

Laboratory manuals

UbiLAB



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



Erasmus+



NATIONAL AGENCY
for European Educational
Programmes and Mobility

Table of Contents

Table of Contents.....	1
1 Introduction	6
2 Programmable Logic Controllers Laboratory	8
2.1 Introduction	8
2.2 Laboratory equipment.....	8
2.3 System architecture for remote access.....	11
2.4 Laboratory experiment	13
2.4.1 Introduction.....	13
2.4.2 Remote access procedure	13
2.4.3 Goals of the experiment.....	13
2.4.4 Experimental setup	14
2.4.5 Experimental results	20
2.5 Conclusion.....	20
2.6 References	21
2.7 Acknowledgements	21
3 Scilab Virtual Software Laboratory	23
3.1 Introduction	23
3.2 Scilab	24
3.3 Laboratory experiment 1: AM Modulation.....	24
3.3.1 Goals of the experiment.....	24
3.3.2 Experimental setup	25
3.3.3 Experimental procedure.....	25
3.3.4 Experimental results	27
3.4 Laboratory experiment 2: AM Demodulation	27
3.4.1 Goals of the experiment.....	27

3.4.2	Experimental setup	27
3.4.3	Experimental procedure.....	28
3.4.4	Experimental results	29
3.5	Laboratory experiment 3: Digital Modulation Techniques	29
3.5.1	Goals of the experiment.....	29
3.5.2	Experimental setup	30
3.5.3	Experimental procedure.....	30
3.5.4	Experimental results	33
3.6	Conclusion.....	34
4	Digital Signal Processing Laboratory.....	36
4.1	Introduction	36
4.2	Real-Time Audio Filtering.....	37
4.2.1	Experimental setup	37
4.2.2	Goal of the experiment.....	39
4.2.3	Experimental results	39
4.3	Real-Time Moving Spectrogram.....	41
4.3.1	Experimental setup	42
4.3.2	Goals of the experiment.....	43
4.3.3	Experimental results	43
4.4	Real-Time PWM Signal Generation	45
4.4.1	Experimental setup	46
4.4.2	Goals of the experiment.....	46
4.4.3	Experimental results	46
4.5	Conclusion.....	48
5	Embedded Systems Laboratory.....	50
5.1	Introduction	50
5.2	Laboratory experiment 1.....	51

5.2.1	Experimental setup	52
5.2.2	Goals of the experiment.....	52
5.2.3	Experimental results	52
5.3	Laboratory experiment 2.....	53
5.3.1	Experimental setup	54
5.3.2	Goals of the experiment.....	54
5.3.3	Experimental results	55
5.4	Laboratory experiment 3.....	56
5.4.1	Experimental setup	57
5.4.2	Goals of the experiment.....	57
5.4.3	Experimental results	57
5.5	Laboratory experiment 4.....	59
5.5.1	Experimental setup	60
5.5.2	Goals of the experiment.....	60
5.5.3	Experimental results	60
5.6	Laboratory experiment 5.....	61
5.6.1	I2C transmission.....	62
5.6.2	Experimental setup	63
5.6.3	Goals of the experiment.....	63
5.6.4	Experimental results	63
5.7	Conclusion.....	67
6	Virtual Instrumentation Laboratory Platform	69
6.1	Introduction	69
6.2	Laboratory experiment 1: Voltmeter accuracy verification (calibration).....	71
6.2.1	Experimental setup	72
6.2.2	Experimental results	74
6.3	Laboratory experiment 2: Verification of the first Kirchhoff law.....	77

6.3.1	Simulation of ideal electrical circuit.....	78
6.3.2	Experimental setup	80
6.4	Laboratory experiment 3.....	82
6.4.1	Experimental setup	83
6.4.2	Experimental results	85
6.5	Least squares approximation	86
6.6	Conclusion.....	86
7	System Administration Laboratory.....	88
7.1	Introduction	88
7.2	Gaining remote access.....	90
7.3	Laboratory experiment 1: Accessing the Command Line.....	92
7.3.1	Experimental setup	92
7.3.2	Goals of the experiment.....	92
7.3.3	Experimental results	92
7.3.4	Task.....	92
7.4	Laboratory experiment 2: Managing Files from the Command Line	94
7.4.1	Experimental setup	94
7.4.2	Goals of the experiment.....	94
7.4.3	Task.....	94
7.4.4	Experimental results	96
7.5	Laboratory experiment 3: Editing Files Using Vim's Visual Mode.....	97
7.5.1	Experimental setup	97
7.5.2	Goals of the experiment.....	97
7.5.3	Experimental results	97
7.5.4	Task.....	98
7.6	Conclusion.....	99
8	Automation System Control Laboratory	101

8.1	Introduction	101
8.2	Laboratory experiment 1 - Dual Temperature Control System.....	102
8.2.1	Experimental setup	102
8.2.2	Goals of the experiment.....	103
8.2.3	Experimental results	105
8.3	Laboratory experiment 2 - Closed-loop Control	107
8.3.1	Experimental setup	107
8.3.2	Goals of the experiment.....	107
8.3.3	Experimental results	107
8.4	Laboratory experiment 3 - Wind Levitation System.....	109
8.4.1	Experimental setup	109
8.4.2	Goals of the experiment.....	111
8.4.3	Experimental results	112
8.5	Conclusion.....	114
9	Authors of Virtual Laboratory Manuals	115

1 Introduction

The UbiLAB (A **ub**iquitous virtual **lab**oratory framework) project aims towards creating a framework for ubiquitous virtual, remote and software laboratories implemented in the cloud. The project took place from 2020-2023, and was financed by the Erasmus+ Programme under the number 2020-1-MK01-KA226-HE-094548.

The project partners are three higher education organizations: The Faculty of Electrical Engineering and Information Technology (FEEIT) at the University Ss. Cyril and Methodius in Skopje, N. Macedonia, the University of Maribor (UM FER) in Slovenia, and the Anhalt University of Applied Sciences (HSA) in Germany.

As a result of the limitations imposed by the Covid-19 pandemic, the presence of students in the university laboratories has been significantly limited. One of the most affected aspects of students' study experience are the laboratory experiments that used to be conducted on-site, in the laboratories. This project worked towards implementing digital technologies in the process of laboratory work and experiments. These are rarely completely transferrable into the digital world. Nevertheless, today's challenges have pushed forward the need to intensively explore down this road and provide the best possible solutions.

The project resulted in an innovative framework, designed to support different types of endpoints: hardware devices, virtual devices, software solutions. This UbiLAB framework presents a foundation for a rich and diverse set of laboratory experiments, enables customizable modules supporting software and hardware exercises, promotes collaborative engagement on several levels, and uses open-source software which provides reusability and flexibility.

This document contains the manuals for the remote laboratory exercises designed as part of the project's fifth intellectual output. It contains detailed experiment implementation guidelines in order to optimally exploit the effects of the remote virtual laboratory learning approach.

The laboratory manuals given in this document can also be accessed at <https://github.com/UbiLAB-project/UbiLAB-Laboratory-manuals>, along with the respective code files.

The UbiLAB framework can be accessed at <https://github.com/UbiLAB-project/UbiLAB-Framework>.

Laboratory manual

Programmable Logic Controllers Laboratory

UbiLAB



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



2 Programmable Logic Controllers

Laboratory

2.1 Introduction

This remote laboratory exercise is intended for a course named Programmable Logic Controllers (PLC). The laboratory equipment contains a PLC and a personal computer (PC) that would be connected within the university network. In addition, a conveyor belt process is provided which should be electrically connected with the PLC. This means, that the remote laboratory exercise would require a one-time physical access in order to complete the electrical connection. After the electrical connection has been completed, the students will be required to develop a control program for the process and test its operation afterwards.

The physical access laboratory experiment involves interaction between the students and the conveyor belt. In order for the students to observe their results, they put metallic and non-metallic objects on the conveyor belt and afterwards they observe the output from the sensors. They also interact with the push buttons or the analog potentiometer from the process. In the remote access laboratory, this interaction can be realized by involving augmented reality (AR) with virtualized objects, push buttons, potentiometer conveyed through a remote live stream of the conveyor.

For this occasion, students will be remotely connecting through Remote Desktop Protocol (RDP), facilitated through the Apache Guacamole framework, to each workstation placed within the laboratory. This will provide a remote workstation for each student in order to be able to develop a control program which would be uploaded to the PLC through the laboratory network. Afterwards, students can observe and interact with the conveyor belt by a specific web application deployed on the Moodle framework. The web application will live stream the conveyor belt from the laboratory and will concurrently utilize AR in order to enable the interaction with the virtualized objects. Each outcome of the interaction will be communicated to the PLC through the university network. The PLC program will act according to the received messages and students will be able to observe the results from their developed control program through the live video stream.

2.2 Laboratory equipment

The laboratory contains a total of eight workstations. Each workstation provides:

- Programmable logic controller: Mitsubishi FX3GE-24MT/DSS [1]
- Variable frequency drive: Mitsubishi D700-SC

- Programmable logic controller: Student made
- Voltage supply: 24VDC
- Conveyor belt process
- Personal computer with appropriate software
- Wires with appropriate safety terminal connectors
- Live stream camera

Each device provides electrical connection to its terminal through safety plug-in connectors shown on the board next to the device. Each connection is designated according to device's datasheet.

Figure 2-1 shows an example of a workstation with the mentioned equipment.

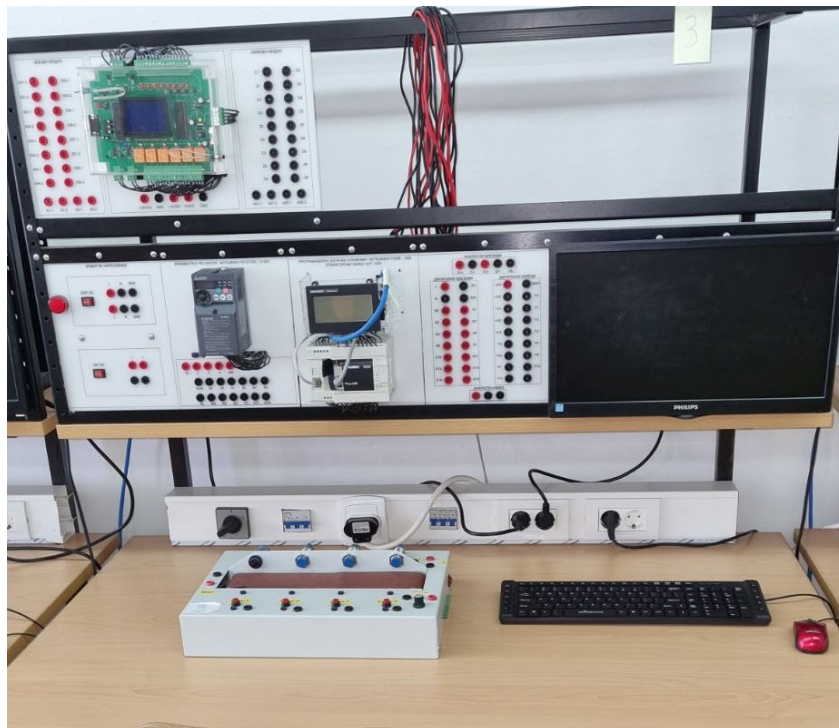


Figure 2-1. A single laboratory workstation.

The conveyor belt process is shown in Figure 2-2. The conveyor belt process includes four digital proximity sensors, of which three are inductive and one is photoelectric, four push buttons, of which one pair are momentary switches and the other are maintained switches, direct current (DC) motor for the conveyor and an analog potentiometer. It can be summarized that the plant in total requires 8 digital inputs, 1 digital output and 1 analog input from the PLC.

Each personal computer and PLC is connected to the university network. The full network topology is shown in Figure 2-3 and the actual network equipment in Figure 2-4.

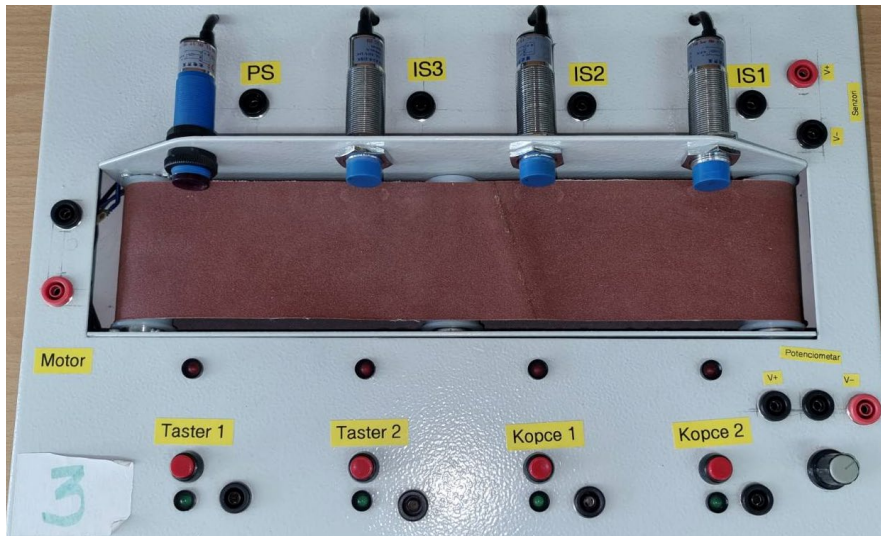


Figure 2-2. The conveyor belt process.

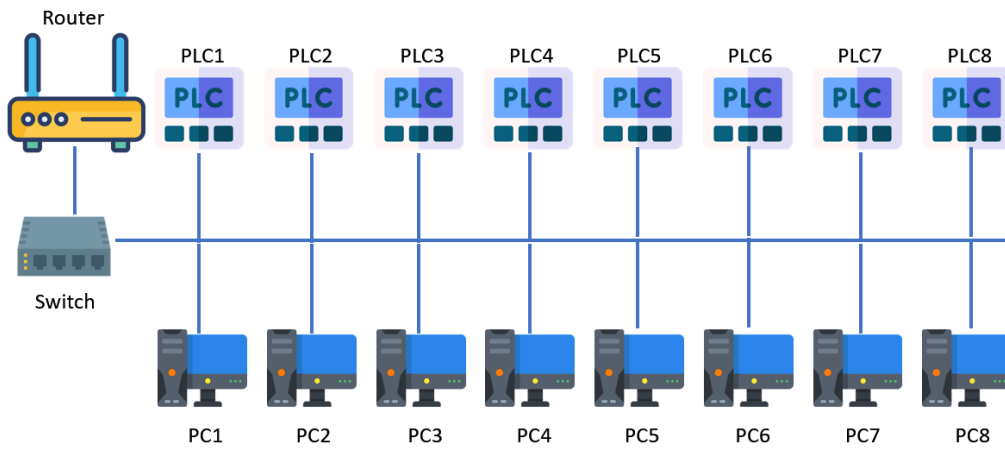


Figure 2-3. Network topology.

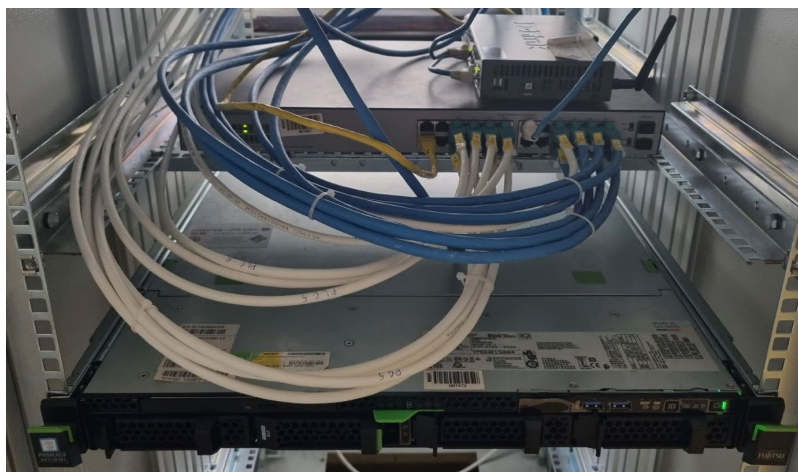


Figure 2-4. Network equipment used for the setup.

2.3 System architecture for remote access

Understanding the system architecture for remote access to the laboratory is important for executing remote laboratory experiments. The student will be required to establish two links in order to be able to access the laboratory:

- PC access
- Process access

The PC access will enable the student to remotely access the personal computer within the laboratory. This eliminates the necessity for students to install appropriate PLC software on their computers, instead they can utilize the installed software on the laboratory computers. This can be realized through RDP, and facilitated by the Apache Guacamole Framework in order to be easily integrated into the UbiLAB framework.

The Moodle LMS implementation enables the student to access the AR application for interaction with the conveyor belt process. Students will log in to the UbiLAB framework and use the appropriate course links to enter the AR application. Accordingly the AR application will provide live stream of the conveyor belt process and virtual interaction with the sensors, push buttons and potentiometer. The complete architecture is shown in Figure 2-5.

