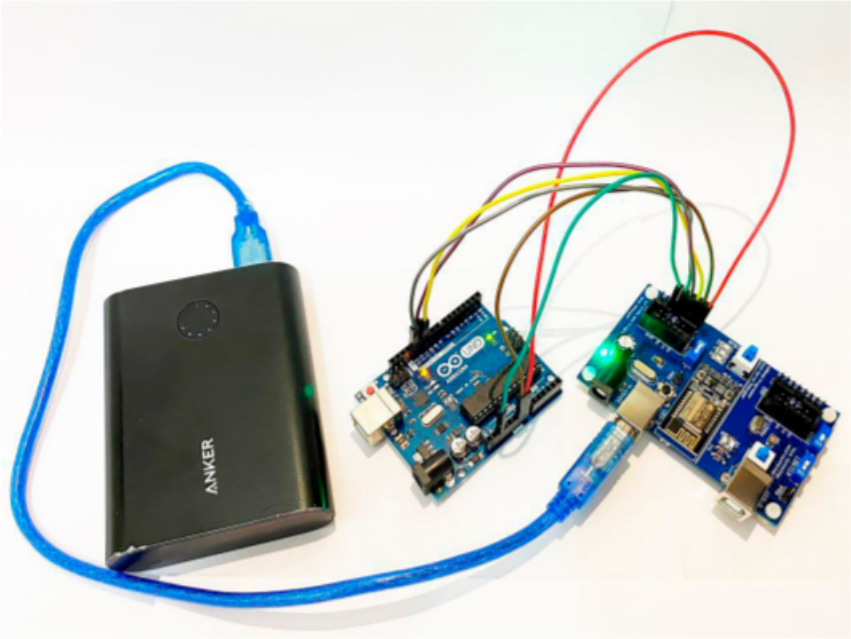

FIGURE 3.9: Wireless Programmer with Wifi. [Source: Ahmed A. et al. [13]]

ZigBee Remote Programming: The figure 3.10 shows the embedded system with the use of the ZiBee module. Target board loaded with a firmware image for blinking a led. The orange led on board proves that programming has been completed successfully.

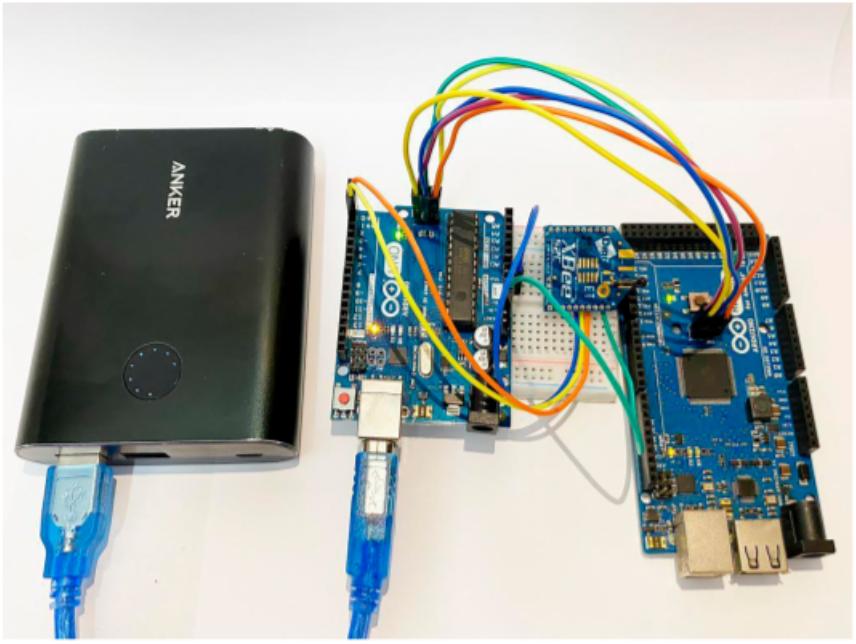

FIGURE 3.10: Wireless Programmer with ZigBee. [Source: Ahmed A. et al. [13]]

Both of these cases are functional without phasing any problems and the paper concludes with the recommendation of the utilization in industrial systems, in educational and academic projects.

3.1.4 Development and Application of Remote Laboratory for Embedded Systems Design

The paper by Parkhomenko and Anzhelika V. [14] presented at 12th International Conference on Remote Engineering and Virtual Instrumentation (REV) presents the RELDES(Remote Laboratory for Embedded Systems Design), an approach to remote access a laboratory based on using ready platforms. The proposed approach intends to improve the current technology in the field of embedded systems design with the possibility of accelerating the stages of integration and hardware-software testing.

Regarding the implementation, a stand with experiments is connected to the laboratory computer, via the serial interface. The computer with Linux Debian OS acts as the server of the remote laboratory. The server provides access to programming and testing the components and video stream of the laboratory experiments. Video stream is been achieved with the implementation of "ffmpeg". It is a cross-platform solution to record, convert and stream audio and video. The server receives and processes the web client requests and performs the corresponding functions depending on the data received:

- Compiling the received initial code (Arduino, with the use of the console utility Ino).
- Returning the compilation results (using the HTTP protocol, as well as the sends request for compilation).
- Uploading binaries to the microcontroller (provided that the controller is free).
- Client queuing (if an experiment is occupied).

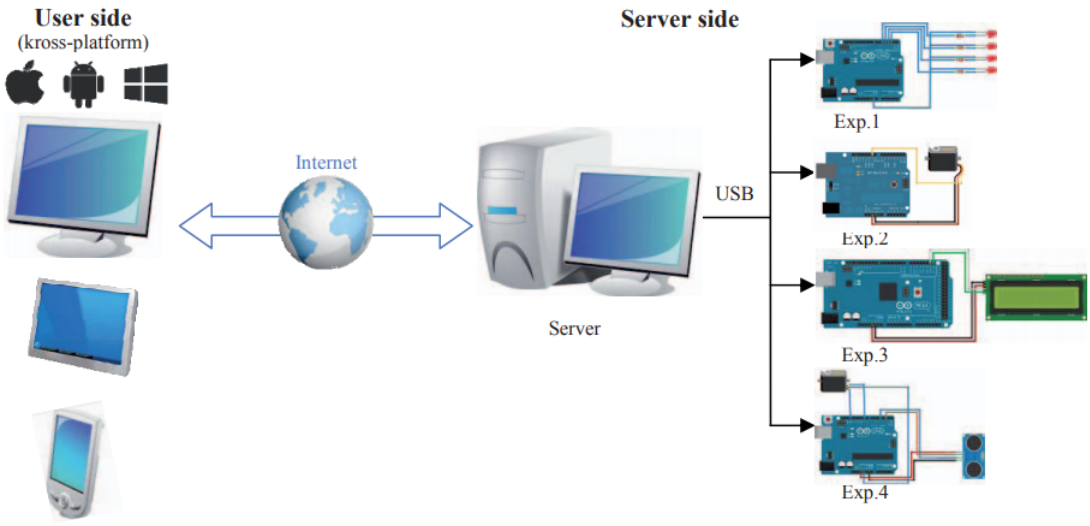

FIGURE 3.11: An overview of RELDES. [Source: Parkhomenko Anzhelika et al. [14]]

3.1.5 Remote Programming and Reconfiguration System for Embedded Devices

The work [9] presents a concept of a system that can be used as a useful remote management and control tool for a built-in device. It can be used on wireless sensors and IoT nodes based on a microcontroller or a programmable field gate chip (FPGA).

The system, which is shown in figure 3.12, consists of two separate main parts: The hardware part called Remote Programming Device (RPD) and the management part which is a user application for communicating with one or more RPDs. The RPD is capable of remotely reprogramming internal memories of microcontrollers with provided binary firmware files and optionally reconfiguring FPGA integrated circuits. Communication between the two parts of the system is done using the LWM2M protocol. For the management part, the Eclipse Leshan implementation of the LWM2M server is chosen. It provides a Web-based user interface (UI) that allows interaction with connected RPDs. The communication of the RPD with the target device is done according to the protocol Joint Test Action Group IEEE 1149.1 (JTAG) and specifically with Serial Wire Debug (SWD). The implementation of the hardware is based on The Nucleo boards equipped with STM32F429ZI MCU with ARM Cortex-M4 microprocessor core. This embedded system provides an ethernet port from which to access the Internet.

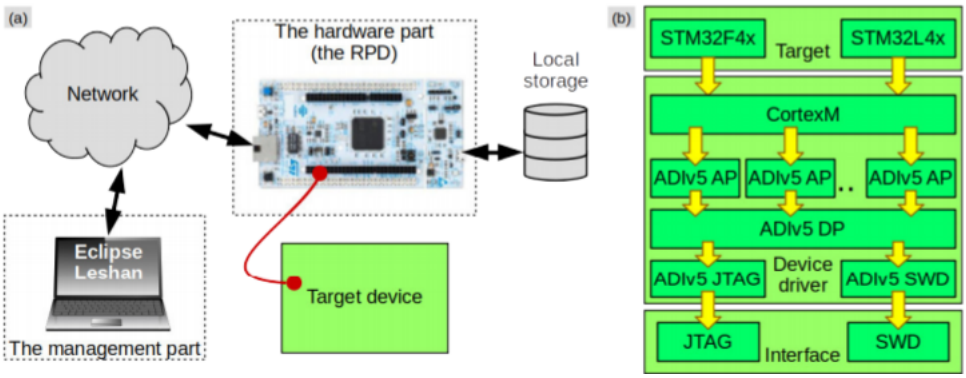

FIGURE 3.12: RPD architecture (a) and RPD abstraction layers diagram (b). [Source: Michalec et al. [9]]

The whole communication between the user and the built-in device includes two parts: the management interface which is carried out through the LWM2M server and the file download which is carried out through Hypertext Transfer Protocol(HTTP) server. The following figure 3.13 shows in more detail the communication process.