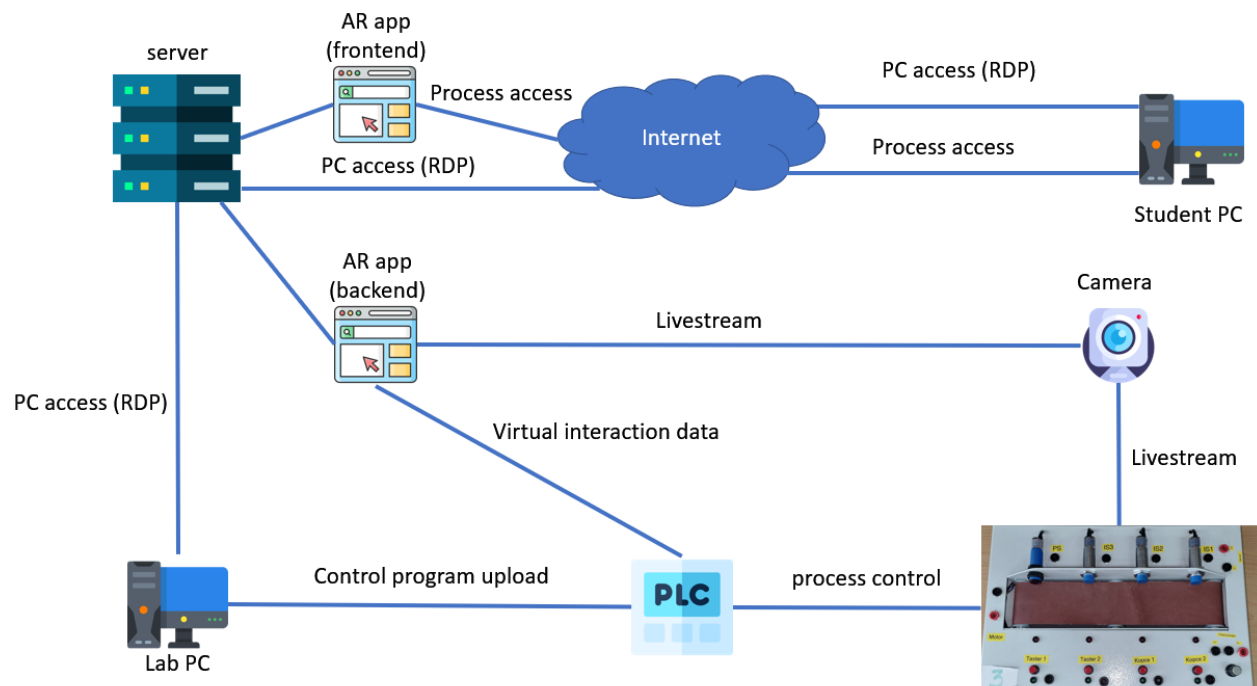


Figure 2-5. System Architecture.

In the physical access laboratory, students had to put real life objects in front of sensors, push real life buttons and interact with the potentiometer. The AR application virtualizes this interaction by adding push buttons and potentiometer. In addition, it also adds virtual objects on the live stream footage which can move with the conveyor belt accordingly. These objects can interact with virtual sensors by evaluating their mutual position through AR processing algorithms. This enables to asses if an object is in front of a sensor and therefore assesses the sensor output accordingly.

The frontend of the AR application is web based, hence easily accessible through any browser. It provides the preprocessed live stream footage along with the virtual push buttons and potentiometer. Figure 2-6 illustrates the interface of this application. Students can use the "Place Object" button to place or retrieve a virtual object on the conveyor. They can also interact with the push buttons and potentiometer. Each function of the interface widgets is shown in Table 2-1.

Table 2-1. Functions of the interface widgets.

Button name	Function
T1	Momentary switch
T2	Momentary switch
B1	Maintained switch
B2	Maintained switch
Place object	Maintained switch for placing and retrieving a virtual object from the conveyor belt
Potentiometer	Simulate a 0-10V on the analog input of the PLC

Figure 2-7 shows a real life use where the virtual object is shown in green along with appropriate sensor states (red – off; green - on).

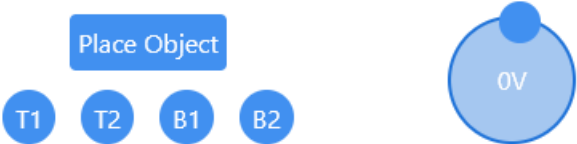


Figure 2-6. Application interface.

The backend of the same AR application retrieves live stream footage from the camera which is preprocessed using AR algorithms to add the objects and evaluate their interaction with the virtual sensors. In addition, it uses Melsec Communication protocol [2], from Mitsubishi, to communicate with the PLC. The states of the push buttons, potentiometer and virtual sensors are sent through modifying appropriate bits and registers in the PLC.

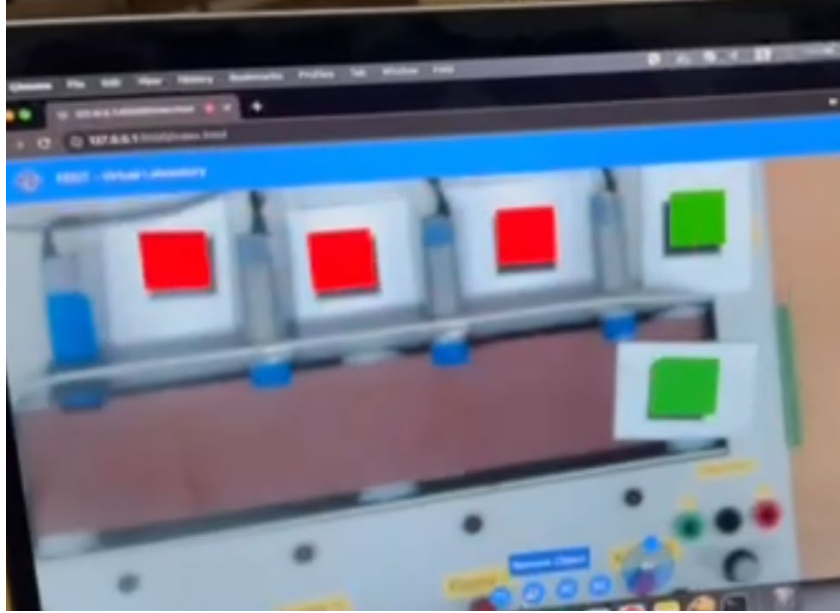


Figure 2-7. A detected virtual object on the conveyor belt.

2.4 Laboratory experiment

2.4.1 Introduction

In this laboratory experiment we are going to develop a control program for transporting objects on a conveyor belt. The objects are placed on the start of the conveyor belt and they should be transported on the other end. When an object arrives at the end, a photoelectric sensor proximity sensor detects its arrival and the conveyor belt should stop moving. As soon as the object is picked up, the conveyor belt should continue moving. The process can be started with a "start" push button and stopped with a "stop" push button. An emergency button should be utilized as well.

2.4.2 Remote access procedure

A general laboratory experiment requires the following steps for a remote access:

1. Student establishes access to laboratory PC through RDP (integrated into the framework)
2. Student logs in through his UbiLAB framework account
3. Student uses the appropriate course link to open the AR application
4. Student uses the lab. PC software to create and upload control program to PLC
5. Student uses the AR application to virtually interact with the conveyor belt process in order to evaluate his control program

2.4.3 Goals of the experiment

- Develop a control program for conveyor belt process
- Transport an object from one end to another
- Use the photoelectric sensor to detect if an object has arrived at the end
- Always stop the conveyor when an object arrives at the end

- The conveyor should continue moving after object is picked up at the end
- Use start and stop push buttons for starting and stopping the process
- Use emergency button for emergency states

2.4.4 Experimental setup

2.4.4.1 Electrical connection

In order to convey this experiment the following devices will be used:

- Programmable logic controller: Mitsubishi FX3GE 24MT/DSS
- Conveyor belt process
- Voltage supply: 24VDC

In order to execute the experiment the conveyor belt, shown in Figure 2-2, needs to be connected to the PLC along with appropriate voltage supply. Each PLC and Power supply connection terminal is accessible through safety plug-in connectors mounted on a wiring board, shown in Figure 2-8.

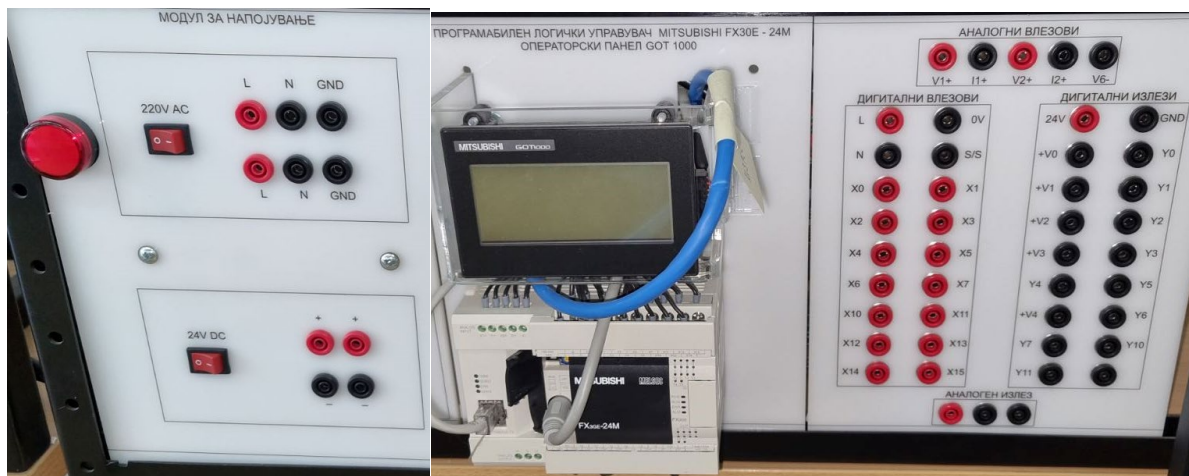


Figure 2-8. Wiring terminals for each PLC and power supply.

Using the schematics, shown in Figure 2-9, the following steps need to be executed in order to electrically connect the devices [3]:

- Connect the PLC to 24VDC power supply
- Connect the PLC digital output Y0 to one end of the DC motor and connect the other and to V- from the 12VDC power supply
- Connect the +V0 terminal, from the PLC, to the V+ terminal from the 12VDC power supply

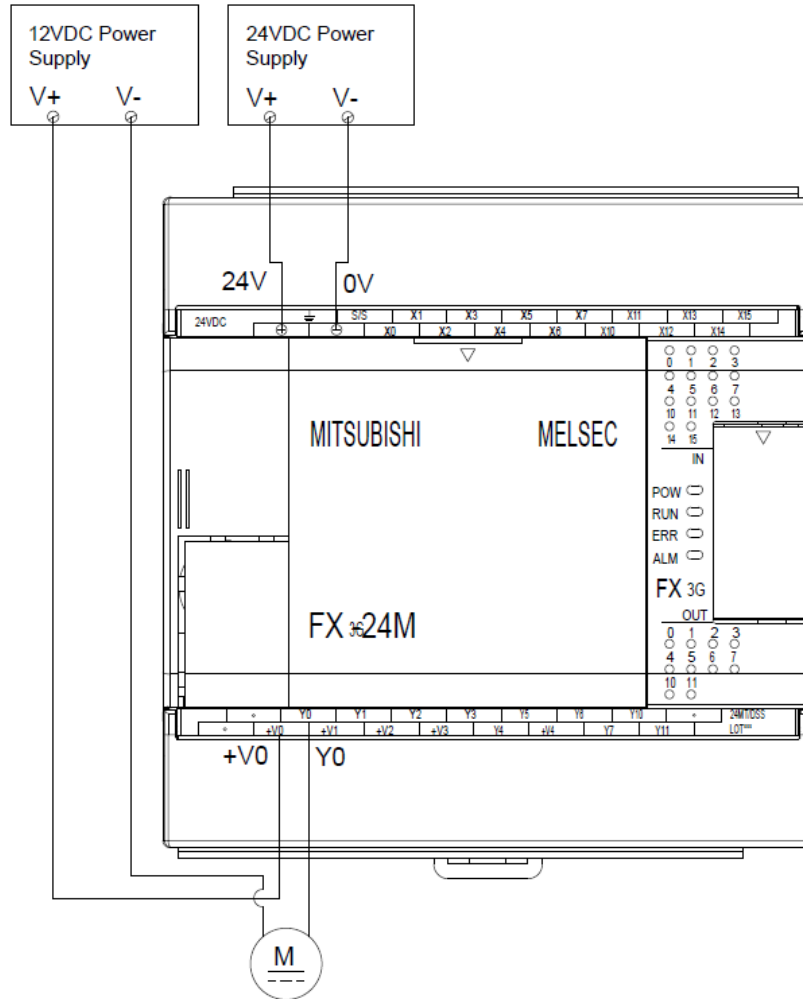
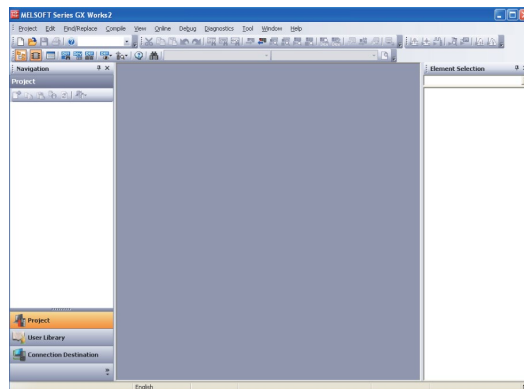


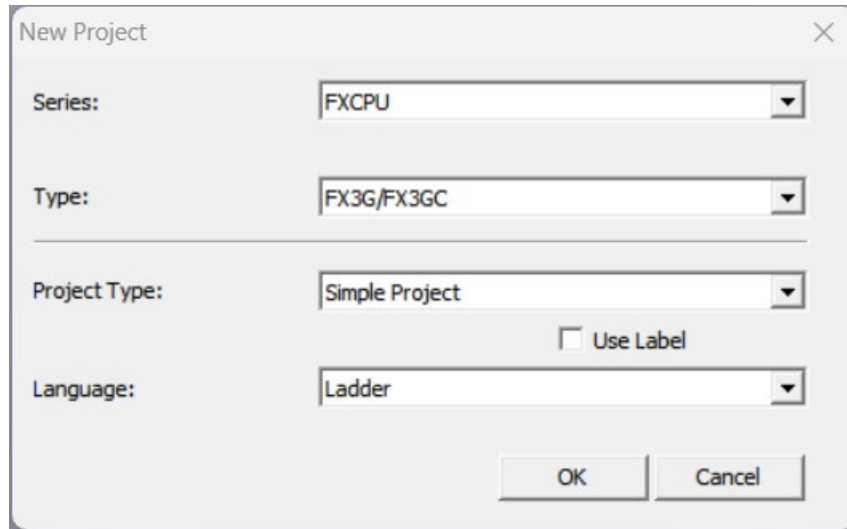
Figure 2-9. Schematic of the power supply connection to the PLC.

2.4.4.2 PLC software and program upload

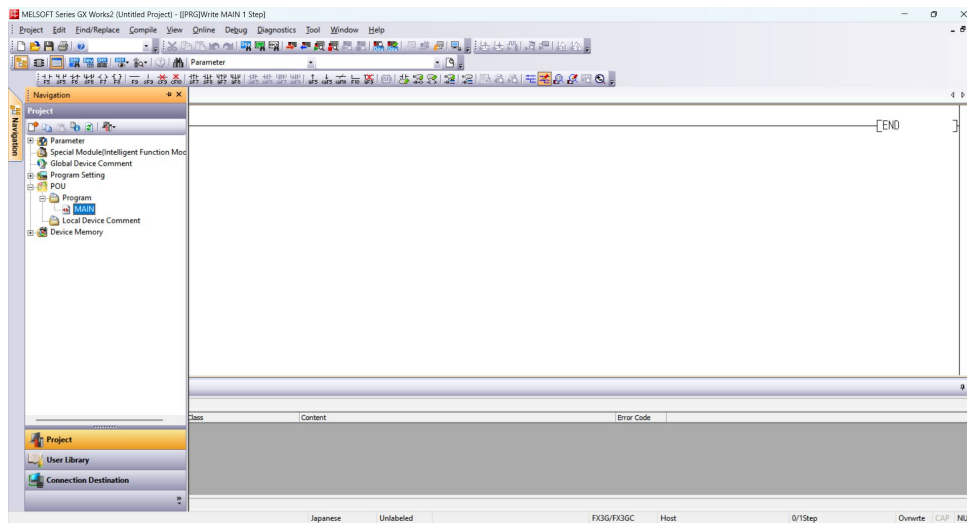
The PLC control program will be developed by using GXWorks2 software from Mitsubishi Electric. The following screen should appear when starting the software:



Create a new project using the menu bar by clicking **Project -> New**. A new window should appear allowing to choose PLC parameters. Choose the following parameters and click **OK**:



A new project should be created, displaying the following screen:



This project will use Ladder Logic programming language. This is a symbolic language which uses symbol instructions being graphically connected in order to program a control logic. Each symbol can be inserted within the program sheet divided by multiple square, each square for one instruction. Use the following symbols in order to graphically construct the ladder program.



Inserting an instruction can be realized by double clicking a square within the program sheet. A new window will appear as shown:



Choose the appropriate instruction and digital input/output address within the field.

The only sensor/actuator electrically connected to the PLC is the DC motor. Considering that we connected the DC motor to address Y0, we will use this address within our control program. Any additional sensor used within the control program is virtualized therefore we should use internal addresses for each virtualized sensor. The addresses should be used according to Table 2-2.

Table 2-2. Addresses and tags for the used devices.

Device	Tag	Address
Inductive sensor 1	IS1	M0
Inductive sensor 2	IS2	M1
Inductive sensor 3	IS3	M2
Photoelectric sensor	PS	M3
Momentary switch 1	T1	M4
Momentary switch 2	T2	M5
Maintained switch 1	K1	M6
Maintained switch 2	K2	M7
Potentiometer	Potenciometer	D0

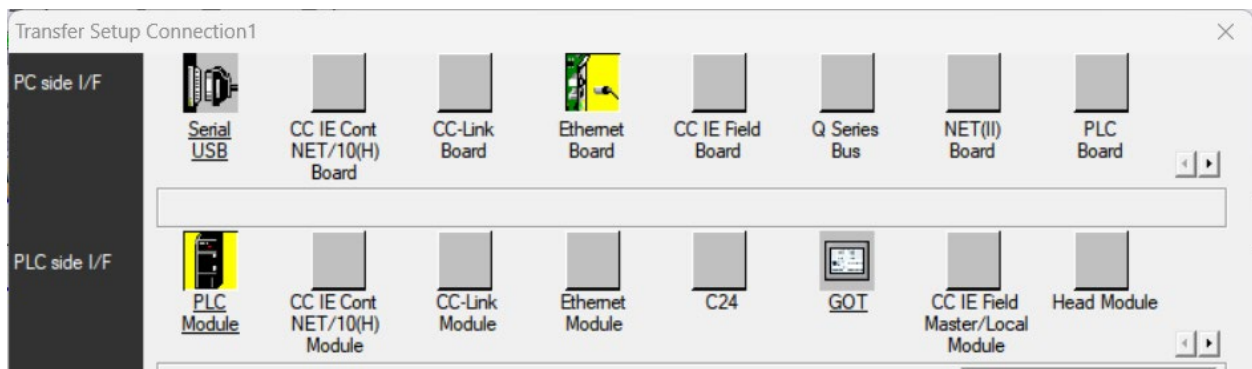
Use the push buttons T1, T2 and K1 as start, stop and emergency stop (NO) push button.

A control program solution is shown in Figure 2-10 [5]. We use the first rung in order to define the work state of an auxiliary bit with address M10. The conveyor belt process should work if this bit is logic 1 and otherwise if logic 0. The state of this bit depends on the ladder instruction states for the start, stop and emergency push button. The second rung defines whether or not the conveyor belt should be moving according to the work bit and photoelectric sensor state. Use the menu bar to compile the program by clicking **Compile->Build**.



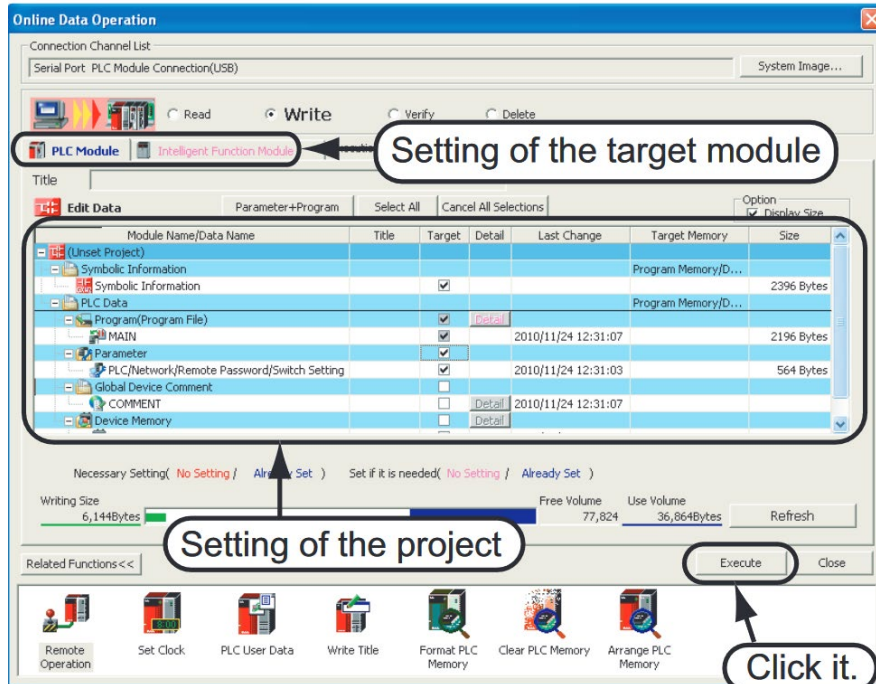
Figure 2-10. Control program solution.

Using the navigation panel go **Connection Destination** and double click **Connection1**. A new window should appear allowing to establish connection with the PLC. Choose **Ethernet Board** for PC side communication and **PLC Module** for the PLC side.



Double click the **PLC Module** to open setup the communication parameters for the PLC. Choose the appropriate IP address for your PLC and click **OK**. Test the connection by clicking **Connection Test** and click **OK** after confirming that the connection was successful.

Now that we have established succesful connection to the PLC we may proceed with uploading the developed control program. Use the menu bar to open the programming windows by clicking **Online->Write to PLC**. Choose the MAIN program file as the only file to upload to the PLC and click **Execute**.



2.4.4.3 AR application

In order to test the uploaded program the student may interact with the process through the AR application by using the interface buttons shown in Figure 2-6. Figure 2-11 illustrates the interface of the AR application.

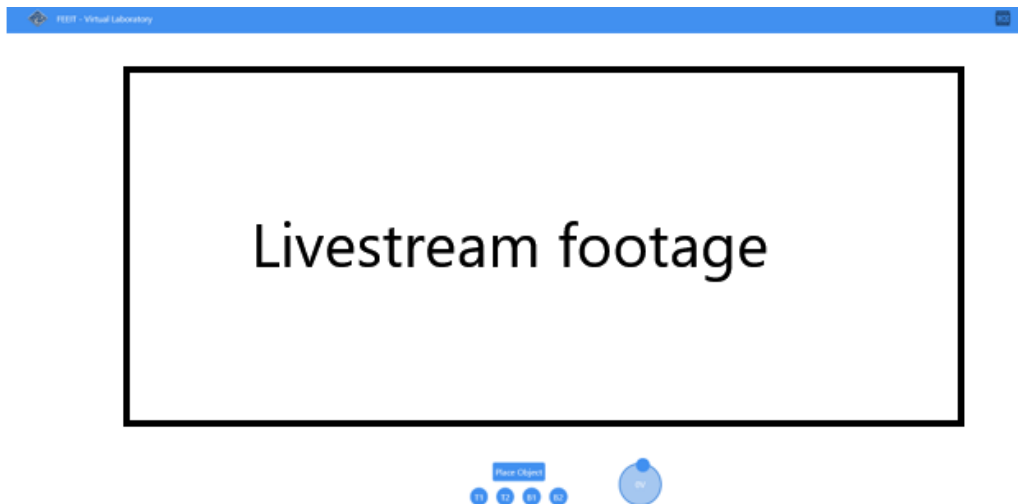


Figure 2-11. AR application interface.

2.4.5 Experimental results

Use the AR application in order to test the program workflow.

- Use the **Place object** to place a virtual object on the conveyor belt. A green square should appear on the conveyor belt.
- Click the **T1** to start the process. If the program upload was successful, the virtual green object should start moving. Observe the triggering of each sensor, as the object passes in front of them.
- Each sensor designates its state by a green or red square right next to it, as shown in Figure 2-7. If the photoelectric sensor is successfully triggered, the conveyor belt should stop.
- Use the **Place object** button to remove the object. The conveyor belt should start moving again

Figure 2-12 shows a series of images that present the workflow of the AR app.

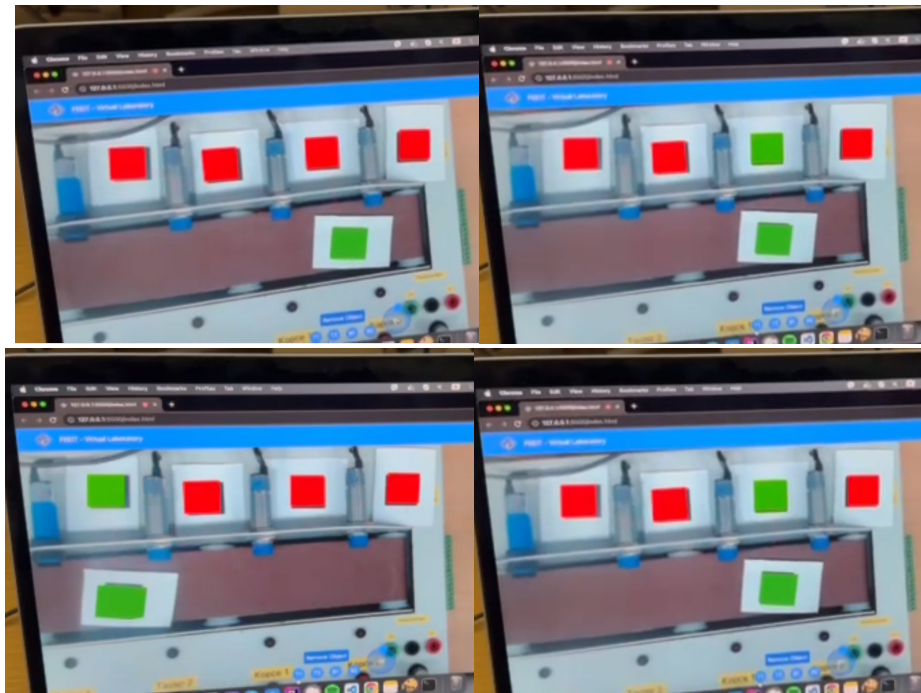


Figure 2-12. AR application workflow.

2.5 Conclusion

A remote access PLC laboratory was presented utilizing AR technology. It was shown that necessary interaction for evaluation PLC programs can be executed remotely through AR virtualized objects and sensors. Furthermore, this was demonstrated through a simple laboratory experiment. A control program was developed and its operation was tested through an AR application enabling remote access to the laboratory.

2.6 References

- [1] Mitsubishi Electric, Product Information, FX3GE MELSEC Compact PLC, All-in-one-standard including analog and network features, EBG 243 EN
- [2] Mitsubishi Electric, Programmable Controller MELSEC-F, FX3U-ENET User's Manual
- [3] Mitsubishi Electric, Programmable Controller MELSEC-F, FX3GE Series Programmable Controller, Hardware Manual, JY997D49401J
- [4] Mitsubishi Electric, Engineering Software, GX Works2 Beginner's Manual (Simple Project)
- [5] Mitsubishi Electric, Programmable Controllers MELSEC-F, Programming Manual, The FX Series of Programmable Controller

2.7 Acknowledgements

- <https://www.flaticon.com/free-icons/plctitle> - created by Freepik – Flaticon
- <https://www.flaticon.com/free-icons/network-switch> - created by Chattapat - Flaticon
- <https://www.flaticon.com/free-icons/server> - created by Freepik – Flaticon
- <https://www.flaticon.com/free-icons/router> - created by Vectors Tank – Flaticon
- <https://www.flaticon.com/free-icons/server> - created by Freepik – Flaticon
- <https://www.flaticon.com/free-icons/web-camera> - created by Freepik – Flaticon
- <https://www.flaticon.com/free-icons/web-pagetitle> - created by Talha Dogar – Flaticon

Laboratory manual

Scilab Virtual Software Laboratory

UbiLAB



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



3 Scilab Virtual Software Laboratory

3.1 Introduction

The Virtual Software Laboratories (VSL) developed via the UbiLAB platform, can be used and implemented in a vast variety of online courses and applications. VSLs are implemented in the UbiLAB framework through the Moodle LMS (however, everything described below can be easily implemented on other Learning Management Systems, such as Open edX).

In order to define and organize a VSL in the UbiLAB framework several elements need to be incorporated and connected. First, the access to the VSL needs to be configured. This is realized by using the Apache Guacamole framework and integrating it via the iFrame plugin in the LMS. Namely, Apache Guacamole is an open-source remote desktop gateway that allows users to access their desktops or applications through a web browser. It provides a way for users to access their desktops and applications from anywhere with an internet connection. The Guacamole server is installed on a server machine and configured to allow connections to remote desktops or applications. It uses standard protocols like VNC and RDP to communicate with the remote desktop, and it provides a web interface that allows users to access their remote desktops or applications from any device with a web browser.

The Guacamole server can be configured to enable access to different software environments (Linux and Windows-based virtual machines, Docker or LXC containers and similar) equipped with the needed software applications for the online course, in our case Scilab. These environments actually present the resources in the LMS that the student needs to reserve. The reservation process is done through the Scheduling module implemented into the LMS, described later.

When the Guacamole server is configured, the appropriate web interface can be embedded within an LMS iFrame. The iFrame is a web feature that can be easily embedded in different LMS platforms (Moodle). It allows embedding external web pages or applications (in this case the Guacamole web interface) within the LMS course page. An iFrame (short for inline frame) is a HTML element that can display the contents of another web page within a frame on your own/current web page. By embedding the Guacamole web interface within an iFrame, the students can access the remote virtual laboratory software environments or applications without leaving the LMS platform.

When the resources in the course have been configured, in such a way that one virtual machine (virtual software environment) is one resource that can be used by one (or several, depending on the configuration/logic of the exercise) students. Then by using the implemented Scheduler module from the LMS (Moodle), teachers can organize the laboratory exercises for the entire

semester, create suitable laboratory slots (even for the entire semester), define laboratory exercise durations, set the maximum number of participants per exercise, and specify which students or groups can sign up for each exercise. The scheduler is connected to the actually definer resource in the course as explained previously. Then, students can sign up for available exercise slots directly from within the course, and teachers can manage and view their exercises through the Scheduler interface. Students are allowed access to the resource (virtual software environment) just during the time slot he/she reserved.

3.2 Scilab

Scilab is a free, open-source numerical computation software package for scientific and engineering applications. It provides a powerful computing environment for solving complex mathematical problems, creating and manipulating graphs and visualizations, and performing data analysis.

Scilab is similar in functionality to other popular numerical computing environments such as MATLAB, GNU Octave, and Python's NumPy library. It includes a large library of built-in mathematical functions, as well as tools for linear algebra, signal processing, optimization, statistics, and more.

Scilab also supports the development of user-defined functions and modules, making it highly customizable and extensible. It can be used for a wide range of applications, including scientific research, engineering design, and education.

Overall, Scilab is a powerful and flexible software package that provides a comprehensive set of tools for numerical computing and scientific analysis.

3.3 Laboratory experiment 1: AM Modulation

3.3.1 Goals of the experiment

The objective of this laboratory exercise is to provide a practical understanding of amplitude modulation (AM) using Xcos, a graphical modeling and simulation tool in Scilab. Amplitude modulation is a widely used technique in communication systems for transmitting information by varying the amplitude of a carrier wave. In this lab, we will use Xcos to simulate an AM system, explore the key components, and analyze the effects of different parameters on the modulation process.

3.3.2 Experimental setup

- Launch Scilab on the virtual machine and open the Xcos environment.
- Familiarize yourself with the Xcos interface, which consists of a palette of blocks, a model editor area, and a simulation toolbar.
- Locate the blocks specific to AM modulation in the Xcos palette:
 - GENSIN_f: sine wave generator;
 - CONST_m: constant value generator.
 - PRODUCT: element-wise multiplication of its vector inputs
 - BIGSOM_f: performs addition on its scalar or vector inputs
 - CSCCOPE: displays its input with respect to simulation time
 - CLOCK_s: generates a regular train of events that are scheduled by parameter

3.3.3 Experimental procedure

1. Start by creating a new Xcos model. Drag and drop the necessary blocks from the palette onto the model editor area to build the AM modulation system (as shown in Figure 3-1). Connect the blocks to establish the signal flow.

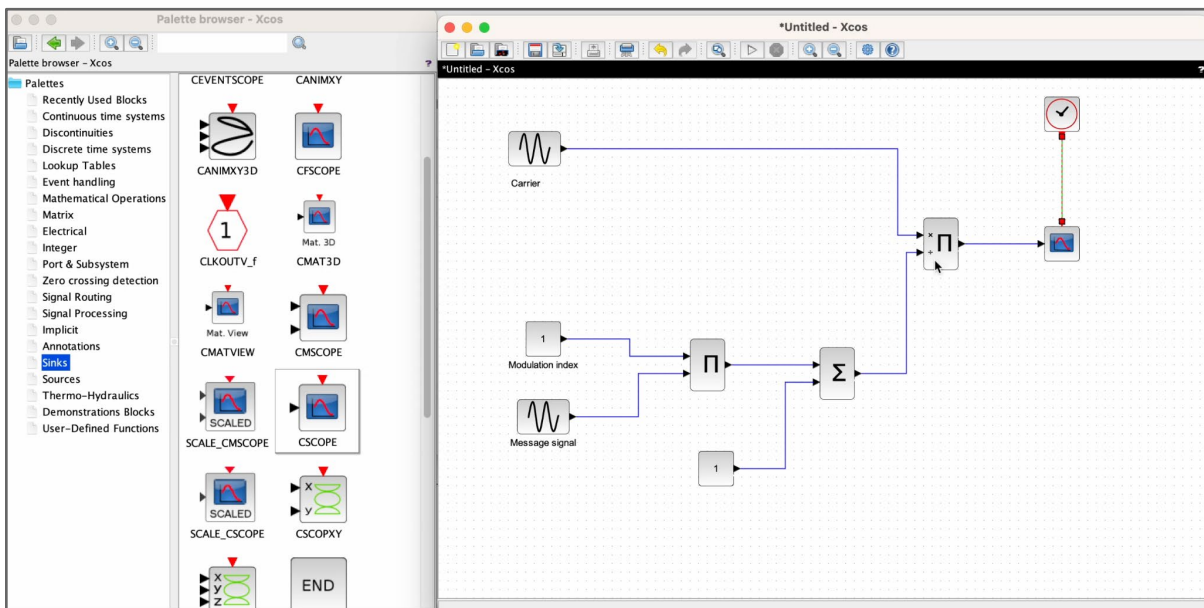


Figure 3-1. Scheme of the AM modulation system.