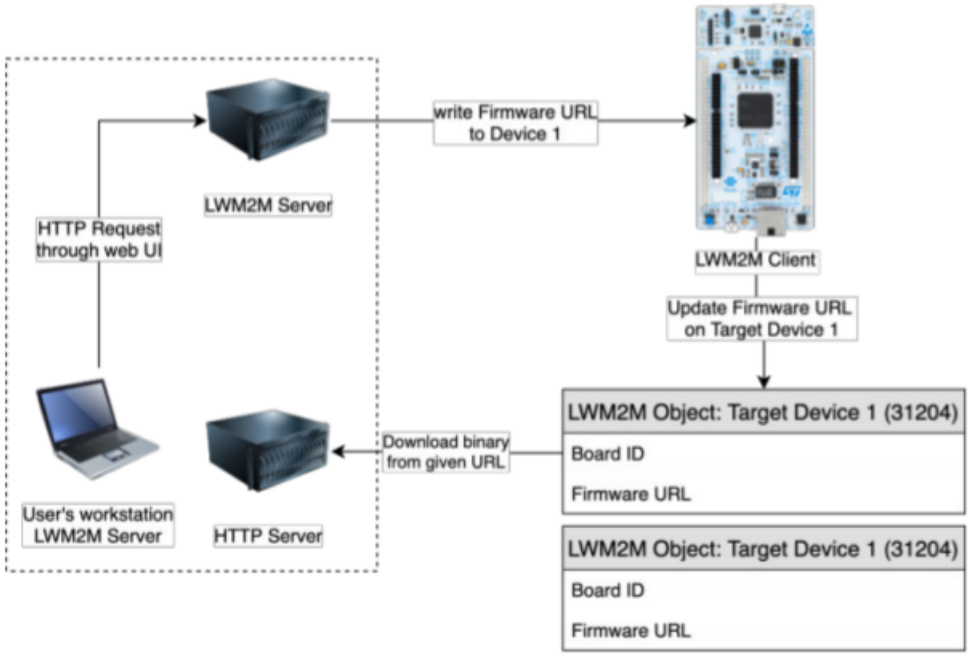

FIGURE 3.13: Communication flow of RPD system. [Source: Michalec et al. [9]]

3.2 Thesis Approach

The approach of this thesis was based on the idea of the Internet of Things (IoT), which is considered an interesting technology with new features and a field of the future in embedded systems that continues to grow rapidly. The main objective of this thesis was to provide a solution for remote access to laboratory equipment and to take advantage of the rapid development of technology in order to offer a useful tool in education.

To achieve this goal, the thesis focused on the implementation of a microcontroller that runs the essential functions for the development system, and a separate Wifi module that hosts the web server. Both the microcontroller and the wifi module were selected based on the criteria presented in the previous chapter, such as hardware and software architecture, processing power, memory, cost, hardware interface, security, temperature tolerance, and power efficiency.

The choice of using a web-based application instead of a GUI application was more convenient and met the purposes of the implementation. The web-based application allows for full functionality from the client side without the use of any external application, except for a compiler that gives the binary file that the client will upload to program the target microcontroller.

Furthermore, the approach of the thesis was to fully access the target microcontroller of the laboratory, its programming, and control. The method proposed in this thesis not only satisfies the requirements but also offers several advantages over traditional methods. For example, it allows for real-time feedback, easy scalability, and flexibility in terms of access and control. Additionally, it allows for the remote monitoring and control of laboratory equipment, which is especially useful in situations such as the current pandemic, where traditional in-person access to laboratory equipment may be limited.

Overall, the approach of this thesis was to provide a solution for remote access to laboratory equipment through the implementation of an IoT-based system that utilizes a microcontroller and a wifi module. The system is designed to be user-friendly and easy to use, while also providing a high level of functionality and control over the laboratory equipment.

Chapter 4

Design and Implementation of the Platform

In order to complete this thesis, a platform was developed that fulfills the requirements of a Remote Laboratory. Although the platform is unique in its implementation, it is based on the main concepts of previous implementations. The platform was created to be highly scalable and can be used in a variety of different laboratory environments. The design and implementation of the platform were focused on achieving the goal of providing remote access to the laboratory and programming and control of the target-microcontroller in a cost-effective and efficient manner. The platform comprises of two main aspects, one related to achieving remote access to the laboratory and the other related to programming and controlling the target microcontroller. Further details will be provided in the following sections as the methods and main components used for implementation are described.

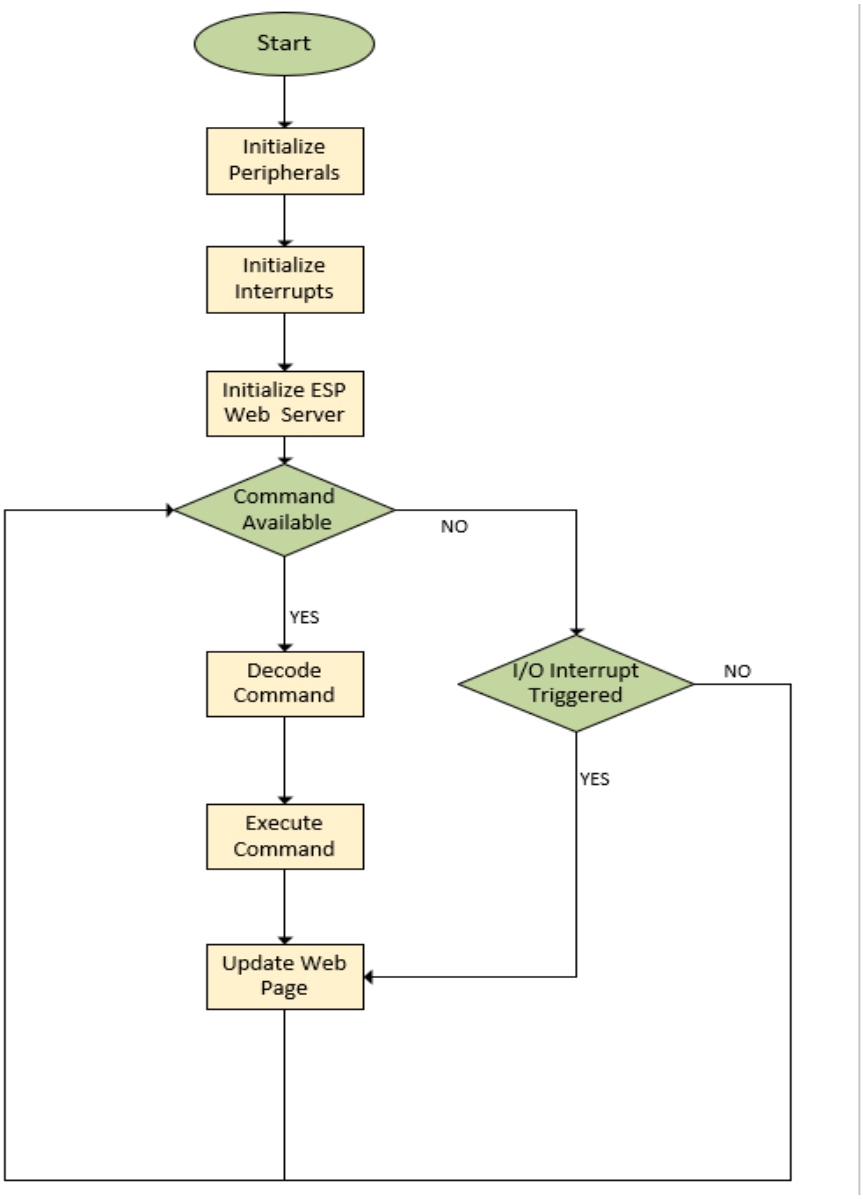

FIGURE 4.1: The flow diagram of the main process of the microcontroller of this thesis.

Above, the flow diagram shows the sequence of processes executed by the system during initialization and at each iteration. First, the system initializes the peripheral components that are needed. In the beginning, the system initializes the peripheral components that are needed. Then it initializes the interrupts mainly related to the UART that enables the communication with the wifi module and the I2C for communication with the I/O Expanders. Subsequently, the system sets the priorities and initializes the interrupt handlers. Then the ESP12E web server is initialized as described in section 4.2 and this is when the main iteration loop of the program starts.The

system checks if it has received a command from the remote user. In case it has received a command, it first decodes the command and then executes it, if the conditions are met. Before checking for the next command it updates the Esp Web Page with a message about the progress of the command. At any time if there is a change in any output the interrupt handler is activated and the Esp Web Page is immediately updated with the new output values.

The schematic diagram presented in Figure 4.2 provides a visual representation of the communication and interaction between the modules and components within the platform. It serves as a key resource for understanding the implementation details and the flow of information in the system.

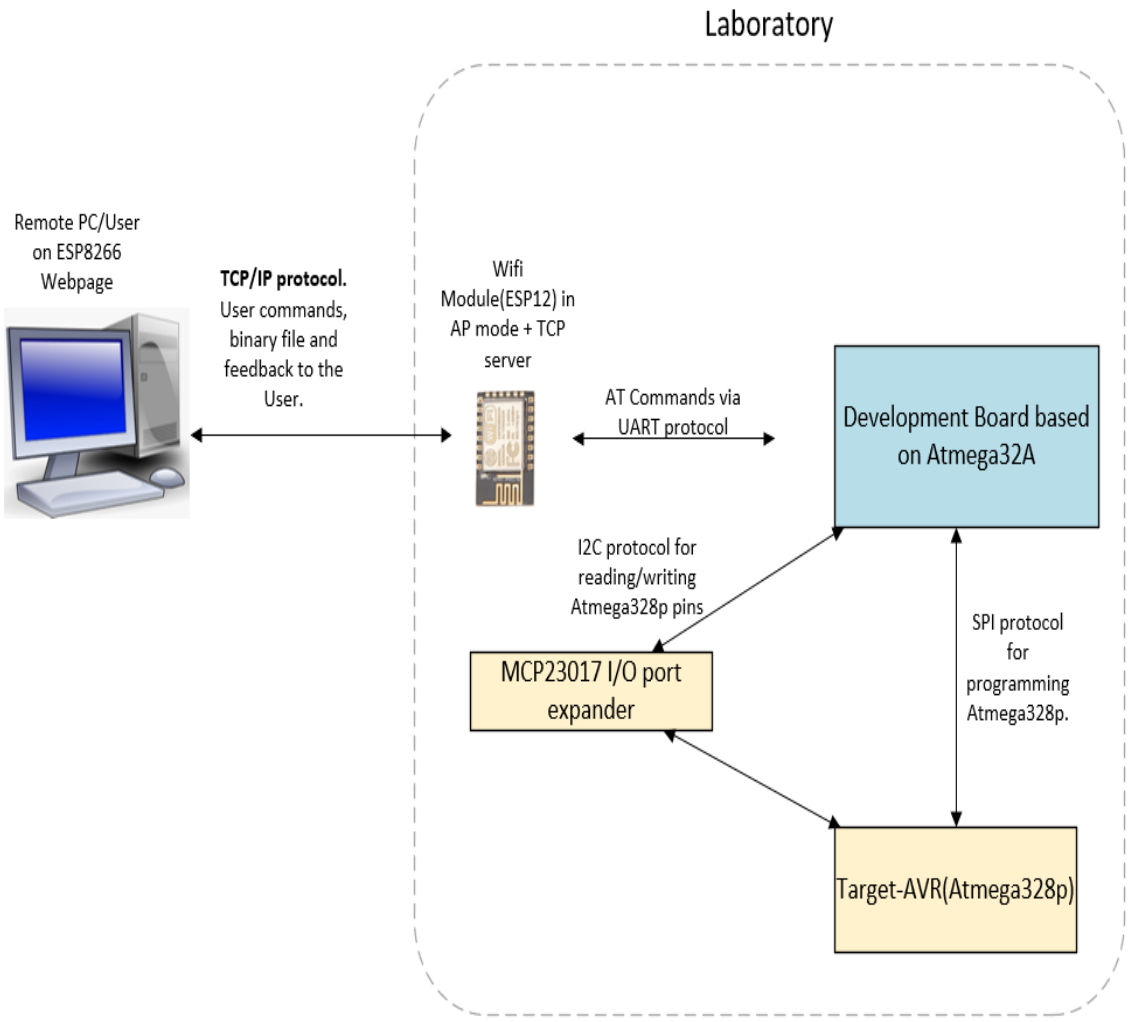

FIGURE 4.2: Schematic diagram of the Platform.