2. Set the frequency and amplitude of the carrier signal and the message signal (sine wave generators), as shown in Figure 3-2 and Figure 3-3.

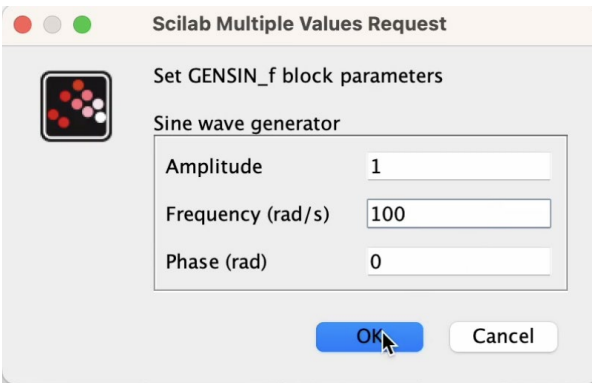


Figure 3-2. Carrier signal.

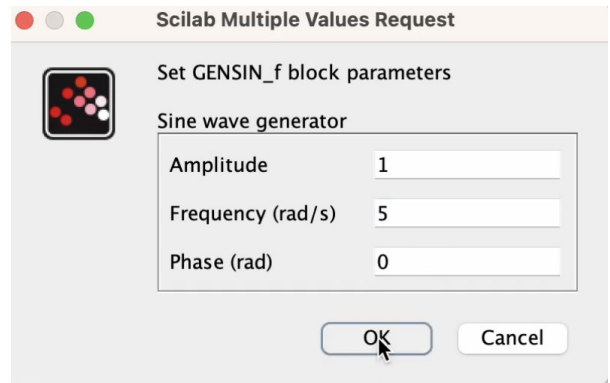


Figure 3-3. Modulation signal.

3. Run the simulation and observe the modulated signal on the Scope block. Pay attention to the changes in amplitude and frequency as a result of the modulation process.
4. Vary the parameters of the modulating signal, such as frequency and amplitude, to observe their effects on the modulated signal. Note any changes in amplitude modulation depth or spectral content. Additionally, vary the modulation index.

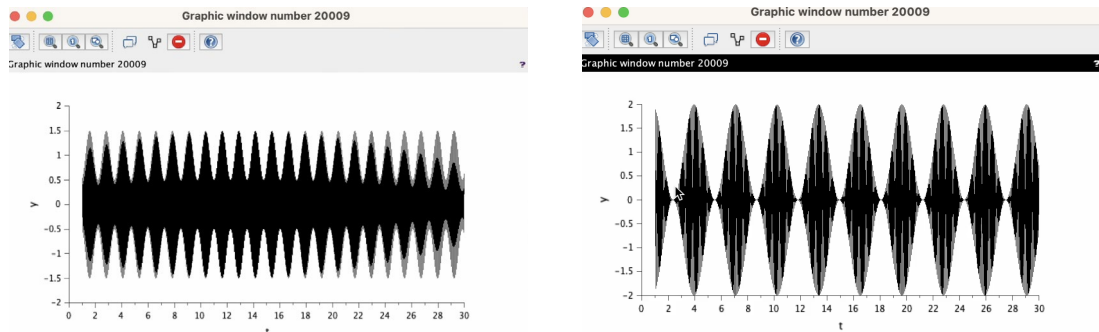


Figure 3-4. Modulated signal for different system parameters.

5. Experiment with different carrier frequencies while keeping the modulating signal constant. Observe how the carrier frequency affects the characteristics of the modulated signal (Figure 3-4).
6. Use Xcos's simulation features to analyze the demodulation process. Introduce an AM Demodulator block and connect it to the modulated signal. Connect the output of the demodulator to another Scope block to visualize the demodulated signal.

3.3.4 Experimental results

This laboratory exercise provided a practical understanding of amplitude modulation using Xcos in Scilab. By building and simulating an AM system, you have observed the effects of different parameters on the modulation and demodulation processes. Through Xcos's graphical modeling capabilities, you have gained insight into the modulation depth, carrier frequency, and spectral content of the modulated signal. This knowledge will serve as a foundation for further exploration of modulation techniques and their applications in communication systems.

3.4 Laboratory experiment 2: AM Demodulation

3.4.1 Goals of the experiment

The objective of this laboratory exercise is to provide a practical understanding of amplitude demodulation using Xcos, a graphical modeling and simulation tool in Scilab. Amplitude demodulation is the process of extracting the original modulating signal from an amplitude-modulated (AM) carrier signal. In this lab, we will use Xcos to simulate an AM demodulation system, explore the key components, and analyze the effects of different parameters on the demodulation process.

3.4.2 Experimental setup

- Launch Scilab on the virtual machine and open the Xcos environment.
- Familiarize yourself with the Xcos interface, which consists of a palette of blocks, a model editor area, and a simulation toolbar.
- Locate the blocks specific to AM modulation in the Xcos palette:
 - GENSIN_f: sine wave generator;
 - CONST_m: constant value generator.
 - PRODUCT: element-wise multiplication of its vector inputs
 - BIGSOM_f: performs addition on its scalar or vector inputs
 - CSCCOPE: displays its input with respect to simulation time
 - CLOCK_s: generates a regular train of events that are scheduled by parameter
 - CLR: SISO linear system represented by its rational transfer function Numerator/Denominator

3.4.3 Experimental procedure

1. Start by creating a new Xcos model. Drag and drop the necessary blocks from the palette onto the model editor area to build the AM demodulation system (as shown in Figure 3-5). Connect the blocks to establish the signal flow.

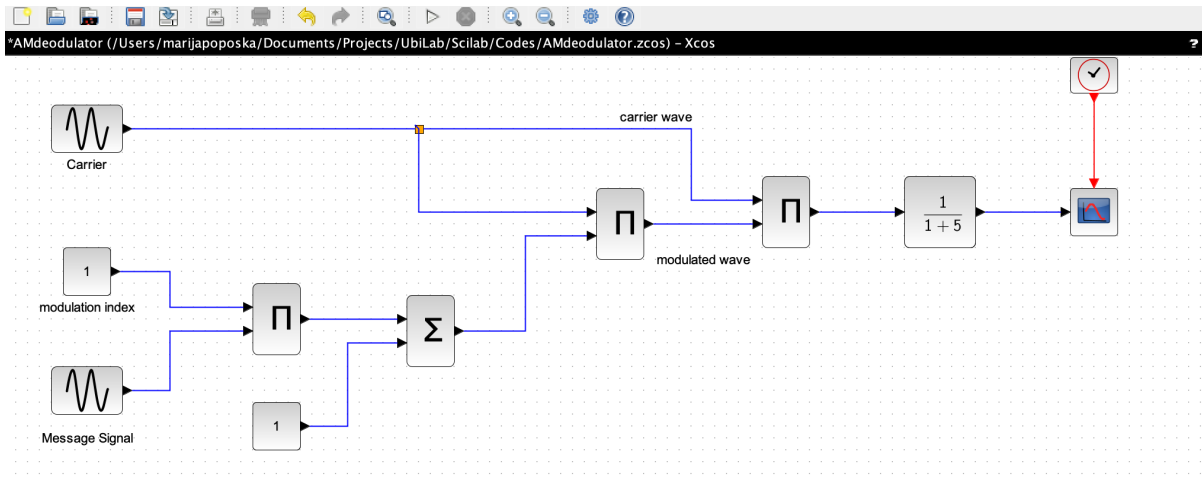


Figure 3-5. Scheme of the AM demodulation system.

2. Set the frequency and amplitude of the carrier signal and the message signal (sine wave generators), as shown in Figure 3-6 and Figure 3-7.

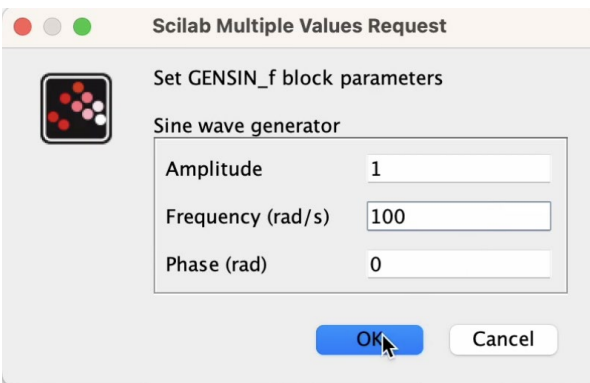


Figure 3-6. Carrier signal.

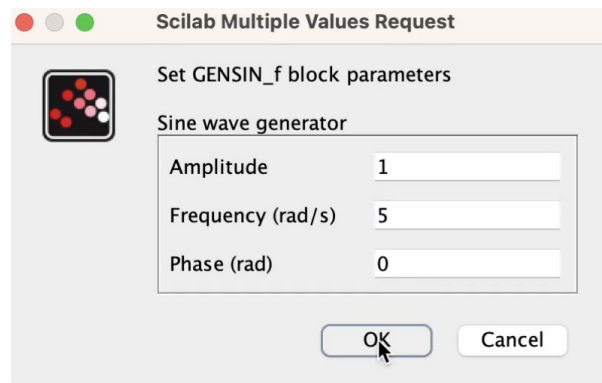


Figure 3-7. Modulation signal.

3. Set the values of the low-pass filter (Figure 3-8).

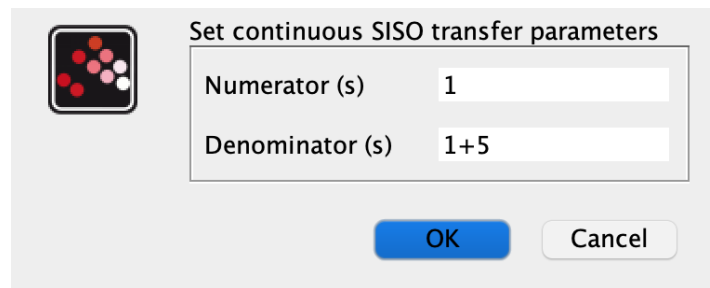


Figure 3-8. Low-pass filter.

4. Run the simulation and observe the demodulated signal on the Scope block. Pay attention to the changes in amplitude and frequency as a result of the demodulation process.
5. Vary the parameters of the modulating signal, such as frequency and amplitude, to observe their effects on the demodulated signal. Note any changes in amplitude modulation depth or spectral content. Additionally, vary the values of the low-pass filter.

3.4.4 Experimental results

This laboratory exercise provided a practical exploration of amplitude demodulation using Xcos in Scilab. By building and simulating an AM demodulation system, you have observed the effects of different parameters and techniques on the fidelity and accuracy of the demodulated signal. Through Xcos's graphical modeling capabilities, you have gained insight into envelope detection, synchronous demodulation, and the impact of noise on the demodulation process. This knowledge will serve as a foundation for further exploration of demodulation techniques and their applications in communication systems.

3.5 Laboratory experiment 3: Digital Modulation Techniques

3.5.1 Goals of the experiment

The objective of this laboratory exercise is to provide a hands-on understanding of digital modulation techniques, specifically Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Phase Shift Keying (PSK). These techniques are widely used in digital communication systems to transmit digital data over a carrier signal by manipulating amplitude, frequency, or phase. In this lab, we will use Scilab to simulate and analyze ASK, FSK, and PSK systems, and investigate their characteristics and performance.

3.5.2 Experimental setup

- Launch Scilab on your computer and create a new script or open the Scilab editor.
- Familiarize yourself with the Scilab environment and basic programming concepts, such as variables, functions, and plotting capabilities.

3.5.3 Experimental procedure

1. Start by defining the necessary parameters for ASK, FSK, and PSK systems, such as the carrier frequency, symbol rate, modulation index, and phase offsets. Assign appropriate values to these parameters based on the desired characteristics of each modulation technique.
2. Generate a digital signal, such as a binary sequence, to be modulated using ASK, FSK, and PSK techniques. Assign appropriate values to represent the digital data.

```
clc;
clear;
xdel(winsid());
sym=10;//no. of symbols
g=[1 1 1 0 1 0 0 1 0 1 ]//binary data
f1=1;f2=2;//frequencies of carrier
t=0:2*%pi/99:2*%pi;//range of time
```

3. Implement the ASK modulation scheme by creating a function or series of Scilab commands to encode a digital signal onto an amplitude-modulated carrier wave. This can involve multiplying the digital signal with the carrier signal to achieve the desired modulation.

```
cp=[];bit=[];mod_ask=[];mod_fsk=[];mod_psk=[];cp1=[];cp2=[];
for n=1:length(g);
    if g(n)==0;
        die=zeros(1,100);
    else g(n)==1;
        die=ones(1,100);
    end
end
```

```

end
c_ask=sin(f1*t);
cp=[cp die];
mod_ask=[mod_ask c_ask];
end
ask=cp.*mod_ask;//ASK modulated signal

```

4. Implement the FSK modulation scheme by creating a function or series of Scilab commands to encode the digital signal onto frequency-modulated carrier waves. This can involve generating two carrier signals at different frequencies and switching between them based on the digital signal.

```

for n=1:length(g);
    if g(n)==0;
        die=ones(1,100);
        c_fsk=sin(f1*t);
    else g(n)==1;
        die=ones(1,100);
        c_fsk=sin(f2*t);
    end
    cp1=[cp1 die];
    mod_fsk=[mod_fsk c_fsk];
end
fsk=cp1.*mod_fsk;//FSK modulated signal

```

5. Implement the PSK modulation scheme by creating a function or series of Scilab commands to encode the digital signal onto phase-modulated carrier waves. This can involve adjusting the phase of the carrier signal according to the digital signal.

```

for n=1:length(g);
    if g(n)==0;

```



```

        die=ones(1,100);
        c_psk=sin(f1*t);
    else g(n)==1;
        die=ones(1,100);
        c_psk=-sin(f1*t);
    end
    cp2=[cp2 die];
    mod_psk=[mod_psk c_psk];
end
psk=cp2.*mod_psk;//PSK modulated signal

```

6. Plot and analyze the modulated signals in the time domain to observe the effects of each modulation technique on the carrier wave.

```

subplot(4,1,1);plot(cp,'LineWidth',1.5);//plot binary signal
xgrid;
title('Binary Signal');//title
mtlb_axis([0 100*length(g) -2.5 2.5]);//axis range
subplot(4,1,2);plot(ask,'LineWidth',1.5);//plot of ASK modulated signal
xgrid;
title('ASK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//axis range
subplot(4,1,3);plot(fsk,'LineWidth',1.5);//plot of FSK modulated signal
xgrid;
title('FSK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//axis range
subplot(4,1,4);plot(psk,'LineWidth',1.5);//plot of PSK modulated signal
xgrid;
title('PSK modulation');//title of plot
mtlb_axis([0 100*length(g) -2.5 2.5]);//range of axis

```

3.5.4 Experimental results

This laboratory exercise provided a practical exploration of digital modulation techniques, specifically ASK, FSK, and PSK, using Scilab. By implementing and simulating these techniques, you have gained insight into how they manipulate carrier signals to transmit digital data. Through analyzing the modulated signals and demodulating them to obtain the original digital data, you have evaluated the performance of each modulation technique (Figure 3-9). This knowledge will serve as a foundation for further exploration of digital communication systems and modulation schemes.

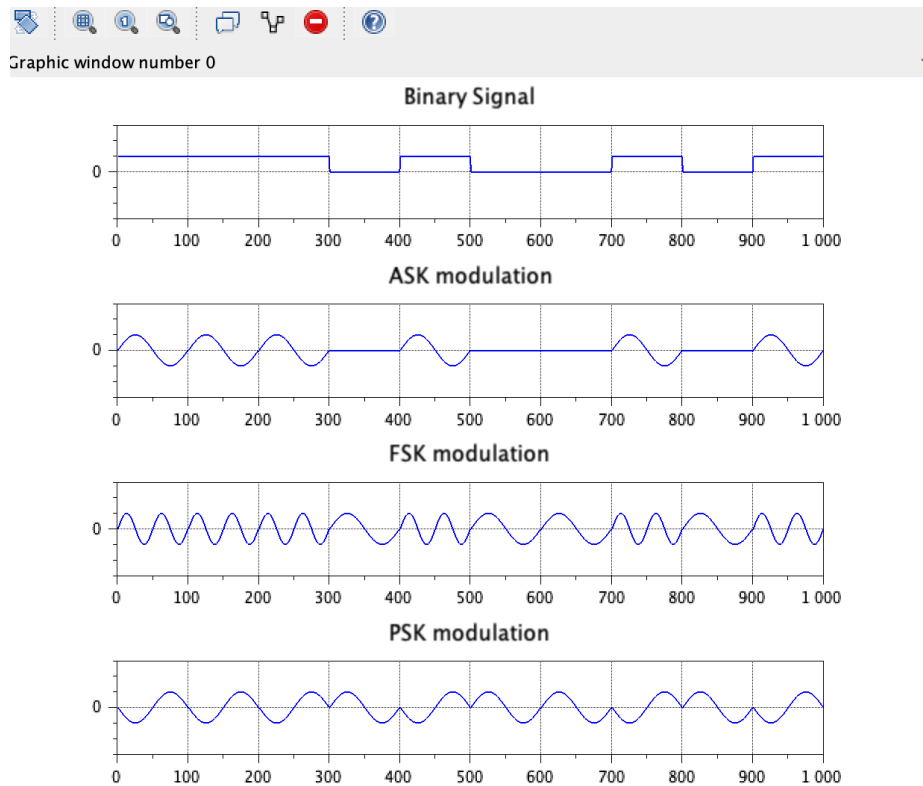


Figure 3-9. Results of the digital modulation techniques.

3.6 Conclusion

In conclusion, the Scilab experiment on analog and digital modulation provided valuable insights into the fundamental concepts and techniques used in communication systems. Through hands-on simulations, we explored both analog modulation methods, such as Amplitude Modulation (AM), as well as digital modulation techniques, including Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Phase Shift Keying (PSK).

Amplitude modulation allowed us to understand the relationship between the modulating signal and the carrier. On the other hand, digital modulation provided us with insights into how digital data is conveyed using discrete symbols. ASK, FSK, and PSK modulation schemes showcased diverse approaches to encoding digital information onto carrier waves, each with its unique strengths and applications.

By leveraging the capabilities of Scilab, we were able to visualize, analyze, and demodulate complex signals, enabling a deeper understanding of the performance and characteristics of each modulation technique.

Laboratory manual

Digital Signal Processing Laboratory

UbiLAB



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



Erasmus+



NATIONAL AGENCY
for European Educational
Programmes and Mobility

4 Digital Signal Processing Laboratory

4.1 Introduction

The Digital Signal Processing (DSP) course provides students with interactive remote-controlled experiments within the DSP laboratory. Three NI myRIO 1900 devices are programmed to host different experiments each. More about myRIO 1900 can be read in its official manual provided by National Instruments:

<https://www.ni.com/docs/en-US/bundle/myrio-1900-getting-started/resource/376047d.pdf>

The experiments are based on the exploitation of the Signal Processing Toolkit applied for:

- Real-Time Audio Filtering;
- Real-Time Moving Spectrogram;
- Real-Time PWM Generation.

They have a friendly user interface (UI), from which the remote user can change predefined parameters and observe the response in numerical, graphical and audio format. The LabVIEW software is used both for the experiment's development and for the user front-end and remote control. It should be noted that students do not need any LabVIEW programming knowledge, they just use the UI. However, for enthusiasts who have a basic knowledge of LabVIEW, there is an opportunity for accessing the LabVIEW projects for exploring additional signal processing tools.

The remote performance of the laboratory experiments offers the same possibilities as if the experiments are performed physically in the laboratory. The students need to be remotely connected through the Apache Guacamole Remote Desktop interface, to each workstation (PC computer) to access the UI and myRIO device. Additionally, connection through Zoom will be needed if the students want to process audio signals streamed from their microphone. To do this, the equipment needed from the student's side are a microphone for real-time audio streaming and headphones for hearing the result from the audio processing.

Note: For doing these experiments, students are expected to have a basic knowledge of STFT, spectrograms, IIR filters, and PWM signals. A brief reminder about these topics is given at the beginning of each of the experiments.

4.2 Real-Time Audio Filtering

IIR stands for Infinite Impulse Response, which refers to a type of digital filter commonly used for audio filtering. IIR filters are characterized by feedback, which allows for more efficient implementation compared to FIR (Finite Impulse Response) filters.

In audio filtering applications, IIR filters are used to shape the frequency response of a signal by attenuating or amplifying specific frequency components. They can be useful for tasks such as equalization, low-pass filtering, high-pass filtering, band-pass filtering, and more.

IIR filters are defined by their transfer function, which describes the relationship between the input and output of the filter in the frequency domain. The transfer function typically takes the form of a ratio of polynomials in the z-domain, where z represents the complex variable of the frequency response.

The two main types of IIR filters commonly used in audio processing are:

1. **Butterworth Filters:** Butterworth filters have a maximally flat frequency response in the passband and a gradual roll-off in the stopband. They are commonly used for applications where a smooth frequency response is desired, such as audio equalization. Butterworth filters can be designed as low-pass, high-pass, band-pass, or band-stop filters.
2. **Chebyshev Filters:** Chebyshev filters trade off frequency response flatness for steeper roll-off. They can achieve steeper roll-off rates than Butterworth filters but with some ripple in the passband or stopband. Chebyshev filters are suitable when a more aggressive filtering is needed.

Designing and implementing IIR filters typically involves filter specification: Definition of the desired frequency response characteristics, such as cutoff frequency, passband ripple, stopband attenuation, and filter order.

It's important to note that designing and implementing IIR filters requires careful consideration of stability to avoid issues like filter instability or excessive ringing. Additionally, appropriate filter design and implementation techniques should be employed to meet the desired audio filtering requirements effectively.

4.2.1 Experimental setup

The setup for this experiment, from the laboratory side, is consisted of:

- NI myRIO 1900 device connected to a PC within the DSP laboratory;
- 3.5 mm audio jack male to male cable connected from the PC's audio output to myRIO's audio input;
- 3.5 mm audio jack male to male cable connected from myRIO's audio output to the PC's audio input;
- LabVIEW user interface.

The setup needed from the student's side is:

- Remote Desktop application;
- Zoom application (optional);
- Microphone (optional);
- Headphones (optional).

The audio signal can be streamed from the PC, for example from YouTube, or from a microphone. If a microphone is used as an audio source, then a Zoom meeting needs to be launched.

The user interface is shown in Figure 4-1. It provides a real-time plot of the raw audio waveform and a real-time raw vs. filtered audio spectrum. Students can change the predefined parameters about IIR filter specifications and the volume of the input audio signal.

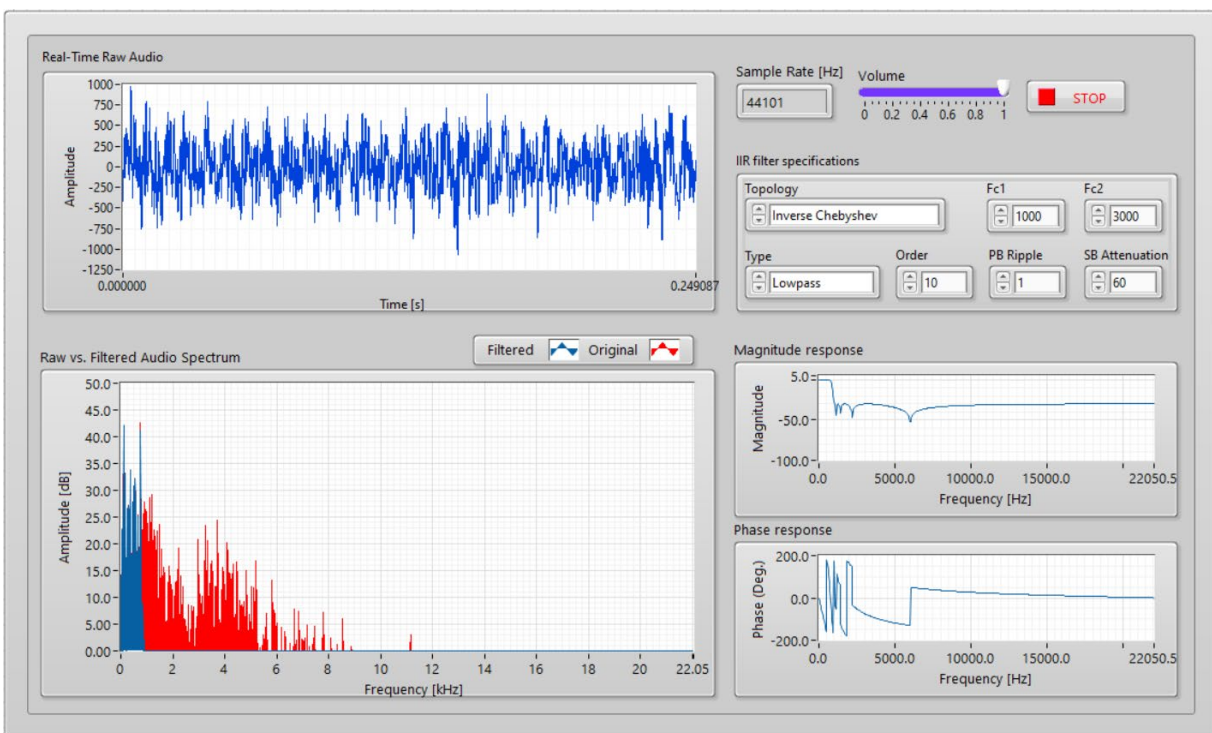


Figure 4-1. User interface for real-time audio filtering.

The offered filter specifications are the following:

- Filter topology:
 - Butterworth;
 - Chebyshev;
 - Inverse Chebyshev;
 - Elliptic;
 - Bessel;
- Filter type:
 - High-pass;

- Low-pass;
- Band-pass;
- Band-stop;
- First cut-off frequency – F_{c1} ;
- Second cut-off frequency – F_{c2} ;
- Filter order;
- Pass band ripple;
- Stop band attenuation.

Note: When either low-pass or high-pass filter is used, the cut-off frequency is F_{c1} .

4.2.2 Goal of the experiment

The aim of this experiment is to perform real-time IIR filtering on the input audio signal and to observe the change in its spectrum according to the filter responses, as well as to listen to the filtered audio result.

4.2.3 Experimental results

Let's separate the input audio signal into different frequency ranges, called Bass, Midtone and Treble.

Bass is the lowest range of frequencies in an audio signal. To extract them we use a low-pass filter, for example a Butterworth filter, with a cut-off frequency $f_c = 450Hz$ and 3rd order. The result is shown in Figure 4-2. The spectrum of the raw audio signal is given in red, while the spectrum of the filtered audio signals is given in blue.

Midtone is the middle range of frequencies in an audio signal. Here we use a band-pass Butterworth filter, with cut-off frequencies $f_{c1} = 450Hz$ and $f_{c2} = 3000Hz$ and from 3rd order. The result is shown in Figure 4-3.

Treble is the highest range of frequencies in an audio signal. For that aim, we use a high-pass Butterworth filter with cut-off frequency $f_c = 3000Hz$ and from 3rd order. The result is shown in Figure 4-4.

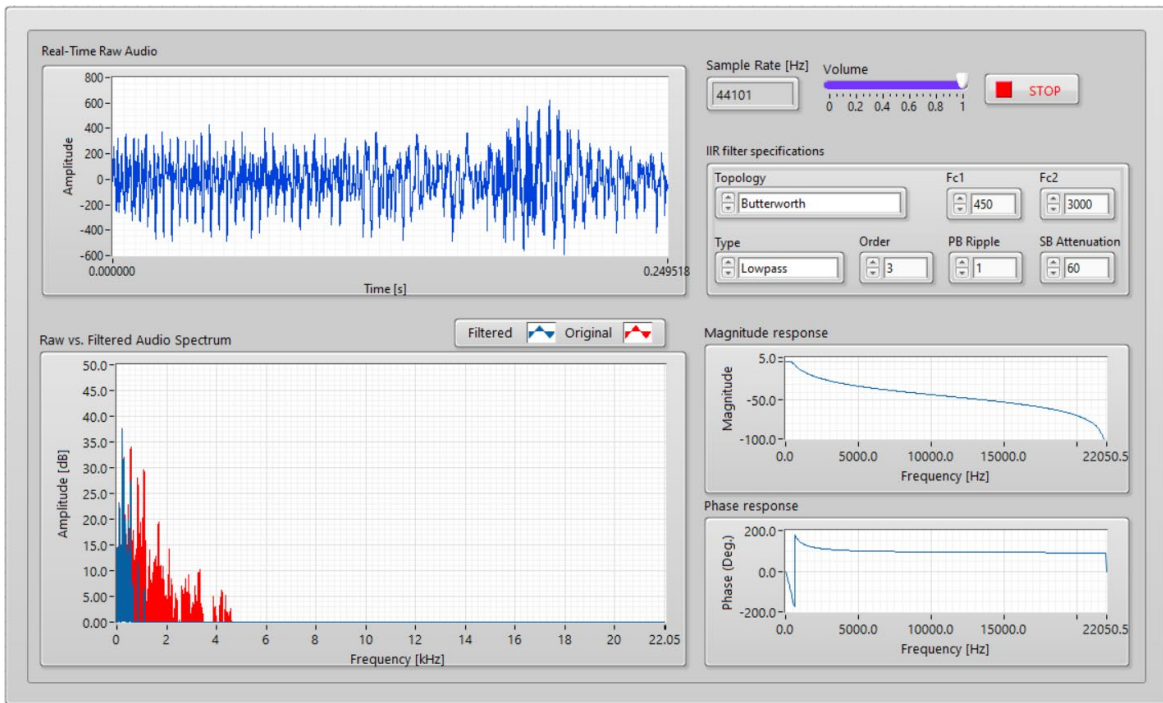


Figure 4-2 Bass frequency range from an input audio signal.

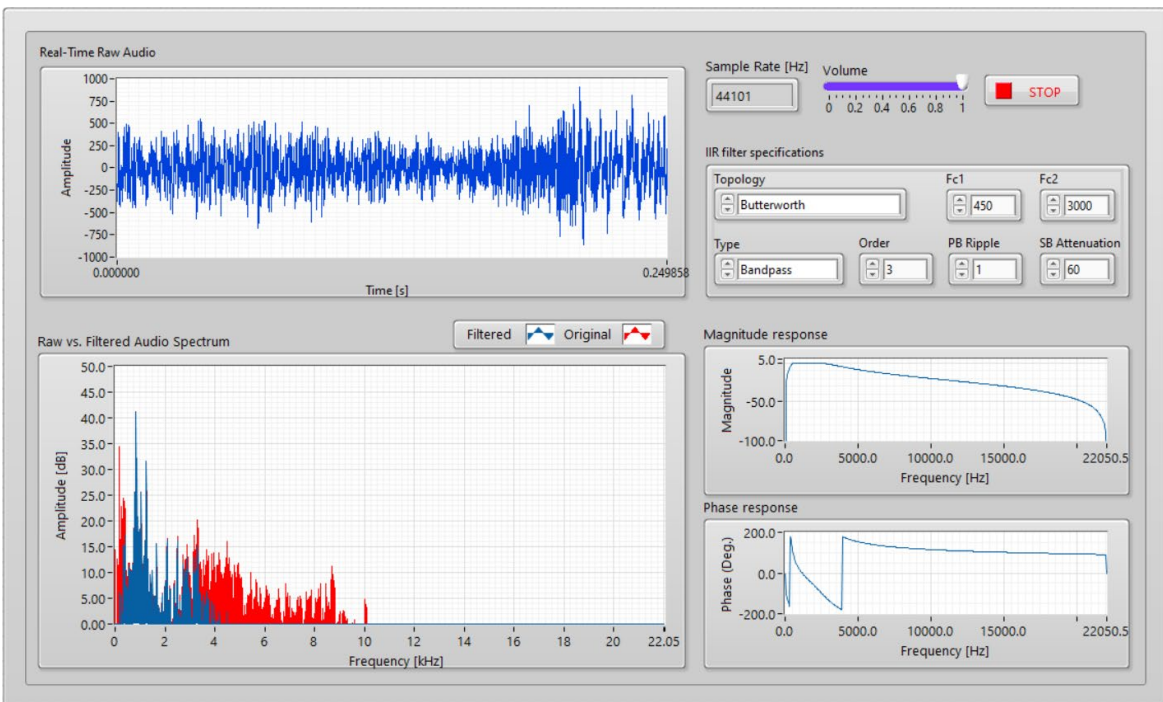


Figure 4-3. Midtone frequency range from an input audio signal.

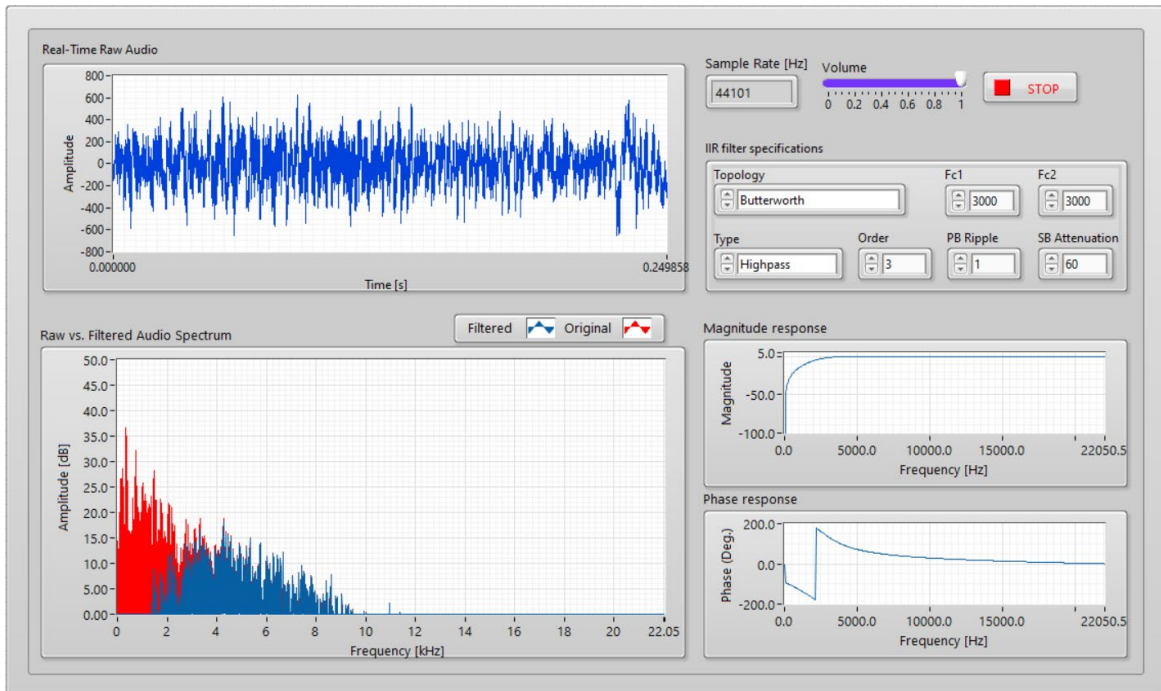


Figure 4-4. Treble frequency ranges from an input audio signal.