4.1 Component Selection and Integration in Implementation

Prior to the individual module presentations detailing their implementation, it is crucial to introduce the block diagram, which serves as an overview of the system's design and functionality. This block diagram provides a visual representation of the architectural structure and interconnections of the newly developed platform within this thesis.

The schematic diagram presented in Figure 4.2 provides a visual representation of the communication and interaction between the modules and components within the platform. It serves as a key resource for understanding the implementation details and the flow of information in the system.

4.1.1 Microcontroller

The platform is based on the ATmega32A-PU microcontroller, which serves as the main means of control and programming for my system. I chose this microcontroller for its versatility and wide range of features that perfectly met the requirements of my project. Furthermore, I selected the ATmega32A-PU due to its excellent power efficiency. Its low power consumption allowed for longer operation on battery power, extending the system's runtime and enhancing overall efficiency. In addition, my decision to use the ATmega32A-PU was influenced by its large community base and extensive documentation. This ensured that I had access to a wealth of resources, code examples, and support. The strong community backing provided valuable knowledge and assistance during the development process.

Overall, the ATmega32A-PU's combination of versatile features, power efficiency, and strong community support made it the ideal choice for my system. According to the above, I arrived at the conclusion that the ATmega32A-PU possessed the necessary capability to consistently deliver reliable performance and effectively streamline the development process.

The ATmega32A-PU microcontroller is part of Atmel's AVR series, which is known for its powerful processing capabilities and efficient use of resources. This microcontroller is equipped with a variety of communication interfaces, such as UART and I2C, that enable seamless communication with other components of the system. Additionally, it has a rich set of peripheral

systems that support the various functionalities of the system, such as timers, interrupts, and other digital and analog inputs/outputs.

Another important factor in the selection of the ATmega32A-PU microcontroller was my familiarity with the AVR series, which made it easier for me to program and debug the system. Overall, the ATmega32A-PU microcontroller serves as the backbone of the system, providing the necessary resources and capabilities to enable the system to function as intended.

The ATmega32A-PU microcontroller has the following features:

- High-performance, Low-power AVR® 8-bit Microcontroller
- 40-pin PDIP
- 131 Powerful Instructions – Most Single-clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Up to 16 MIPS Throughput at 16 MHz
- On-chip 2-cycle Multiplier
- 32 Programmable I/O Lines
- 32K Bytes of In-System Self-programmable Flash program memory
- 1024 Bytes EEPROM
- 2K Byte Internal SRAM
- JTAG (IEEE std. 1149.1 Compliant) Interface
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- On-chip Analog Comparator
- Four PWM Channels
- 8-channel, 10-bit ADC
- Two 8-bit Timer/Counters
- One 16-bit Timer/Counter
- Real Time Counter with Separate Oscillator
- Byte-oriented Two-wire Serial Interface
- Internal Calibrated RC Oscillator

- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- Speed Grades: 0 - 16 MHz

The platform that hosted the microcontroller and enabled the development, testing, and verification of the system is the STK500. This development board provides us with all the necessary tools we need to program and test our system. The STK500 is equipped with a variety of peripherals such as LEDs, buttons, and a serial port that we can use to interface with our microcontroller.

As a development environment, Microchip Studio was chosen for developing and debugging the ATmega32A-PU. This is one of the most well-known and reliable development environments for microcontrollers of the AVR family. Microchip Studio offers a wide range of features such as a powerful code editor, a project manager, and a built-in debugger that allows us to easily write, test, and debug our code.

To ensure optimal performance, a 16MHz external crystal was used as the system frequency for the microcontroller. This high-frequency crystal allows the microcontroller to execute instructions at a faster rate, which in turn improves the overall performance of the system. By using an external crystal, we were able to achieve accurate and stable timing, which is essential for many of the system's functions.

Overall, the STK500 development board, Microchip Studio development environment, and the 16MHz external crystal were all essential components that enabled us to effectively program, test and verify our system using the ATmega32A-PU microcontroller. The combination of these tools and technologies allowed us to achieve our system's goals and functionality.

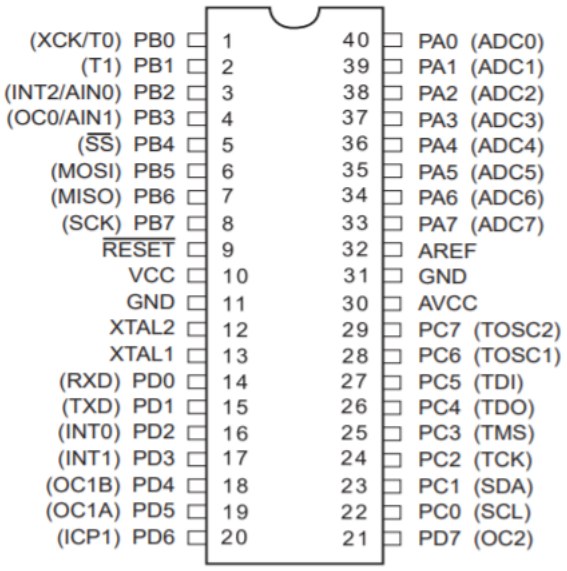

FIGURE 4.3: Schematic diagram of ATmega32A-PU pins.

4.1.2 Wifi Module

The whole process of communication with the laboratory was based on the ESP12E module [15]. ESP-12E is a small-sized Wi-Fi module that is commonly used to create a wireless network connection for any microcontroller. This module is a versatile and powerful solution for adding Wi-Fi capabilities to systems or for functioning as an independent application. It was selected for this project because it is one of the most widely used and reliable wifi modules on the market, and it fully satisfies the system requirements. Additionally, it is a cost-effective solution for developing IoT applications.

In the implementation, only the wifi capabilities of the ESP-12E module are utilized without taking advantage of its processing power, as that is provided by the separate microcontroller that is used in the system. The core processor of the ESP-12E module is the esp8266, which is produced by Espressif Systems and comes with a full TCP/IP stack [16] and microcontroller capability. This allows the ESP-12E to serve as a Wi-Fi adapter to any microcontroller that supports simple connectivity through a SPI or UART interface.

The ESP-12E module operates at 3.3v DC and provides 22 pins as shown in figure 4.4 which offer multiple types of communication options. These include 11 I/O pins and pins for UART, SPI, I2C, I2S, PWM, ADC, IR

Interface, and Power and Control Pins. These various communication options allow for flexibility and customization in the system's design and functionality. The ESP-12E module is a reliable and efficient solution for adding Wi-Fi capabilities to the system and facilitating communication with the laboratory.l Pins.

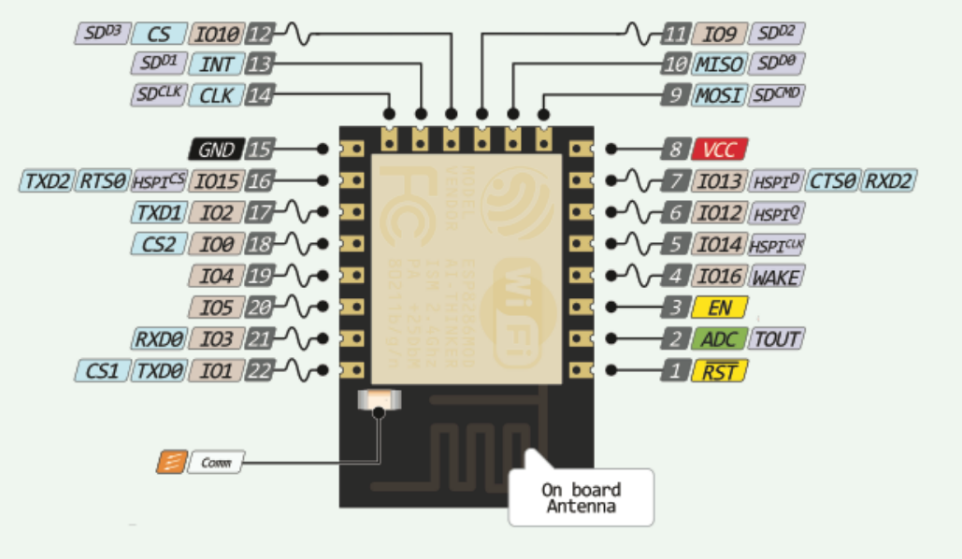

FIGURE 4.4: Schematic diagram of ESP-12E Pins.

Provided that the wifi module in this implementation is used only as a wifi adapter only the pins related to UART communication are used while the other pins are available either for reprogramming of the wifi module or for debugging or for the future possibility of extending the implementation. Other ESP-12E wifi modules features:

- 802.11 b/g/n support
- 802.11 n support (2.4 GHz), up to 72.2 Mbps
- Defragmentation
- 2 x virtual Wi-Fi interface
- Automatic beacon monitoring (hardware TSF)
- Support Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode
- Certification: Wi-Fi Alliance
- 2.4 GHz 2.5 GHz (2400 MHz 2483.5 MHz) Frequency Range

- Antenna: PCB Trace, External, IPEX Connector, Ceramic Chip
- Tensilica L106 32-bit processor
- Operating Voltage 2.5 V 3.6 V
- Operating Current: Average value: 80 mA
- Wi-Fi Mode Station/SoftAP/SoftAP+Station
- Security WPA/WPA2
- Network Protocols: IPv4, TCP/UDP/HTTP

To program the ESP-12E, there are several options available. One popular method is using the Flash Download Tool provided by Espressif, which allows for flashing firmware onto the module. This can be done using Lua scripting language or with the Arduino IDE using libraries specifically designed for the ESP-12E. Another option is using the AT command firmware developed by Espressif, which provides a set of commands that can be used to control and operate the module. This implementation uses the AT command firmware, which can be easily flashed onto the ESP-12E using the Flash Download Tool.