Task 1: Make a comparison between the spectrums and the responses of the filters.

Task 2: Try different filter specifications, observe and listen the result.

4.3 Real-Time Moving Spectrogram

STFT, which stands for Short-Time Fourier Transform, is a widely used signal processing technique for analyzing and visualizing the frequency components of a time-varying signal.

The STFT breaks down a signal into a series of short overlapping windows and performs a Fourier transform on each window to obtain its frequency content. This allows us to observe how the frequency content of the signal changes over time.

Here's how the STFT spectrogram is computed:

1. The input signal is divided into small segments or windows. The choice of window length and overlap between windows depends on the specific application and desired time-frequency resolution.
2. Each windowed segment of the signal is multiplied with a window function (e.g., Hamming, Hanning, Blackman ...) to reduce spectral leakage and improve frequency resolution.
3. A Fourier transform (typically the Fast Fourier Transform, or FFT) is applied to each windowed segment, producing a frequency spectrum for that specific time interval.

4. The resulting spectra are stacked together to form a two-dimensional representation called the spectrogram. The x-axis represents time, and the y-axis represents frequency. The color intensity or grayscale value at each time-frequency point corresponds to the magnitude or power of the frequency component present in that segment.

By analyzing the spectrogram, you can identify the dominant frequencies at different points in time and observe how they evolve. Spectrograms are commonly used in various fields, including audio processing, speech analysis, music analysis, vibration analysis, and more.

It's worth noting that the STFT spectrogram is a time-frequency representation, providing valuable information about the spectral characteristics of a signal over time. However, it has limitations in terms of temporal and frequency resolution, and some details may be lost due to windowing and overlap. Different choices of window size, type and overlap can trade-off between time and frequency resolution to suit specific analysis requirements.

4.3.1 Experimental setup

The setup for this experiment, from the laboratory side, consists of:

- NI myRIO 1900 device connected to a PC within the DSP laboratory;
- 3.5 mm audio jack male to male cable connected from the PC's audio output to myRIO's audio input;
- LabView user interface.

The setup needed from the student's side is:

- Remote Desktop application;
- Zoom application (optional);
- Microphone (optional);
- Headphones (optional).

The audio signal can be streamed from the PC, for example from YouTube, or from a microphone. If a microphone is used as an audio source, then a Zoom meeting needs to be launched. In the following example the audio file sound1.wav is used, attached as a source file to this experiment.

The user interface is shown in Figure 4-5. It provides a real-time raw audio graph and a real-time moving spectrogram. Student can change the predefined parameters of the window type, and window length.

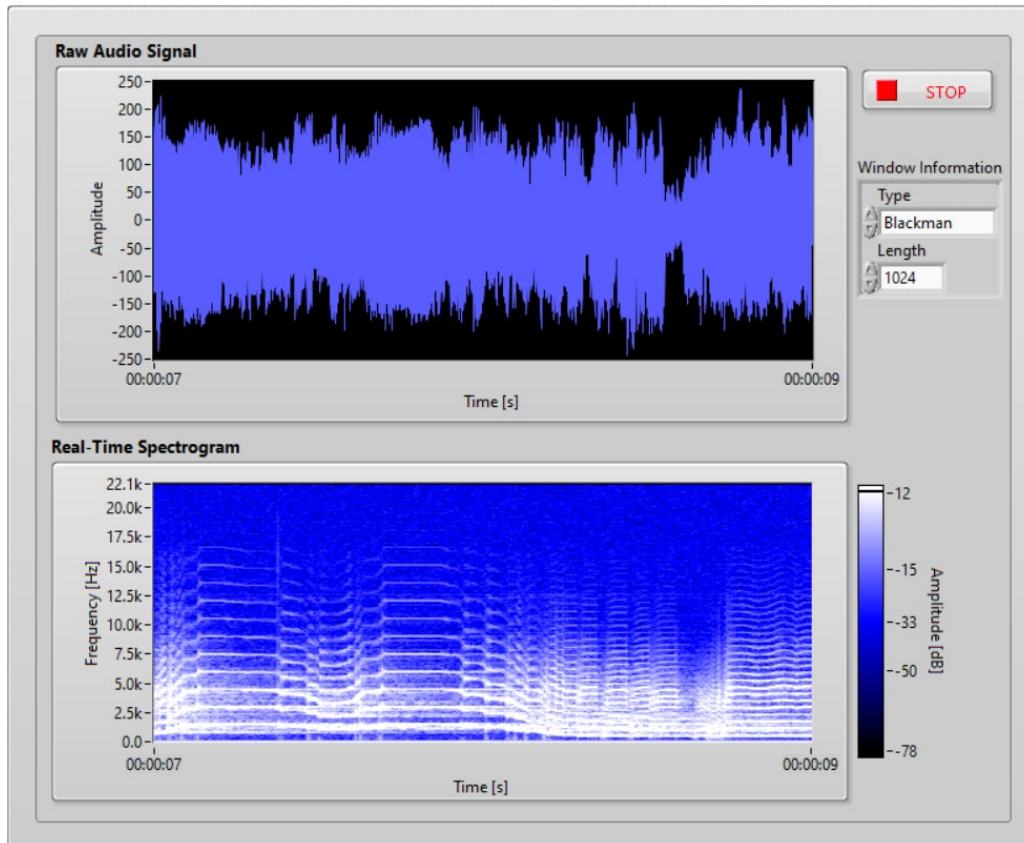


Figure 4-5. User interface for real-time moving spectrogram.

The offered window types are the following:

- Hanning;
- Hamming;
- Blackman – Harris;
- Exact Blackman;
- Blackman;
- Flat Top.

4.3.2 Goals of the experiment

The goal of this experiment is to observe the impact of the different window type and different window length on the quality of the audio spectrogram.

4.3.3 Experimental results

Choose the window type to be “Hamming”. Let’s try two different window lengths and observe the obtained results.

4.3.3.1 Narrow window

In the first case we can use a narrow window with a length equal of 64. The result is shown in Figure 4-6. It can be seen that this window gives good time localization but very poor frequency resolution.

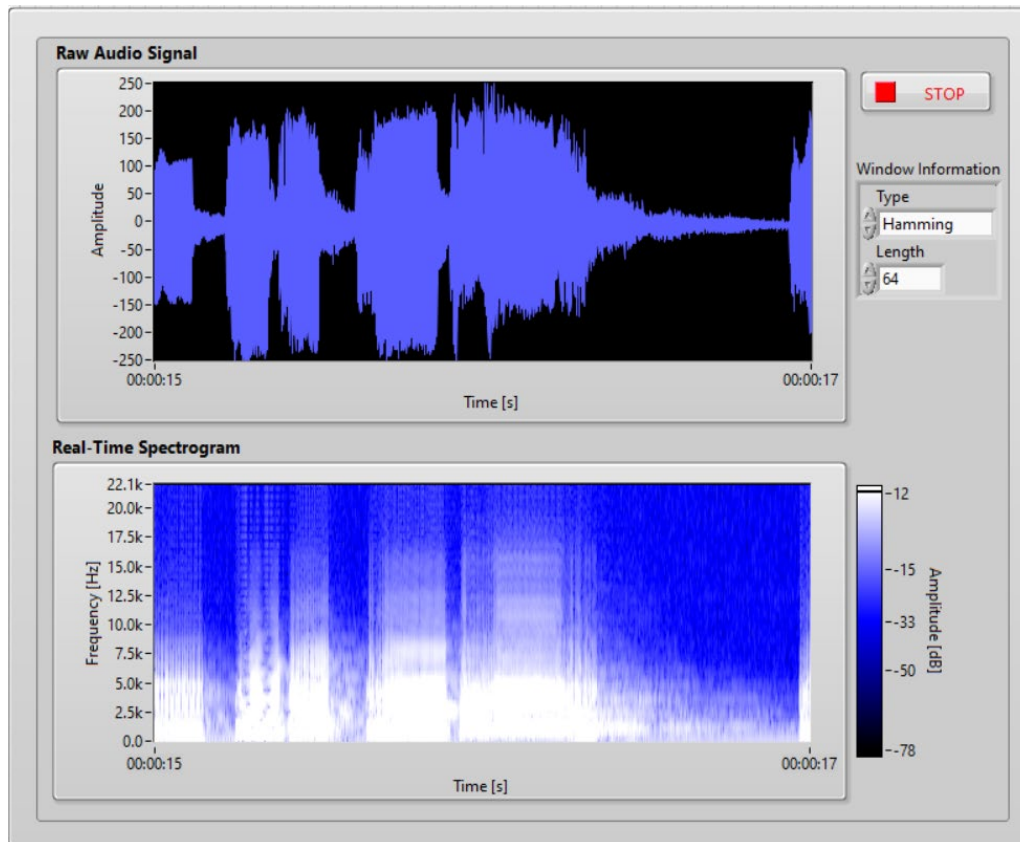


Figure 4-6. Narrow window.

4.3.3.2 Wide window

In the second case we will set the length of the window to 512. From Figure 4-7, we can conclude that this window gives better frequency resolution, but now the time localization is worse.

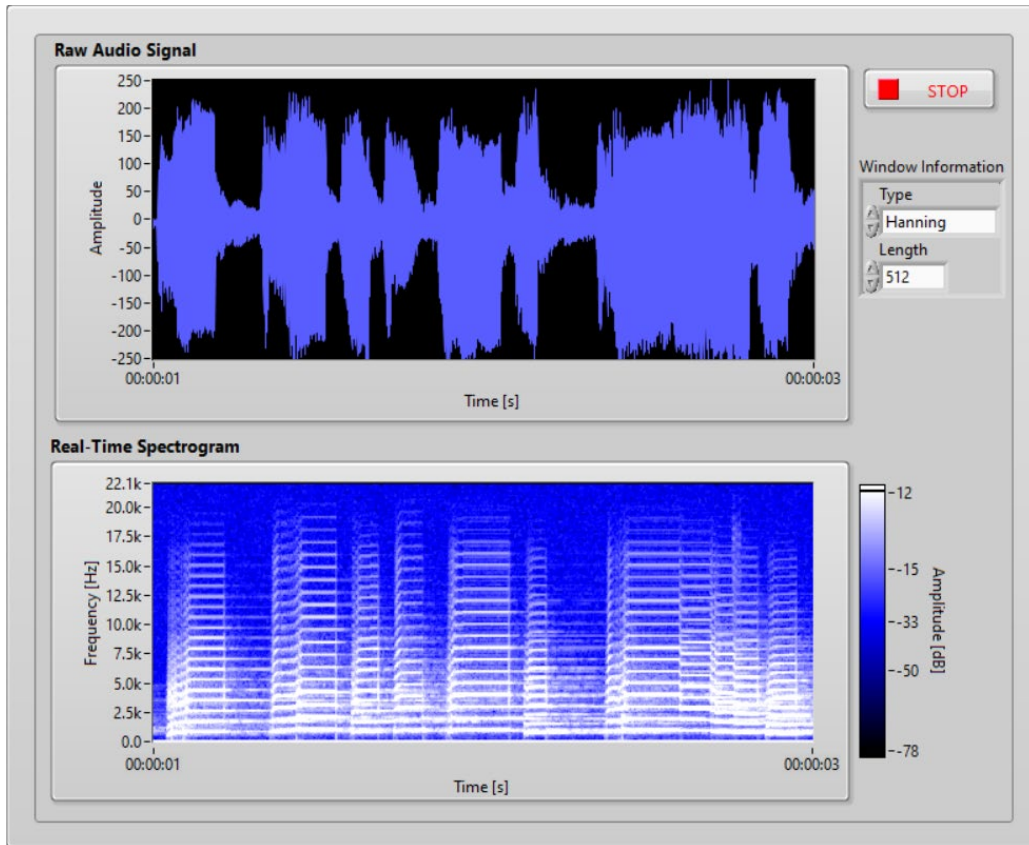


Figure 4-7. Wide window.

We can conclude that a narrow window gives better time localization but a poor frequency resolution, while a wide window gives better frequency resolution, but a worse time localization.

Task 1: Try different window lengths and observe the results.

Task 2: Make a comparison on the impact of the window type on the audio spectrogram.

4.4 Real-Time PWM Signal Generation

PWM stands for Pulse Width Modulation, which is a modulation technique used in electronics to encode information in the form of a pulsing signal. It is widely used in various applications, including controlling the speed of motors, dimming LEDs, generating analog signals, and more.

In PWM, the signal is typically a square wave with a fixed frequency and variable duty cycle. The duty cycle refers to the ratio of the pulse width (ON time) to the total period of the signal. By changing the duty cycle, the average power delivered to a device or component can be controlled.

For example, in motor control, a PWM signal can be used to regulate the speed of a motor. By adjusting the duty cycle of the PWM signal, the effective voltage and power applied to the motor

can be varied. A higher duty cycle will result in a higher average voltage and power, thus increasing the motor speed, while a lower duty cycle will reduce the average voltage and power, slowing down the motor.

PWM signals are generated using specialized circuits or microcontrollers capable of producing such signals. The frequency of the PWM signal determines how quickly it switches between high and low states, while the duty cycle determines the percentage of time the signal spends in the high state.

The advantage of PWM is that it allows for precise control of power levels or analog-like signals using digital components. By rapidly switching the signal on and off, the average power delivered can be adjusted smoothly, creating the effect of a variable voltage or analog control.

4.4.1 Experimental setup

The setup for this experiment, from the laboratory side, is consisted of:

- NI myRIO 1900 device connected to a PC within the DSP laboratory;
- LabView user interface.

The setup needed from the student's side is:

- Remote Desktop application.

The user interface is shown in Figure 4-8. It provides real-time PWM signal generation. Student can change the predefined parameters about the frequency and duty cycle of two PWM signals and observe their graphical representation and their control over two LEDs. The first PWM signal also controls the blinking of LED 0 on myRIO.

4.4.2 Goals of the experiment

The aim of this experiment is to observe how a PWM signal is interpreted by digital and analog outputs, to which LEDs are connected.

4.4.3 Experimental results

Generate two identical PWM signals and observe how these signals are interpreted by the digital and the analog output, both used to control LED lighting. The light intensity of the LED is between 0 and 255.

In the first case, in the case of the digital output, the generated signal is interpreted as it has only two values or states, "0" and "1" or "OFF" and "ON". Hence, in this case the LED is blinking.

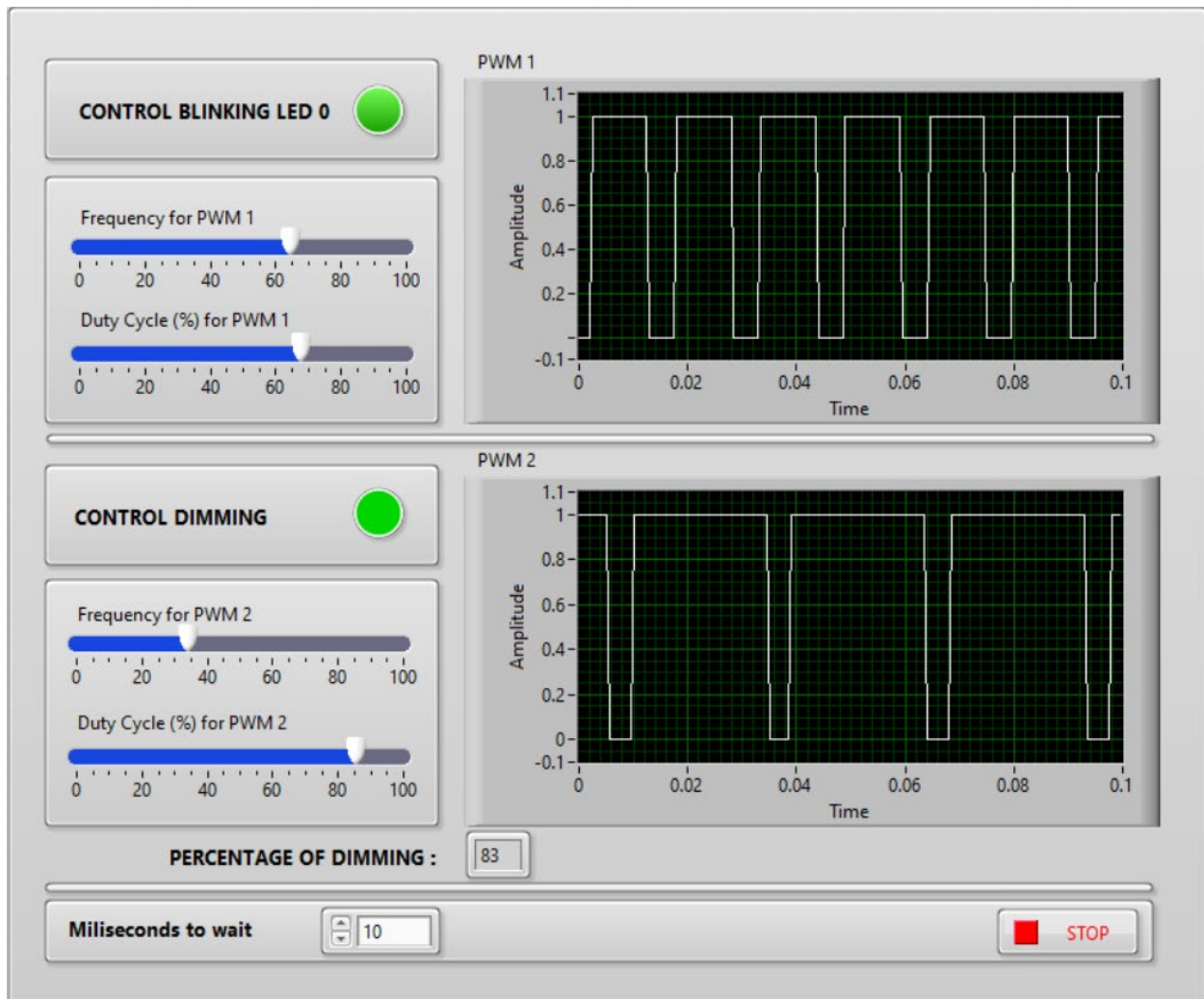


Figure 4-8. User interface for real-time PWM signal generation.

The blinking speed depends on the selected frequency, while the time spent in state “ON” depends on the selected duty cycle.

In the second case, in the case of the analog output, the generated PWM signal regulates the LED dimming (regulates the lightning intensity in the range of values between 0 and 255). By changing the frequency and the duty cycle you can change the percentage of dimming according to the following equation:

$$\text{Percentage of dimming} = \text{Duty cycle} - \text{Frequency} \times 6 \times 10^{-4}$$

Task: Try different frequencies and duty cycles and observe the results.

4.5 Conclusion

These experiments allow students to practically apply their knowledge from the field of IIR filtering, STFT, spectrograms and PWM signal generation, whether they work physically in a laboratory or remotely from home. With the friendly user interfaces, students can try different cases, compare results and draw conclusions.

Laboratory manual

Embedded Systems Laboratory

UbiLAB



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



Erasmus+



NATIONAL AGENCY
for European Educational
Programmes and Mobility

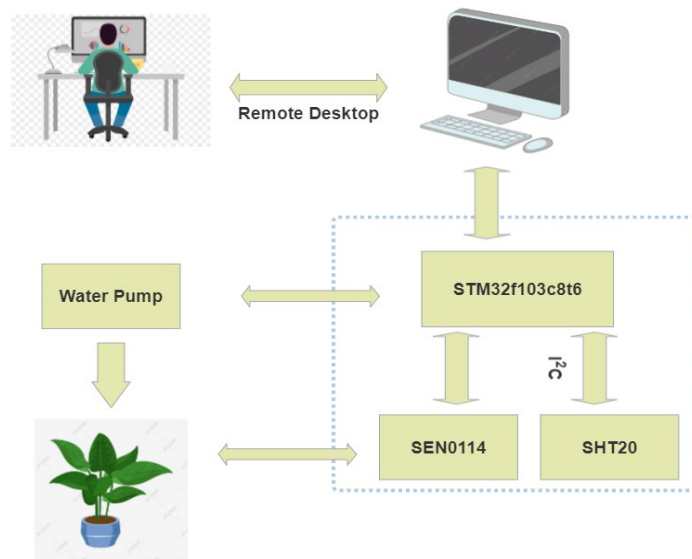
5 Embedded Systems Laboratory

5.1 Introduction

Remote Embedded laboratory is a remote laboratory that is part of the UbiLAB framework. The ultimate purpose of this remote laboratory is to provide a unique learning experience for students and enthusiasts interested in the field of embedded systems and microcontroller programming. The course consists of five laboratory exercises each designed to be done in a single session. The topics that are covered in this course are: General purpose I/O ports, Timers for general purpose, analog/digital convertor (ADC), universal asynchronous/synchronous receiver/transmitter (USART) and inter-integrated circuits (I2C). The combination of all five exercises leads to an autonomous irrigation system project. Students can access the connected system through the Apache Guacamole framework embedded into the LMS course from anywhere with internet connection. When the connection is established, they can control the hardware setup. The user has the freedom to choose between Arduino IDE and CooCox for the software development environment. Additionally, Hercules SETUP software is used to interface with the Serial monitor. After uploading and running the program, results can be observed through the Serial monitor and the live video stream.

The hardware solution consists of a personal computer, STM32f103c8t6 microcontroller, LED diodes that are connected on three different digital pins of the microcontroller, SEN0114 soil moisture sensor that is connected to an analog pin, SHT20 - temperature and humidity sensor connected to I2C compatible pins. Additionally, USB to TTL Adapter is connected between the microcontroller and personal computer for USART communication for serial print of the outputs and a video camera for observation of all visible outputs. This remote lab setup allows real-time interaction with the hardware solution. Students can send commands, read sensor data and observe the responses in real-time through a user-friendly interface. In this way students can understand the concepts better and verify the expected behavior of embedded systems.

The laboratory setup and the remote access is presented in Figure 5-1.



5.2 Laboratory experiment 1

The first laboratory experiment consists of a simple code that configures a selected I/O port which turns a LED diode connected through a 220 ohms resistor on or off depending on the set logic level. Additionally, two more diodes can be added in order to simulate the operation of a traffic light by using a for loop delay.

GPIO stands for General Purpose Input/Output. It is a type of interface that allows an electronic device, such as a microcontroller or a processor, to communicate with external devices by providing digital input/output signals. GPIO pins can be configured as either input or output pins, depending on the system requirement. When configured as an input, the GPIO pin can read the state of external devices, such as sensors or switches, by detecting changes in the voltage level on the pin. On the other hand, when configured as an output pin, GPIO can control the state of external devices, turning LEDs on/off or driving a motor, by outputting a voltage level on the pin.

In order to use the STM GPIO pins, first we have to configure them. The GPIO configuration consists of the following steps:

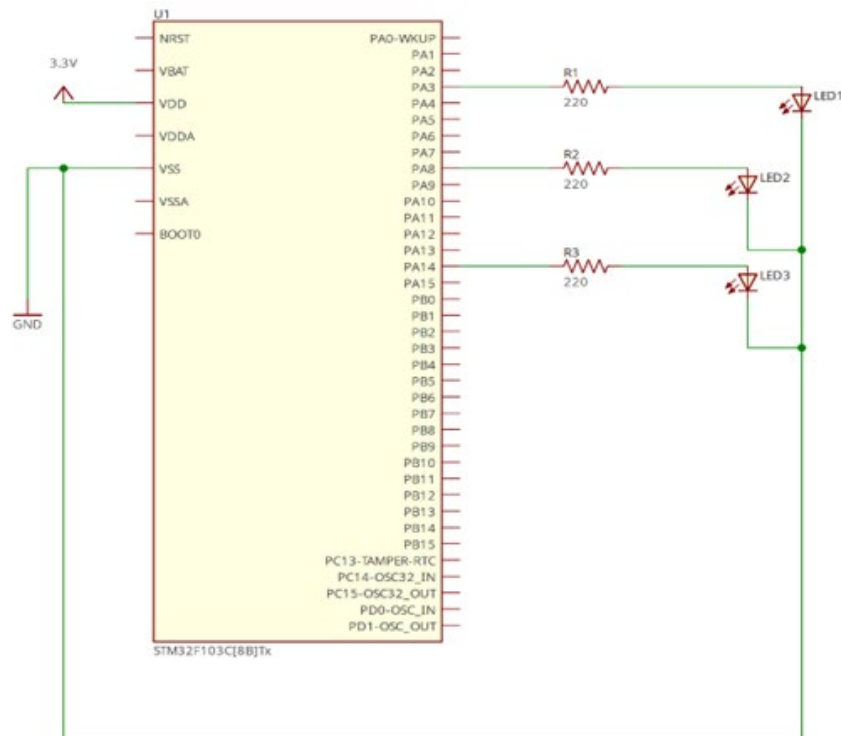
- Enable the peripheral clock
- Specify the GPIO pins to be configured
- Specify the speed for the configured pins
- Specify the operating mode for the configured pins
- Remap the pin if an alternate function is used

5.2.1 Experimental setup

Equipment:

- Embedded system development board (STM32f103c8t6)
- Three LEDs (red, green and yellow color)
- Three 220 ohm resistors
- Breadboard and jumper wires

The anode (positive lead) of the red LED is connected to pin PA3 of the development board and the cathode (negative lead) is connected to one end of the 220 ohm resistor. The other end of the resistor is connected to the GND pin of the STM. These connections are repeated for the yellow and green diode by connecting them to pins PA8 and PA14 respectively, as shown in Figure 5-2.



5.2.2 Goals of the experiment

The goal of the first exercise is to get started and become familiar with the tools and environment, as well as understanding the general purpose I/O ports.

5.2.3 Experimental results

Write a program in C language to configure and control the GPIO pins. Your program should do the following:

- Initialize pins PA3, PA8 and PA14 as output pins. In order to do that the peripheral clock must be enabled, select the mode of the pin as output and select the GPIO speed.
- Turn on the LED diodes on and off by using the GPIO_WriteBit() function.
- Insert a delay by using a for loop for longer on and off time.

The code is given below. For additional exercise, students can write a program to simulate the operation of traffic light by implementing multiple for loop delays.

```
#include<stm32f10x_gpio.h>
#include<stm32f10x_rcc.h>

GPIO_InitTypeDef GPIO_InitStructure;
int led=0;

int main(void) {

    RCC_APB2PeriphClock (RCC_APB2Periph_GPIOA, ENABLE);
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init (GPIOA, &GPIO_InitStructure);

}
while(1) {
    GPIO_WriteBit (GPIOA, GPIO_Pin_1, led);
    for(int i=0;i<900000;i++){
        led=~led;
    }
}
```

5.3 Laboratory experiment 2

This experiment begins with a brief overview of the operation of integrated timers and prescaler. In its most basic form timer modules represent a digital logic circuit that counts up every clock cycle. More functionalities are implemented in hardware to support the timer module so it can count up or down. The timer module can have a prescaler to divide the input clock frequency by a selectable value.

STM32 microcontrollers have a variety of timer modules that can be used to generate interrupts, measure time intervals and perform other time-based operations. Here are some of the most common timer modules of the STM32 microcontroller family:

- Basic Timers (TIM6 and TIM7): simple timers that can generate interrupts with a frequency up to 84 MHz They can be used for timekeeping, periodic events and software timing loops.
- General Purpose Timers (TIM2, TIM3, TIM4, TIM5, TIM9, TIM10, TIM11): versatile timers that can be configured for a wide range of timing and control applications.

They support input capture, PWM (Pulse Width Modulation) modes, output compare and other modes of operation.

- Advanced Control Timers (TIM1, TIM8): high-end timers that support advanced features such as motor control and high-resolution PWM. These timers operate at frequencies up to 168 MHz and offer multiple channels for advanced control applications.
- Low-Power Timer (LPTIM1, LPTIM2): timers that operate in ultra-low-power modes for extended battery use.

These different hardware timers in the STM32 microcontroller can operate in multiple modes. An STM32 timer module can operate in any of the following modes: timer mode, counter mode, PWM mode, advanced PWM mode, output compare mode, input compare mode etc.

5.3.1 Experimental setup

The setup for this experiment is the same as in experiment 1.

Equipment:

- Embedded system development board (STM32f103c8t6)
- Three LEDs (red, green and yellow color)
- Three 220 Ω resistors
- Breadboard and jumper wires

The anode (positive lead) of the red LED is connected to pin PA3 of the development board and the cathode (negative lead) is connected to one end of the 220 Ω resistor. The other end of the resistor is connected to the GND pin of the STM board. These connections are repeated for the yellow and green diodes by connecting them to pins PA8 and PA14 respectively, as shown in Figure 5-3.

5.3.2 Goals of the experiment

In this experiment, students will learn how to program the timers and PWM of an embedded system to control the brightness of an LED.

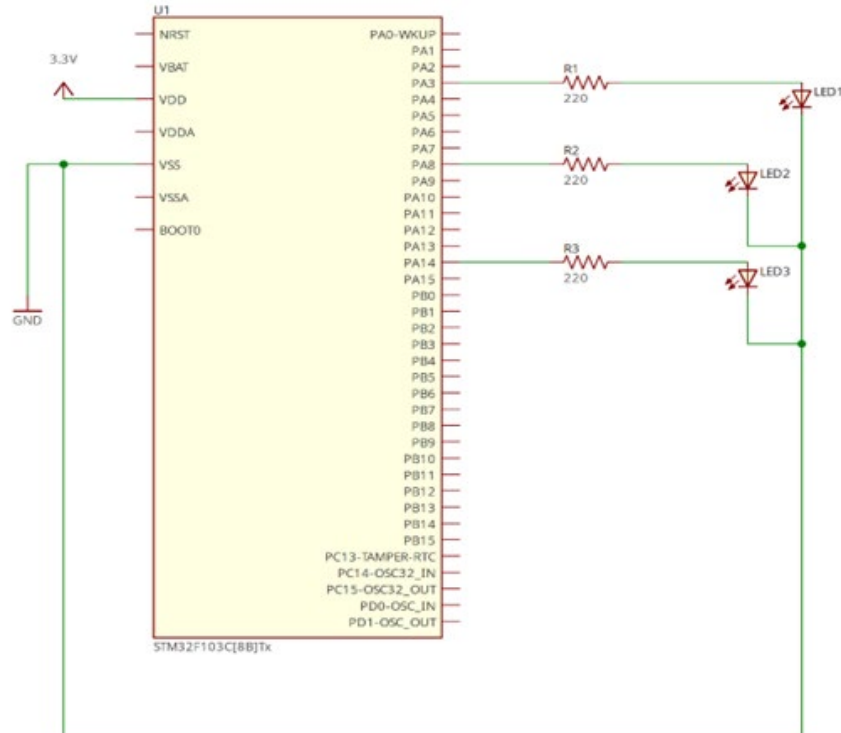


Figure 5-3. Experiment 2 connection schematic.

5.3.3 Experimental results

Write a program in C language to control and configure the timers and PWM:

- Turn on the clock for timer 2 (TIM2).
- Set the prescaler value to 71 and set the auto-reload register of TIM2 to 999. With this the timer will count till it reaches 999.
- Generate an update event to update the prescaler buffer.
- Reset the interrupt flag and turn on the timer.
- Write a loop to count the given time. Lastly, turn-off the timer.

An example code is given below. In the main program write a code to simulate traffic light operation by using the delay function.

```
void delay_ms (int ms)
{
    RCC -> APB1ENR |= 0x1;
    TIM2 -> PSC = 72-1;
    TIM2 -> ARR = 1000-1;
    TIM2 -> CNT = 0;
    TIM2 -> EGR |= 0x1;
    TIM2 -> SR &= ~(0x1);
    TIM2 -> CR1 |= 0x1;
}
```



```

while (ms > 0)
{
    while ((TIM2 -> SR & 0x1) == 0);
        TIM2 -> SR &= ~(0x1);
        ms--;
    }
TIM2 -> CR1 &= ~(0x1);
RCC -> APB1ENR &= ~(0x1 );
}

```

5.4 Laboratory experiment 3

In experiment 3 students will write a code to configure and read the ADC (Analog to Digital Converter) and use a development board (STM32) to test the program. The STM32 series of microcontrollers have a built-in ADC module that can be used to convert analog signals into digital for processing in software. Some of the key features of the SMT32 ADC are:

- 12 bits resolution
- 16 multiplexed channels
- Conversion modes: continuous conversion, single conversion and scan conversion.
- DMA (Direct Memory Access) support: the ADC can be configured to use DMA to transfer data from the ADC buffer to memory without CPU intervention.
- Trigger modes: software trigger, external trigger and timer trigger.
- Sampling rates: ranging from a few samples per second to several mega-samples per second.

To use the STM32 ADC module, the ADC registers have to be configured using a programming language and then start the conversion by a software or hardware trigger. Once the conversion is complete, the converted digital value can be read from the ADC data register and processed in software.

The SEN0114 moisture sensor, shown in Figure 5-4, is a capacitive soil moisture sensor that can be used to measure the water content in soil. The two metal probes that are inserted into the soil pass current through the soil and then the sensor measures the resistance between them to get the relative moisture level. The resistance between the probes varies depending on the water content in the soil, higher water level results in lower resistance and vice versa. The sensor has three pins: VCC, GND and SIG. The meaning of the measured values is given in Table 5-1.

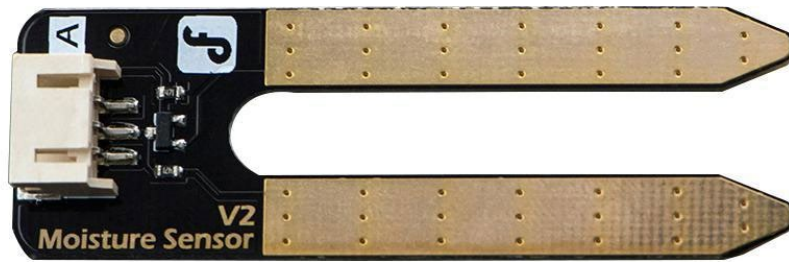


Figure 5-4. The SEN0114 moisture sensor.