Once the firmware is in place, the AT commands of the ESP-12E can be utilized to perform a variety of functions, such as restarting the module, connecting to a WiFi network, changing the mode of operation, and more. These commands are crucial for the proper functioning of the module and are analyzed in detail during the development of this implementation. It's also important to note that the ESP-12E module has a wide range of configurable options and settings, which can be accessed and modified through the AT commands.

In addition to programming and configuring the ESP-12E, proper wiring is also crucial for its proper functioning. The wiring diagram for the ESP-12E is shown in the figure below. The pins related to UART communication are used for programming and debugging, while the other pins are available for reprogramming of the wifi module or for the future possibility of extending the implementation. It's important to ensure that the connections are secure and properly made to ensure proper communication between the module and other devices

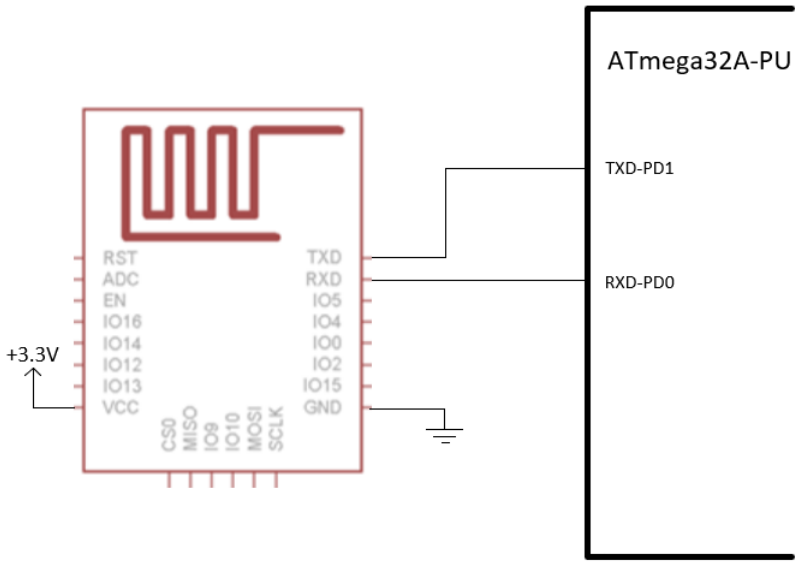

FIGURE 4.5: Connection between ESP-12E and ATmega32A-PU.

4.1.3 I2C EXPANDER 16BIT I/O - MCP23017

The development platform for this project includes an integrated I/O expander, which is essential for expanding the available resources and increasing the number of inputs and outputs for controlling and programming the microcontroller. This allows for more flexibility and the potential for future expansion of the functionality.

After careful consideration, the MCP23017 [17] was chosen for this implementation. This device is considered sufficient as it offers the necessary features required for the project. The MCP23017 is a 16-bit general purpose parallel I/O extension that can be used for I2C or SPI bus applications. In this project, the I2C interface is utilized to communicate with the Atmega32A microcontroller. Custom libraries were developed for the I2C protocol, facilitating efficient and reliable communication. This approach enables the release of Atmega32A pins for other functions. Additionally, we leveraged the interrupt mode feature of the MCP23017 I/O expander to enable real-time reading of the outputs from the target microcontroller. This capability allowed us to promptly capture and process the output data, ensuring accurate and up-to-date information for our system.

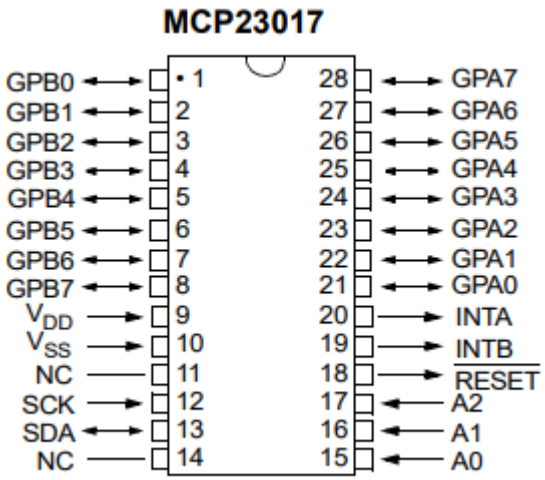

FIGURE 4.6: The MCP23017 integrated circuit.

The connection between the I/O expander and the microcontroller is relatively simple and can be easily achieved through the use of simple wiring. The MCP23017 is connected to the Atmega32A and the connections can be seen in the figure provided. This allows for easy communication between the two devices and enables the MCP23017 to provide the necessary resources for the project. With the use of this I/O expander, the development platform is able to expand its capabilities, providing greater flexibility and the potential for future expansion.

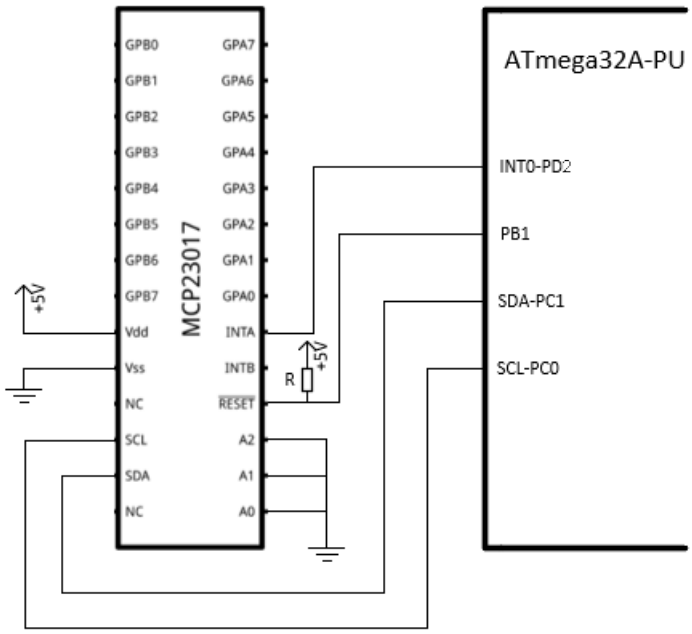

FIGURE 4.7: Connection between I/O Expander and ATmega32A-PU.

4.1.4 Target-Microcontroller

The ATmega328p microcontroller offers a wide range of features and capabilities that make it an ideal choice for students and hobbyists. It is a powerful and versatile microcontroller that can be used to control a variety of devices and systems. The ATmega328p is part of the megaAVR family of microcontrollers, which are known for their high performance and low power consumption.

One of the key features of the ATmega328p is its wide range of input and output pins. This allows for easy connection to a variety of sensors and actuators, allowing students to experiment and test their code in a variety of different applications. Additionally, the ATmega328p offers a variety of different communication interfaces, including UART, I2C, and SPI, which allow for easy connection to other devices and systems.

When it comes to programming the ATmega328p, students should refer to the official datasheet provided by Atmel. This document provides detailed information about the microcontroller's features and capabilities, as well as instructions on how to write and program code for the device. The

student can upload the code to the ESP-Webpage, which will be used to control the microcontroller.

- High-performance, Low-power AVR® 8-bit Microcontroller
- 28-pin PDIP
- 131 Powerful Instructions – Most Single-clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Up to 20 MIPS Throughput at 20 MHz
- On-chip 2-cycle Multiplier
- 23 Programmable I/O Lines
- 32K Bytes of In-System Self-Programmable Flash program memory
- 1K Bytes EEPROM
- 2K Bytes Internal SRAM
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- On-chip Analog Comparator
- Six PWM Channels
- 6-channel 10-bit ADC
- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Real Time Counter with Separate Oscillator
- Byte-oriented Two-wire Serial Interface
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- Speed Grades: 0 - 20 MHz on 4.5 - 5.5V

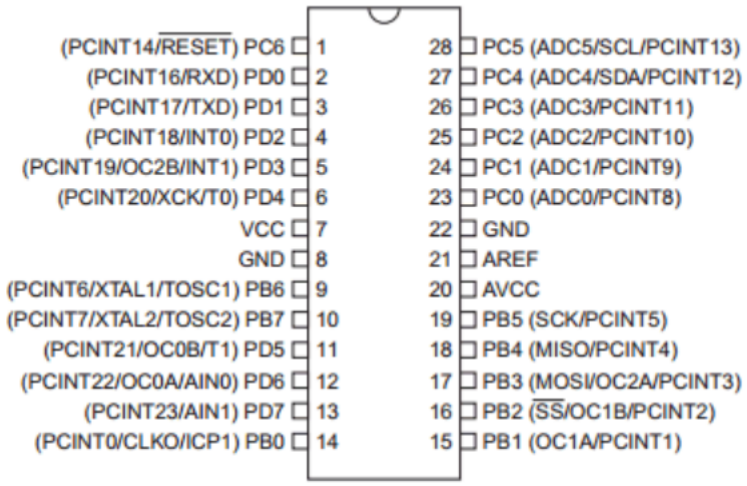

FIGURE 4.8: Schematic diagram of ATmega328p pins.

The ATmega328p is connected to the ATmega32APU, which is used for programming the microcontroller. This connection is only necessary for programming and is not used for controlling the ATmega328p. Instead, the ATmega328p is controlled using the I/O Expander described above. This allows for more flexibility and versatility in terms of the circuits that can be added to the microcontroller for each experiment.

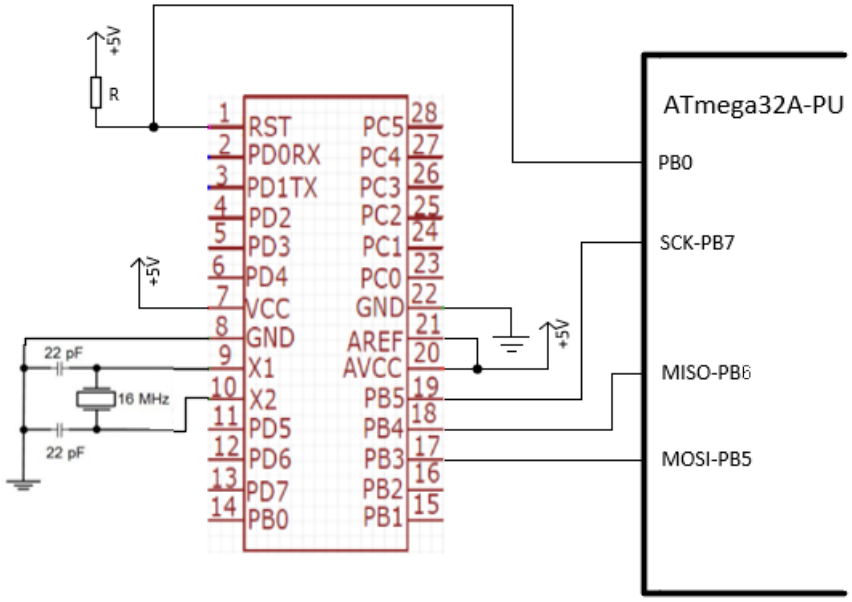

FIGURE 4.9: Connection between ATmega328p and ATmega32A-PU.

4.2 Remote Access to the Laboratory of the University

In order to utilize the full capabilities of the ESP12 wifi module, it was programmed with firmware based on AT commands, provided by Espressif. According to this, I developed a set of libraries. These libraries were specifically designed for use with the host microcontroller, ATmega32A-PU, and allow for the sending and receiving of AT commands between the two devices. The ATmega32A-PU sends the AT commands to the ESP-12E, which then executes the corresponding command and sends back a response indicating the success or failure of the request. These libraries contain a set of AT commands that are specifically used for setting up a server based on the TCP/IP protocol on the ESP-12E. The Transmission Control Protocol/Internet Protocol (TCP/IP) is a widely used set of communication protocols that enable devices to communicate over a network such as the internet. The use of these AT commands in conjunction with the TCP/IP protocol allow for the creation of a server on the ESP-12E, providing the ability to remotely access and control the device. The list of these AT commands is described in detail below.

1. **Command: "AT", Response:"OK".** Just a command to verify that

the communication between ATmega32A-PU AND ESP-12E is working properly.

2. **Command: "AT+RST", Response:"OK".** Reset ESP-12E to properly accept new commands.

3. **Command: "AT+CWMODE=3", Response:"OK".** Set the Wi-Fi mode to Station+SoftAP. In Station mode, ESP-12E acts as a device and connects to an existing Access point. In Access Point(SoftAP) mode ESP-12E acts as AP and other devices can connect to it. In a TCP server both modes are required to be activated.

4. **Command: "AT+CWJAP_DEF=ssid,password", Response:"WIFI CONNECTED","WIFI GOT IP","OK".** Connect ESP-12E station to a targeted Access Point.

5. **Command: "AT+CIPMODE=0", Response:"OK".** Sets the transmission mode. By choosing 0, normal transmission mode is enabled.

6. **Command: "AT+CIPMUX=1", Response:"OK".** Set the connection type. By choosing 1, multiple connections mode is enabled.

7. **Command: "AT+CIPSERVER=1,80", Response:"OK".** Creates a TCP server on port 80.

8. **Command: "AT+CIPSTO=0", Response:"OK".** , The connection will never time out.

When the commands mentioned above have been completed and properly executed, a TCP server has been created and is ready to accept requests from clients. Then the server waits for new requests from clients and follows the process of serving the requests of the clients that is analyzed below. Atmega32A has libraries to handle and serve all the requests it can accept. Once the server has been set up, in order to connect to the ESP12E web server the user must enter the IP address of the ESP12E in a web browser. The server receives a request from the user's id and ESP12E sends the request to Atmega32A respectively. Atmega32A then sends a page to the user, via ESP12E, which requires a username and password as shown in figure 4.10. In case of correct entry by the user a second page which is the main one with which the user interacts throughout the duration of his experimentation is sent to the user which is shown below in figure 4.11 It should be noted that the pages mentioned above are written in JavaScript[18], HTML[19], and

CSS[20]. In summary, HTML is the standard markup language for documents designed to be displayed in a web browser, CSS is used for describing and styling the presentation of the documents in a web browser and JavaScript makes web pages more interactive and dynamic. The pages are stored in the Atmega32A and sent to the user's browser when required. When the above steps are completed, ESP12E waits to receive and serve the user's requests which depend on the selection of the available options on the page which are analyzed below.

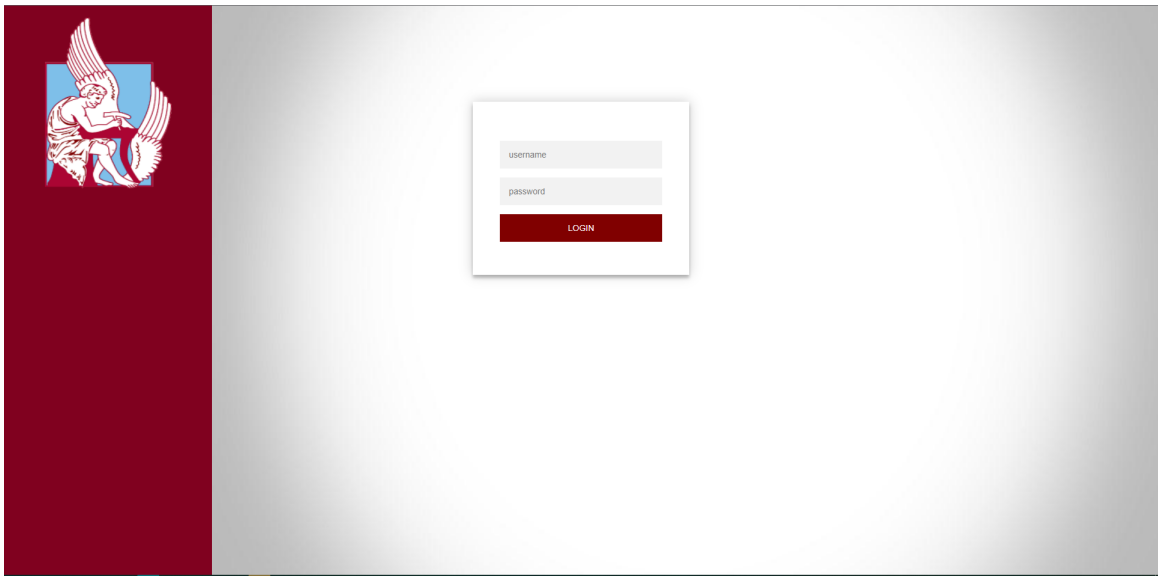

FIGURE 4.10: Login Web Page Developed in this Thesis.

Below is the interface of the page with which the user interacts and controls the target microcontroller, atmega328p. The user has 8 outputs available, all pins of Port D (PD0-PD7) as well as 5 inputs, specifically the first 5 pins of Port C (PC0- PC5). A basic prerequisite for using the application is for the user to initialize the ports as indicated above in his own source code.

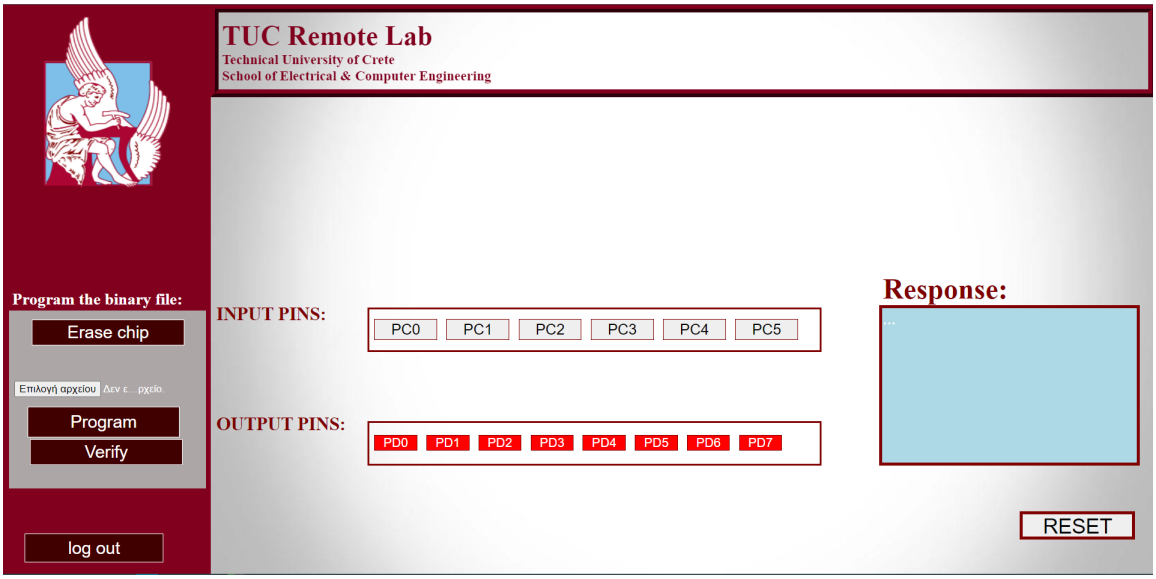

FIGURE 4.11: Main Web Page Developed in this Thesis.

The web page has the following options for the user:

- Erase chip: The user erases the target microcontroller, atmega328p.
- Program: The user with this option, after selecting the binary file first, programs the target microcontroller, atmega328p.
- Verify: The user verifies the correct execution of the programming.
- log out: The user logs out of the page when the experiment is over.
- RESET: The user resets the target microcontroller, atmega328p.

In addition, during the execution of the program:

- The user can see the output values of ATmega328p (PD0-PD7). Depending on their color, their value is indicated. Green for logical 1 and red for logical 0.
- The user can set the value of the ATmega328p inputs by pressing the corresponding button.
- In the Response panel, the user is informed with a relevant message for each function that performs.

From the above pages, the most interesting parts are the 2 interactive elements of the page. These are the Response Panel and the color of the outputs that indicate their value. You can refer to the JavaScript code in

the appendix to observe how the functions below achieve the synchronous update of the aforementioned elements Appendix B.

To ensure the smooth and accurate operation of the process, various AT commands were executed that were not previously discussed during the preparation of the server and the student's experimentation, but are still important. These commands play a vital role in providing information about the correct operation of the connection and various other factors that are necessary for a stable and reliable environment. These commands are used to validate the operation and ensure that all necessary parameters have been configured correctly.

4.3 Programming and Control of the Laboratory Microcontroller

In order for the target-microcontroller to be programmed, the ATmega32APU serves as a programmer based on the AVR910-ISP method [21]. In-System Programming (ISP) allows programming and reprogramming of any AVR microcontroller. This method is particularly useful for the ATmega328p, which is the target microcontroller used in this project. The ATmega328p is equipped with paged flash memory, which means that data is first loaded into a page buffer and then written to flash memory when the buffer is full. This allows for efficient use of memory and reduces the risk of data loss during programming.

The choice of using ISP and the AVR910-ISP method for programming the ATmega328p was made for several reasons. One of the main reasons is that ISP allows for in-system programming, meaning that the microcontroller can be programmed and reprogrammed while still in the system, without the need for a separate programmer. Additionally, ISP uses the Serial Peripheral Interface (SPI) for synchronous serial communication between the programmer and the target microcontroller. The SPI consists of three wires: Serial Clock (SCK), Master In - Slave Out (MISO), and Master Out - Slave In (MOSI). By adding the power supply Vcc, GND, as well as a pin of ATmega32A-PU used to control the Reset of the target-microcontroller we have a six-wire interface for the programming function.

SPI is a preferred protocol for programming microcontrollers because it offers several advantages over other protocols. One of the main advantages is its high-speed data transfer capabilities. The SPI can transfer data at rates of up to several megabits per second, which makes it ideal for programming large amounts of data quickly and efficiently. Additionally, SPI is a simple and straightforward protocol that requires minimal setup and configuration, making it easy to implement and use.

The ATmega32A-PU is the master and the ATmega328p is the slave in this protocol. The protocol was designed specifically for the ATmega328p, which is one of the reasons why a premade programmer was not used. This resulted in a more small-sized, low-cost, and straightforward solution. The use of paged flash memory and the choice of ISP and the AVR910-ISP method for programming the ATmega328p allows for efficient use of memory, fast data transfer and a reliable and easy-to-use programming solution for students.

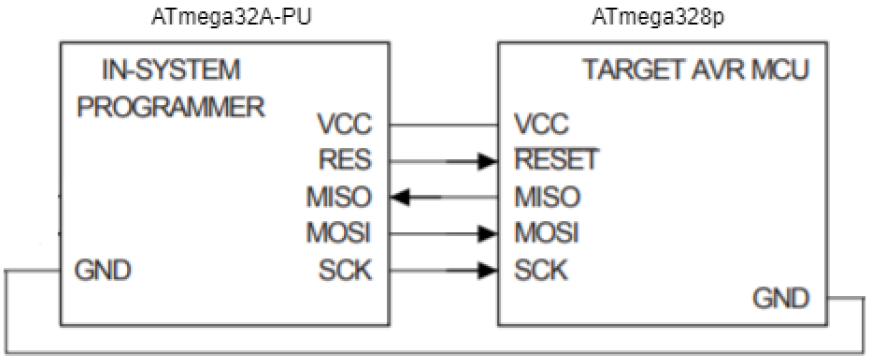

FIGURE 4.12: SPI interface.

The Serial Peripheral Interface (SPI) protocol is a widely-used method for synchronous serial communication between devices. In this implementation, the ATmega32APU serves as the programmer, and the target microcontroller is the ATmega328p.

The programming process begins by pulling the reset low and waiting at least 20ms before issuing the first commands. The first command, Programming Enable, is the only instruction accepted by the SPI interface when starting the process. The expected command must be received before the process can continue.

The second command, Read Device Code, reads the code of the device of the slave. In the case of the ATmega328p, the device code is 0x1E. This command is used to verify that the correct target device is being programmed.

The next two commands, Read Part Family and Read Part Number, also relate to the identification of the target microcontroller. These commands are used to ensure that the correct device family and part number are being programmed.

The ATmega328p has paged flash memory, which means that the data is first loaded into a page buffer and then written to flash memory when the buffer is full. This feature is taken into account in the programming protocol and the proper commands are issued to handle the paged memory.

The rest of the commands in the programming protocol to be executed and described below are used for the standard operations such as reading and writing data to the flash and EEPROM memory, as well as verifying the data that has been written. These commands are well-documented in the ATmega328p datasheet and are widely used in the programming of AVR microcontrollers using the ISP method.

Action	MOSI, sent to the target AVR	MISO, returned from the target AVR
Programming Enable	$AC 53 xx yy	$zz AC 53 xx
Read Device Code	$30 nn 00 mm	$yy 30 nn 1E
Read Part Family and Flash Size	$30 nn 01 mm	$yy 30 nn (Read Part Family and Flash Size)
Read Part Number	$30 xx 02 yy	$mm 30 xx (Part Number)

TABLE 4.1: ISP commands(0)

If the identification is correct the following command 4.2 erases the target-microcontroller. The command 4.2, Erase, is a crucial step in the programming process as it erases the existing data on the target-microcontroller before programming it with new code. This ensures that the target-microcontroller is in a clean state and ready to receive new data. It is important to note that this command is done automatically during the programming process, but it is also available as an option for the user to manually erase the target-microcontroller at any time. This flexibility allows the user to erase the target-microcontroller and start fresh whenever necessary, ensuring that the target-microcontroller is in a known state before programming it.

Action	MOSI, sent to the target AVR	MISO, returned from the target AVR
Erase Chip	$AC 8x yy nn	$zz AC 8x yy
Wait N ms		
Release RESET to end the erase		

TABLE 4.2: ISP commands(1)

Then, after the previous steps have been correctly executed, the data is first loaded into a page buffer with the Load Program Memory Page command. This command is executed until the page buffer is full. The ATmega328P, being the target microcontroller used in this implementation, has a page size of 256 bytes which means that the page buffer will be filled with 256 bytes of data before being written to the flash memory. When the page buffer is full, with the Write Program Memory Page command the page buffer is written to the flash memory of the ATmega328P.

Finally, to ensure that the data has been written correctly and to verify the programming procedure, the Read Flash Program Memory command is used to read the data from the flash memory at specific addresses. This command reads a single byte from the flash memory at a time and compares it to the data that was originally loaded into the page buffer. In this way, the programming procedure is verified and any errors can be identified and corrected.

Action	MOSI, sent to the target AVR	MISO, returned from the target AVR
Load Program Memory Page	$48 xx address data	-
Write Program Memory Page	$4C (adr MSB) (adr LSB) xx	-
Read Flash Program Memory	$28 (adr MSB) (adr LSB) xx	$xx yy zz (data)

TABLE 4.3: ISP commands(2)

The programming process described above is a crucial step in the development of the target-microcontroller, in this case the ATmega328P. The use of the SPI protocol for programming the microcontroller proves to be an effective method as it allows for fast and efficient communication between the master and the slave.

One of the benefits of using the SPI protocol is its simplicity. The standard format for commands consisting of 4 bytes makes it easy for the

master to send commands and for the slave to respond. This results in a streamlined programming process that minimizes the risk of errors.

Additionally, the use of the paged memory flash in the ATmega328P allows for efficient programming as it allows for data to be loaded into a page buffer before being written to the flash memory. This ensures that the programming process is done in a timely manner and minimizes the risk of data loss.

Furthermore, the use of the Read Flash Program Memory command allows for verification of the programming procedure. This ensures that the data is correctly written to the flash memory and minimizes the risk of errors.

Overall, the use of the SPI protocol and the specific instructions implemented in this case, proves to be an effective method for programming the ATmega328P microcontroller. It allows for fast and efficient communication between the master and the slave, minimizes the risk of errors, and ensures that the programming process is done in a timely manner.

Chapter 5

System Verification and Evaluation

This chapter aims to provide a comprehensive analysis of the performance and functionality of the programmer. This includes an examination of the code used, the results obtained from testing, and any relevant observations made during the programming process. The chapter will present and discuss the data collected, including any graphical representations of the results, and provide a thorough evaluation of the programmer's capabilities and limitations

The process of testing the programmer begins by connecting to the IP provided by the ESP8266. I enter the IP address into a web browser. Once connected, I am directed to the login page and prompted to enter the credentials that have been previously configured. These credentials ensure that only authorized users have access to the programming interface. After entering the credentials, I am redirected to the main page of the programming interface.

On the main page, I upload the bin file that I need to program the microcontroller with, which in this case is simple code that blinks a led. The file is generated by the Microchip Studio, but as it does not provide the file in the binary format needed for programming, I convert it from hex to binary. I used the WInHex [22] converter tool to do this, as it is a powerful hex editor that can also convert hex files to binary. However, there are other hex to binary converters available as well, such as HxD, Hex Workshop, and more.

At this point, it should be noted that while the Arduino terminal was used for convenience to monitor the programming process, any terminal can be used and it ultimately depends on the developer's preference. However, using the Arduino IDE terminal has the added benefit of being able to

easily connect to the ESP8266 UART, which is configured to a specific bandwidth speed. The typical baud rate for ESP8266 is 115200 baud/s, but it can be configured to other speeds as well, depending on the developer's needs.

Subsequently,when I uploaded the file, it is noticed that the target microcontroller's LED was blinking at the rate that I had programmed into the code. I also checked in the terminal, which I was using to monitor the process, and confirmed that the process had been executed correctly. Additionally, I tweaked some inputs on the target microcontroller and observed that the LED turned on and off depending on the inputs I had provided. I also used the options, such as Erase Chip, Verify, Log Out, and Reset, which were all successful and had a satisfactory response time. For a more in-depth understanding of the target microcontroller programming process, I have included a corresponding video that demonstrates the entire process and all the steps taken in detail.

It is crucial to note that the code being used is specifically tailored for the purpose of blinking an LED, and the inputs do not affect the outputs in this test scenario. Additionally, as an extra verification measure, several pins on the I/O expander, which is responsible for reading the outputs of the programmer, were connected to buttons. Pressing these buttons resulted in the corresponding outputs on the webpage turning green, and releasing them turned the outputs red again.Furthermore, I have included some screenshots that show the successful responses that I received as further illustration of the process.

The following figure 5.1 provides a clear visual representation of the sequence of events that occurred during the programming process. It illustrates the AT commands and the HTML pages that the ESP8266 transmitted during the programming process, providing a comprehensive overview of the steps taken during the programming process.

FIGURE 5.1: Visual representation of programmer process logs.

The following figure 5.2 provides a visual representation of the programmer in action. It captures a moment during the programming process and allows the viewer to see the physical components and their arrangement. This image provides valuable context and an understanding of the setup used during programming.

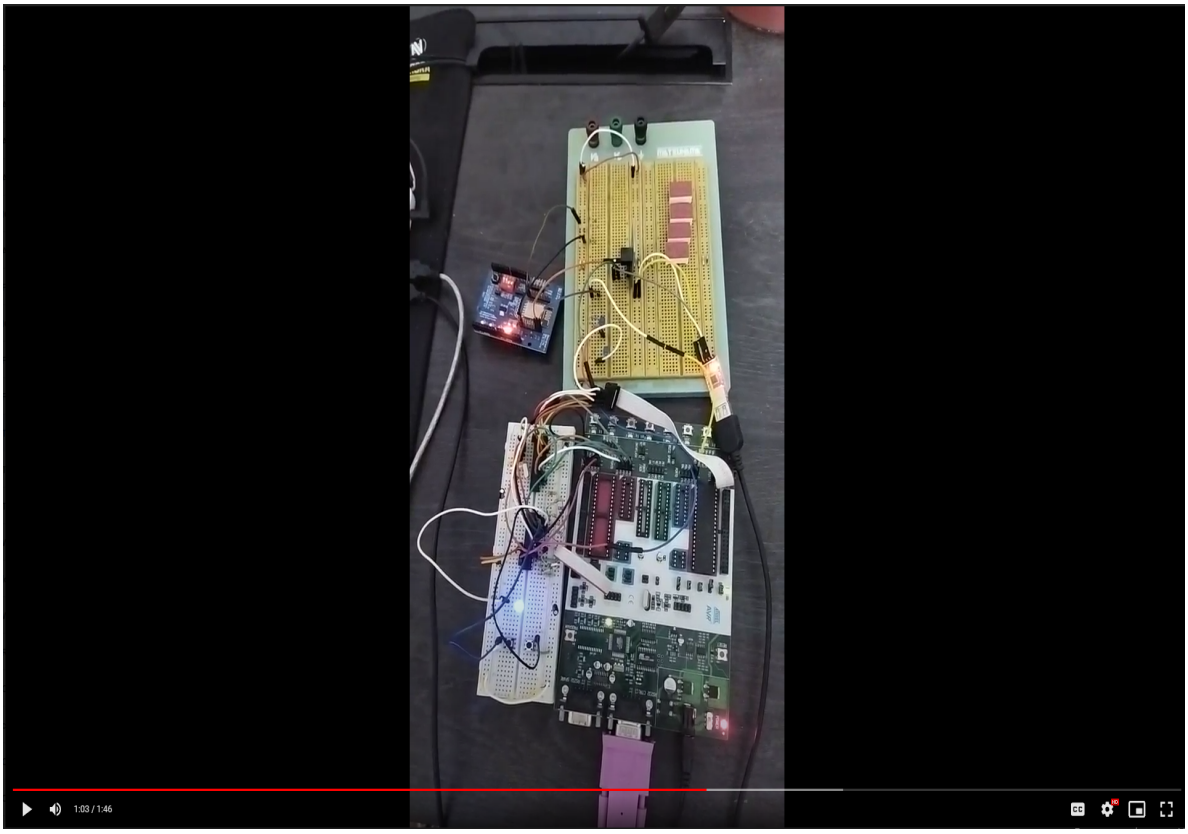

FIGURE 5.2: Real-time Programming Image.

A visual representation of the webpage interface during the programming process can be seen in figure 5.3.It illustrates a snapshot of the webpage interface being utilized during the programming process. It captures a specific moment in time during the programming process, providing a clear and detailed view of the interface and the information displayed. The figure highlights the OUTPUT PINS section, where it can be observed that when the LED is on, the color of the corresponding PIN is green, confirming the correct indication of the outputs of the target microcontroller. Additionally, the Response Panel is shown, displaying the messages received when the PORTC4 and PORTC0 ports were set to 0 and 1, respectively.

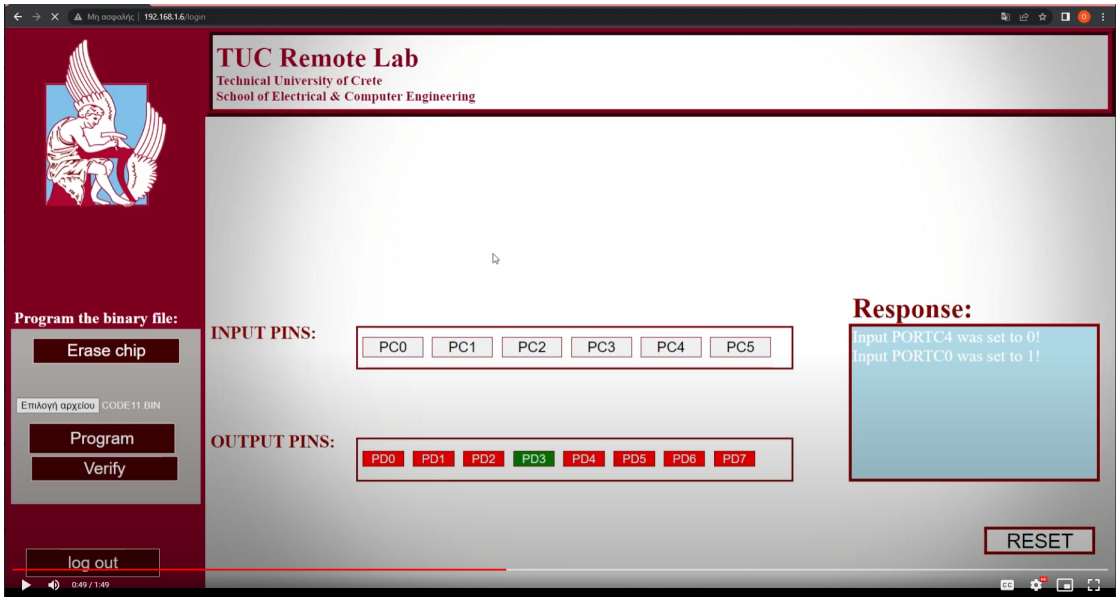

FIGURE 5.3: Webpage Interface during Programming Process.

During the implementation of this program, I faced a range of issues, from simple wiring problems to more significant ones. One of the challenges I encountered was the inconsistent behavior of the responses from the ESP8266 when using different browsers and Wi-Fi networks. The speed of each Wi-Fi network affected the way I was handling the responses, leading to an unstable and unreliable system.

To address this issue, I delved deeper into the response handler and discovered ways to optimize and improve its efficiency. I made changes to the handler to ensure that it could handle the responses from the ESP8266 in a more consistent and stable manner, regardless of the speed of the Wi-Fi network or the browser being used. These changes allowed me to create a full, stable system that provided reliable and accurate responses every time.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In conclusion, the remote programmer developed in this project serves as a valuable tool for the remote programming of embedded systems over the internet. Its user-friendly interface, ease of use, and reliability make it an ideal solution for students and researchers at the University to access and program embedded systems from any location. The remote access feature of this programmer allows for the programming of embedded systems from any location, making it more accessible for students and researchers, and also eliminates the need for expensive dedicated programming equipment, resulting in a cost-effective solution for the University. The remote programmer can greatly improve the workflow of embedded systems development and testing by allowing for faster and more efficient debugging and uploading of code. Additionally, it is worth mentioning that there are many useful libraries available that can be utilized to further enhance the functionality and performance of the remote programmer. This technology has the potential to streamline the process of embedded systems programming and development within the University, making it more convenient and accessible for students and researchers. It has the potential to revolutionize the way we approach embedded systems programming and development within the University, fostering a more innovative and productive learning environment.

6.2 Future Work

The development platform implemented in this thesis is intended to be widely used in the laboratory of the Embedded Microprocessor Systems course. While the graphical interface of the platform offers all the basic functions needed for

the implementation of laboratory experiments, there are a few minor modifications that could be made in order to further optimize the system's functionality and usability.

Currently, the system relies on an STK500 as the main power supply and development platform, which can be a limiting factor in terms of space and accessibility within the lab. A more convenient and standalone solution would be to create a custom PCB that could be used in place of the STK500, which would save space and provide a more efficient, streamlined solution for the lab. One of the main issues that arise when working with remote programmers is the inability of users to access the system when they are outside of the University's network. To overcome this problem, there are a few solutions that could be implemented:

- Port Forwarding: Port forwarding [23] is a method of making a device on your home or business network accessible over the Internet. It involves configuring your router to forward incoming connections on a specific port to a specific device on your network. This would allow the user to access the system remotely over the internet by forwarding incoming connections on the specific port to the device running the remote programmer.
- VPN (Virtual Private Network): Another solution that could be implemented is the use of a VPN [24] service. A VPN service allows the user's remote computer to access the system as if it were connected to the University's network. This is achieved by creating a secure, encrypted tunnel between the user's computer and the University's network. This solution is considered to be more secure than port forwarding, as it encrypts all data passing through the tunnel, making it much more difficult for anyone to intercept the data.

In conclusion, this thesis presents potential additions that are aimed at enhancing the functionality and usability of our remote programmer in a laboratory setting. These improvements are specifically designed to support the development of embedded systems and the creation of a better experience for students and researchers using this platform.

One such addition is the integration of an LCD module on the PCB, which would provide real-time information to the laboratory staff on the status of the programming process. This would allow laboratory staff to monitor the process of the student experiment and ensure its smooth running.

Additionally, the addition of a camera that interacts with the web-page would greatly enhance the debugging process for students. The camera would provide a visual representation of the embedded system, allowing students to see the results of their code in real-time. This would make it easier for students to understand how their code is affecting the embedded system and improve their ability to test and debug their code. Furthermore, there are cost-effective solutions in the market that can be integrated into the remote programmer, making it a more accessible tool for educational institutions. This integration would allow students to benefit from the interactive and hands-on experience provided by the camera and further aid in their understanding and development of embedded systems

Another potential future addition would be the improvement of the handler of the Wi-Fi module responses. This would not only add more functionality to the remote programmer but also allow for more features to be added to the web-page. The improved and adaptable handler would provide a more reliable and efficient connection between the remote programmer and the web-page. These modifications would not only make the remote programmer a more powerful tool for the development of embedded systems but also enhance the experience of students and researchers using the platform by providing them with more options and features for testing and debugging their code.

Finally, the current remote programmer only supports one student to test their code at a time. A future addition with more target microcontrollers and the web-page being available to more students would allow multiple students to test their code simultaneously, and significantly improve the remote programmer and its functionality. These modifications would make the remote programmer a more powerful tool for the development of embedded systems and further enhance the productivity and innovation of students and researchers at the university.

References

[4] F. Yudi Limpraptono et al. “The design of embedded web server for remote laboratories microcontroller system experiment”. In: (2011), pp. 1198–1202. URL: https://ieeexplore.ieee.org/abstract/document/6129302.

[5] Constantine-George Kottikas. “Design and Implementation of a Logic Design Experiment Platform Based on AVR Microcontroller and JAVA Graphics Application”. In: *Technical University of Crete* (2009). URL: https://dias.library.tuc.gr/view/12276.

[6] Lykos Emmanouil. “Design and Implementation of an Integrated System for Logic Design exercises”. In: *Technical University of Crete* (2020). URL: http://purl.tuc.gr/dl/dias/C41217CF-87C0-4F90-B52B-CBF035E966B5.

[7] Patrik Jacko, Milan Guzan, and Andrii Kalinov. “Remote Microcontroller Scanner Design for STM32 Microcontrollers Used to Distance Education Form”. In: (2021), pp. 1–6. URL: https://ieeexplore.ieee.org/abstract/document/9598723.

[8] Wael Badawy et al. “On Flashing Over The Air “FOTA” for IoT Appliances – An ATMEL Prototype”. In: (2020), pp. 1–5. URL: https://ieeexplore.ieee.org/abstract/document/9352203.

[9] Tomasz Michalec et al. “Remote Programming and Reconfiguration System for Embedded Devices”. In: (2019), pp. 467–470. URL: https://ieeexplore.ieee.org/abstract/document/8859983.

[12] Reza Hashemian and Jason Riddley. “FPGA e-Lab, a technique to remote access a laboratory to design and test”. In: *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)* (2007), pp. 139–140. URL: https://ieeexplore.ieee.org/abstract/document/4231487.

[13] Ahmed A. et al. “Wireless ATMEL AVR In-Circuit Serial Programmer based on Wi-Fi and ZigBee”. In: *2020 16th International Computer Engineering Conference (ICENCO)* (2020). URL: https://ieeexplore.ieee.org/document/9357392.

[14] Parkhomenko Anzhelika et al. "Development and Application of Remote Laboratory for Embedded Systems Design." In: *Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV). IEEE* (2015). URL: https://ieeexplore.ieee.org/abstract/document/7087265.

[15] "ESP-12E WiFi Module". In: (2015). URL: https://components101.com/sites/default/files/component_datasheet/ESP12E%20Datasheet.pdf.

[17] "I/O EXPANDER MCP23017 Module". In: (2006). URL: https://ww1.microchip.com/downloads/en/devicedoc/20001952c.pdf.

[21] "AVR910: In-System Programming". In: (2016). URL: http://ww1.microchip.com/downloads/en/appnotes/atmel-0943-in-system-programming_applicationnote_avr910.pdf.

External Links

[1] *labster*. URL: https://www.labster.com/.

[2] *digi*. URL: https://www.digi.com/products/iot-software-services/digi-remote-manager.

[3] *Remote Desktop*. URL: https://www.cyberlinkasp.com/insights/remote-desktop-work/.

[10] Haider Al-Shammari et al. *Energy efficient service embedding in IoT networks*. Apr. 2018. URL: https://www.researchgate.net/figure/Block-diagram-of-IoT-Node_fig2_325638412.

[11] *labview*. URL: https://www.ni.com/en-us/shop/software/products/labview.html.

[16] *Transmission Control Protocol*. URL: https://en.wikipedia.org/wiki/Transmission_Control_Protocol.

[18] *JavaScript*. URL: https://el.wikipedia.org/wiki/JavaScript.

[19] *HTML*. URL: https://el.wikipedia.org/wiki/HTML.

[20] *CSS*. URL: https://el.wikipedia.org/wiki/CSS.

[22] *WinHex*. URL: https://www.x-ways.net/winhex/.

[23] *PortForward*. URL: https://en.wikipedia.org/wiki/Port_forwarding.

[24] *VPN*. URL: https://en.wikipedia.org/wiki/Virtual_private_network.

Appendix A

User Manual

This appendix serves as a user manual for the developed platform, offering step-by-step instructions and guidelines on how to effectively utilize the system's features and functionalities. The manual is designed to assist students in accessing the remote laboratory and performing programming and control operations on the target microcontroller.

User Manual:

Before beginning the experiment, it is important to note the available outputs and inputs of the ATmega328p microcontroller. The outputs include PD0-PD7, while the inputs are PC0-PC5. Additionally, ensure that you have the necessary file in BIN format for programming the microcontroller. If you have a hex file, you can use tools like WinHex to convert it to BIN.

Follow the steps below to program the microcontroller using the Remote Programmer page:

1. Log in to the Remote Programmer page using the provided link (credentials will be provided by the lab).
2. Enter the login credentials on the displayed page to access the main page.
3. Once successfully logged in, you will be directed to the main page.
4. Click on the "Program" button and select the BIN file you want to program.
5. Check the "Response Panel" for a message indicating successful programming.

6. The "OUTPUT PINS" section displays the outputs of the programmed microcontroller.
7. At this stage, you can perform various actions:
    - Set inputs to the microcontroller by pressing the corresponding "INPUT" buttons on the main page.
    - Reset the microcontroller
    - Erase the microcontroller
    - Reprogram the microcontroller with another code by pressing "Program" again and selecting another BIN file
8. Once you have completed your experiment, press "Log Out" to end your session. Note that the system currently does not support multiple users simultaneously.

The Response Panel serves as a real-time feedback mechanism, alerting you to any issues that may arise during the experiment. If you have any questions or encounter any difficulties, contact the laboratory staff.

Additionally, a tutorial video is provided to guide you through the experimental procedure. This video allows you to visualize each step and gain a better understanding of the process. Ensure that you follow these instructions carefully to make the most out of the platform's programming and control capabilities.

Appendix B

Javascript code

Below is the Javascript code which provides the synchronous update of the elements of the main web page:

```
<script>
  function colorGreen(id) {
    var el = document.getElementById(id);
    el.setAttribute('class', 'classB');
  }

  function colorRed(id) {
    var el = document.getElementById(id);
    el.setAttribute('class', 'classA');
  }
  //Updates Response Panel
  function getData(x) {
    var xhttp = new XMLHttpRequest();
    xhttp.open('GET', x, true);
    xhttp.send();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById('R').innerHTML = this.responseText;
      }
    };
  }
  //Updates Color of Output Pins
  function getPINS() {
    var xhttp = new XMLHttpRequest();
    xhttp.open('GET', 'OUTPINS', true);
    xhttp.send();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        if (this.responseText[1] == '1') {
          colorGreen('d0');
        } else if (this.responseText[1] == '0') {
```

```
            colorRed('d0');
          }
          if (this.responseText[2] == '1') {
            colorGreen('d1');
          } else if (this.responseText[2] == '0') {
            colorRed('d1');
          }
          if (this.responseText[3] == '1') {
            colorGreen('d2');
          } else if (this.responseText[3] == '0') {
            colorRed('d2');
          }
          if (this.responseText[4] == '1') {
            colorGreen('d3');
          } else if (this.responseText[4] == '0') {
            colorRed('d3');
          }
          if (this.responseText[5] == '1') {
            colorGreen('d4');
          } else if (this.responseText[5] == '0') {
            colorRed('d4');
          }
          if (this.responseText[6] == '1') {
            colorGreen('d5');
          } else if (this.responseText[6] == '0') {
            colorRed('d5');
          }
          if (this.responseText[7] == '1') {
            colorGreen('d6');
          } else if (this.responseText[7] == '0') {
            colorRed('d6');
          }
          if (this.responseText[8] == '1') {
            colorGreen('d7');
          } else if (this.responseText[8] == '0') {
            colorRed('d7');
          }
          getPINS();
        }
      };
    }
    setTimeout(getPINS, 1000);
</script>
```

Appendix C

Block Diagram

Below is the block diagram of the platform implemented in this thesis:

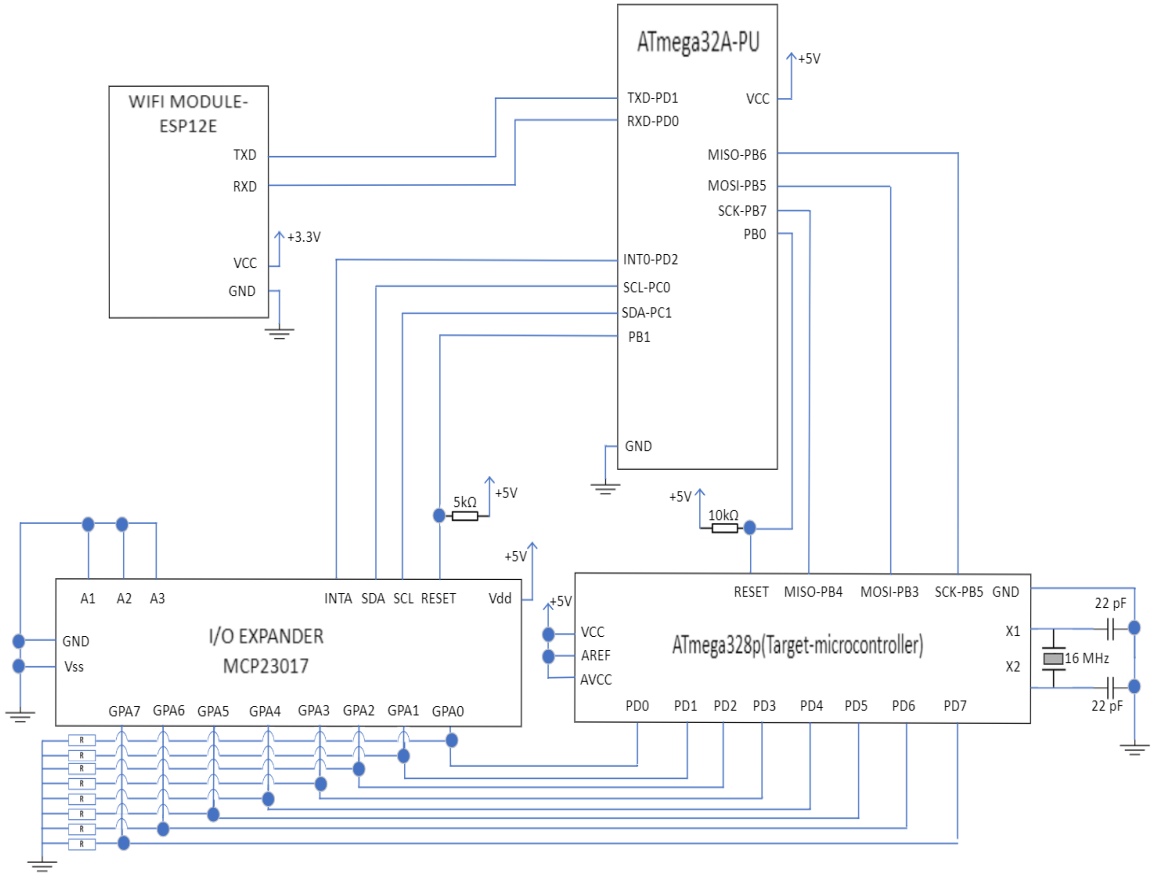

FIGURE C.1: Block diagram of the Platform.