Table 5-1. Meaning of measured values of SEN0114.

Measured values	Meaning
0-500	Very dry
500-1200	Good
>1200	Wet

5.4.1 Experimental setup

Equipment:

- Embedded system development board (STM32f103c8t6)
- SEN0114 moisture sensor
- Breadboard and jumper wires

The moisture sensor is connected to the development board as follows:

- Connect pin VCC to 3.3 V or 5 V
- Connect pin GND to ground
- Connect pin SIG to GPIO pin A5

5.4.2 Goals of the experiment

The goal of the experiment is to convert the measured value from analog to digital form and to store this information in the microcontroller memory.

5.4.3 Experimental results

Write a program in C language to configure and read the ADC. Your program should do the following:

- Configure GPIO pin A5 as an analog input. Enable the peripheral clock and set the GPIO speed.
- Set ADC mode as continuous and disable the external trigger.
- Read the analog voltage value from the ADC data register after each conversion. The voltage value corresponds to the resistance value of the SEN0114 moisture sensor value.
- Convert the analog voltage value to the corresponding moisture content using the calibration equation or lookup table, which can be found in the sensor datasheet.

The code for initialization and reading the measured value is given below.

```
#include"moisture_sensor.h"

GPIO_InitTypeDef GPIO_ADC_InitStruct;
void adc_init(void) {

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    GPIO_ADC_InitStruct.GPIO_Pin=GPIO_Pin_5;
    GPIO_ADC_InitStruct.GPIO_Mode=GPIO_Mode_AIN;
    GPIO_ADC_InitStruct.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_ADC_InitStruct);
    GPIO_PinRemapConfig(GPIO_Remap_ADC1_ETRGREG, ENABLE);
    ADC_DeInit(ADC1);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);

    ADC_InitStruct.ADC_Mode==ADC_Mode_Independent;
    ADC_InitStruct.ADC_ContinuousConvMode=ENABLE;
    ADC_InitStruct.ADC_ScanConvMode=DISABLE;
    ADC_InitStruct.ADC_NbrOfChannel=1;
    ADC_InitStruct.ADC_DataAlign=ADC_DataAlign_Right;
    ADC_InitStruct.ADC_ExternalTrigConv=ADC_ExternalTrigConv_None;
    ADC_Init(ADC1,&ADC_InitStruct);

    ADC_RegularChannelConfig(ADC1,ADC_Channel_5,1,ADC_SampleTime_1Cycles5);
    ADC_Cmd(ADC1,ENABLE);
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration( ADC1);
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);

}

uint16_t read_adc_value(ADC_TypeDef* ADCx) {
    uint16_t value=ADC_GetConversionValue(ADC1);
    return value;
}
```

5.5 Laboratory experiment 4

In experiment 4, we will learn about Universal synchronous asynchronous receiver transmitter (USART). This is serial communication that uses two wires, one for transmitting and the other one for receiving messages. Tx (transmit) pin is used for transmission of data. This pin is connected to Rx (receive) pin. Rx pin is used for receiving messages, as shown in Figure 5-5. Data is sent and received as serial bit stream of 7 or 8 bits at programmed rate. This experiment is a continuation of the previous experiment. The measured values, from the previous experiment, stored on the microcontroller memory are sent and displayed on the serial monitor using the USART module and Hercules SETUP software.

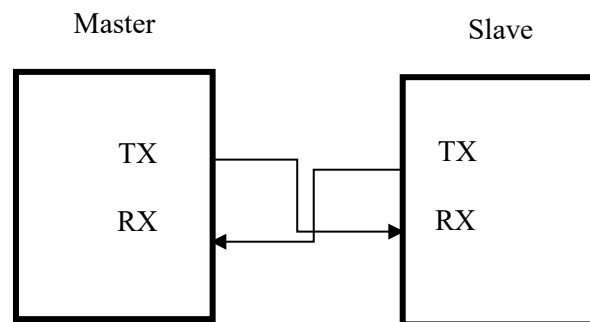


Figure 5-5. USART master-slave connection.

Frame Format

The asynchronous data format consists of a start bit, 7 or 8 bits for data, even/odd or no parity bit and one or two bits for stop condition.

USART transmission

Start bit

The communication starts with the start bit. Tx line is set to high level when it is not transmitting data. When start is requested, the Tx line is switched to low level for one clock cycle. When the transition is detected by receiver, it starts to read data frame.

Data frame

Data frame is 7 or 8 bits. Usually the least significant bit is sent first.

Parity bit

Parity bit is used to check the reliability of data. The bit can be changed during data transfers. When data is received it checks the number of bits with value of 1. If the parity bit is set to even parity the number of 1's in the data should be even or if parity bit is set to odd parity, then the number of 1's should be an odd number.

Stop bit

Stop bit is used to force the communication to end. This is realized when transmitter switches Tx line to high level.

5.5.1 Experimental setup

Equipment:

- Embedded system development board (STM32f103c8t6)
- Breadboard and jumper wires
- SEN0114 moisture sensor
- Hercules SETUP utility

The moisture sensor is connected to the development board as follows:

- Connect pin VCC to 3.3 V or 5 V
- Connect pin GND to ground
- Connect pin SIG to GPIO pin A5

5.5.2 Goals of the experiment

The goal of the experiment is to send the measured values stored on the microcontroller memory to the serial monitor of the PC.

5.5.3 Experimental results

Write a program in C language to configure the USART module and display the measured values on the serial monitor. Your program should do the following:

- Configure the USART peripheral: enable the USART peripheral and choose the appropriate USART instance. Next set up the baud rate, data, stop and parity bits accordingly. Lastly, configure the GPIO pins for USART communication.
- Implement a function to send data over USART. The function should take the measured values stored in the microcontroller memory and use the USART_SendData() or equivalent function to send the data.
- Display the values using the Hercules SETUP. Configure the Hercules SETUP serial terminal to match the settings used in STM32 program (data bits, parity bits, stop bits and baud rate) and start the serial communication in the Hercules SETUP terminal.

5.6 Laboratory experiment 5

In experiment 5, we will learn about Inter-Integrated-Circuit (I2C) protocol and how to use an I2C driver to interface with a digital temperature and humidity sensor. I2C communication is predefined serial communication between several devices. I2C protocol is defined between master and slaves. Master is the device that generates a clock to synchronize the communication (Figure 5-6).

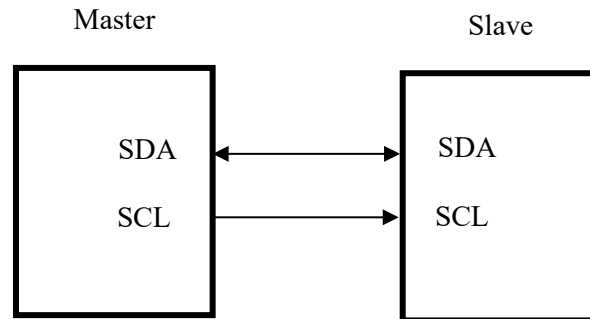


Figure 5-6. I2C master-slave connection.

This protocol requires two wires for communication, but these two wires can support up to 1008 slaves. SDA line is used for data transmission in both directions and SCL wire is used for clock and the direction is from master to the slaves.

Messages contain two frames of data: start condition, an address frame, acknowledge bit and data (Table 5-2). Data transmission starts with start condition by switching SDA line from high level to low before switching SCL line from high to low. The address frame is the starting frame of any communication sequence. It contains the address of the device (7 bits), one bit for read or write command and the 9th bit is used for acknowledge. If the 9th bit is not set up that means that receiving device does not receive frame or the message is not understandable. The direction of SDA depends of write or read bit in address frame. Stop condition initiates end of communication by switching SCL line form high to low voltage level before switching SCL line from high to low voltage level.

Table 5-2. I2C message.

Start	Address frame	Read/Write Bit	ACK/NACK bit	Data frame	ACK/NACK bit	Data frame	ACK/NACK bit	Stop
-------	---------------	----------------	--------------	------------	--------------	------------	--------------	------

5.6.1 I2C transmission

I2C transmission is realized using following steps:

1. The master sends start conditions to the connected devices by switching the SDA line from high to low level before switching SCL line from high to low level.
2. The master sends the address of the device that wants to communicate with and the address is followed by a read/write bit.
3. Slave that has recognized the address that is sent from the master, sends acknowledge bit by switching SDA line to low level for one bit.
4. The master sends or receives data
5. After successful transmission, the acknowledge bit is sent
6. Stop condition is generated by switching SCL line high before switching SDA line high.

The SHT20 is a digital temperature and humidity sensor designed for various applications that require precise temperature and humidity measurements. Some of its key features are:

1. Measurement accuracy: up to $\pm 3\%$ for relative humidity (RH) and $\pm 0.3^\circ\text{C}$ for temperature measurements, thus ensuring reliable and precise data.
2. Digital interface: communication using standard I2C interface.
3. Wide measurement range: it can measure humidity from 0% to 100% RH and temperature from -40°C to $+125^\circ\text{C}$, allowing it to be used in a variety of environments and applications.
4. Fast response time: allowing quick and accurate measurements in dynamic conditions.
5. Calibration and digital signal processing: the sensor is factory calibrated and incorporates digital signal processing techniques that enhance the reliability and stability of the measured data.
6. Low power consumption: suitable for battery powered applications.

The SHT20 sensor is commonly used in weather stations, HVAC systems, industrial automation, medical devices and consumer electronics. Its small size, high accuracy and digital interface make it a reliable choice for humidity and temperature measurement applications. The sensor is given in Figure 5-7.

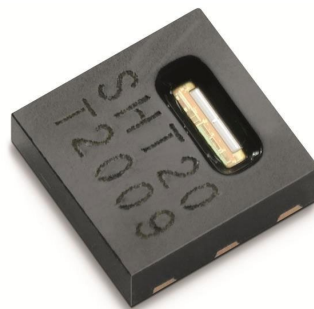


Figure 5-7. The SHT20 digital temperature and humidity sensor.

5.6.2 Experimental setup

Equipment:

- Embedded system development board (STM32f103c8t6)
- SHT20 I2C sensor for temperature and humidity
- Breadboard and jumper wires

The temperature and humidity sensor are connected to the development board as follows:

- Connect pin VCC to 3.3 V or 5 V
- Connect pin GND to ground
- Connect pin SCL to GPIO pin B6
- Connect pin SDA to GPIO pin B7

5.6.3 Goals of the experiment

The goal of this experiment is to establish I2C communication and measure the values of ambient temperature and humidity using SHT20 sensor.

Write a program in C language to configure I2C driver and read the values from SHT20 sensor. Your program should do the following:

- Initialization of I2C communication:
 - Enable APB1 peripheral clock for I2C
 - Enable APB2 clock for SCL and SDA pins
 - Set the pins, mode and speed
 - Set the GPIO mode
 - Remap the configuration of pins
 - Set I2C clock speed, mode, duty cycle, enable acknowledge and address of master

5.6.4 Experimental results

```
#include <stm32f10x_gpio.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_tim.h>
#include <stm32f10x_i2c.h>

#define SHT20_I2C_ADDR                0x80
#define SHT20_NOHOLD_TEMP_REG_ADDR    0xF3
#define SHT20_NOHOLD_HUMDTY_REG_ADDR  0xF5
#define SHT20_HOLD_TEMP_REG_ADDR      0xF3
#define SHT20_HOLD_HUMDTY_REG_ADDR    0xF5
#define ERROR_I2C_TIMEOUT              998
#define ERROR_BAD_CRC                  999
#define SHIFTED_DIVISOR                0x988000
I2C_InitTypeDef I2C_InitStruct;
GPIO_InitTypeDef GPIO_I2C_InitStruct;
```



```

void init_I2C1(void){

    // enable APB1 peripheral clock for I2C1
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
    // enable clock for SCL and SDA pins
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

    /* setup SCL and SDA pins
    * You can connect I2C1 to two different
    * pairs of pins:
    * 1. SCL on PB6 and SDA on PB7
    * 2. SCL on PB8 and SDA on PB9
    */
    GPIO_I2C_InitStruct.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;

    // we are going to use PB6 and PB7
    GPIO_I2C_InitStruct.GPIO_Mode = GPIO_Mode_AF_OD;
    // set pins to alternate function
    GPIO_I2C_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    // set GPIO speed
    //GPIO_InitStruct.GPIO_OType = GPIO_OType_OD;
    // set output to open drain --> the line has to be only pulled low, not
    driven high
    //GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_UP;
    // enable pull up resistors
    GPIO_Init(GPIOB, &GPIO_I2C_InitStruct);
    // init GPIOB
    GPIO_PinRemapConfig(GPIO_Remap_I2C1, ENABLE);

    // Connect I2C1 pins to AF

    I2C_InitStruct.I2C_ClockSpeed=100000;
    I2C_InitStruct.I2C_Mode=I2C_Mode_I2C;
    I2C_InitStruct.I2C_DutyCycle=I2C_DutyCycle_2;
    I2C_InitStruct.I2C_Ack=I2C_Ack_Disable;

    I2C_InitStruct.I2C_AcknowledgedAddress=I2C_AcknowledgedAddress_7bit; //size
    of the address
    I2C_InitStruct.I2C_OwnAddress1=0x00; //address of the
    Microcontroller

    I2C_Init( I2C1,&I2C_InitStruct); //init I2C
    I2C_Cmd(I2C1, ENABLE); //Enables or disables
    the specified I2C peripheral.

}

void I2C_start(I2C_TypeDef* I2Cx, uint8_t address, uint8_t direction){
    // Send I2C1 START condition
    I2C_GenerateSTART(I2Cx, ENABLE);

    // Send slave Address for write
    I2C_Send7bitAddress(I2Cx, address, direction);

}

void I2C_write(I2C_TypeDef* I2Cx, uint8_t data)

```

```

        {

            I2C_SendData(I2Cx, data);
            // wait for I2C1 EV8_2 --> byte has been transmitted
            //while(!I2C_CheckEvent(I2Cx,
I2C_EVENT_MASTER_BYTE_TRANSMITTED));
        }

        /* This function reads one byte from the slave device
        * and acknowledges the byte (requests another byte)
        */
uint8_t I2C_read_ack(I2C_TypeDef* I2Cx) {
    // enable acknowledge of recieved data
    I2C_AcknowledgeConfig(I2Cx, ENABLE);
    // wait until one byte has been received
    //while( !I2C_CheckEvent(I2Cx,
I2C_EVENT_MASTER_BYTE_RECEIVED) );
    // read data from I2C data register and return data
byte
    uint8_t data = I2C_ReceiveData(I2Cx);
    return data;
}

        /* This function reads one byte from the slave device
        * and doesn't acknowledge the recieved data
        */
uint8_t I2C_read_nack(I2C_TypeDef* I2Cx) {
    // disabe acknowledge of received data
    // nack also generates stop condition after last byte
received
    // see reference manual for more info
    I2C_AcknowledgeConfig(I2Cx, DISABLE);

    // wait until one byte has been received
    //while( !I2C_CheckEvent(I2Cx,
I2C_EVENT_MASTER_BYTE_RECEIVED) );
    // read data from I2C data register and return data
byte
    uint8_t data = I2C_ReceiveData(I2Cx);
    return data;
}

void I2C_stop(I2C_TypeDef* I2Cx) {
    // Send I2C1 STOP Condition
    I2C_GenerateSTOP(I2Cx, ENABLE);
}
uint16_t SHT20_readValue(I2C_TypeDef* I2Cx, uint8_t address, uint8_t reg) {
    uint8_t lsb,msb;
    I2C_start(I2Cx,address,I2C_Direction_Transmitter);
    I2C_write(I2Cx,reg);
    I2C_start(I2Cx,address,I2C_Direction_Receiver);
    delay_ms(20);
    msb=I2C_read_ack(I2Cx);
    delay_ms(20);
    lsb=I2C_read_ack(I2Cx);
}

```

```

    u8 checksum=I2C_read_ack(I2Cx);
    delay_ms(20);
    I2C_stop(I2Cx);

    uint16_t raw_value=((uint16_t) msb<<8)|(uint16_t) (lsb);
//    if (SHT20_checkCRC((u16)raw_value, (u8)checksum) != 0)
//        return (ERROR_BAD_CRC);
    return (raw_value & 0xfffc);

}

float SHT20_readHumidity(I2C_TypeDef* I2Cx, uint8_t address){
    float tempRH;
    float rh;
    uint16_t
rawHumidity=SHT20_readValue(I2Cx,address,SHT20_NOHOLD_HUMDTY_REG_ADDR);

//    if (rawHumidity == ERROR_I2C_TIMEOUT || rawHumidity == ERROR_BAD_CRC)
//        return (rawHumidity);
    tempRH=rawHumidity*125.0/65535.0;
    tempRH=tempRH-6.0;
    return tempRH;

}

float SHT20_readTemp(I2C_TypeDef* I2Cx, uint8_t address){
    float temp;
    float rh;
    uint16_t
rawTemp=SHT20_readValue(I2Cx,address,SHT20_NOHOLD_TEMP_REG_ADDR);
//
//    if (rawTemp == ERROR_I2C_TIMEOUT || rawTemp == ERROR_BAD_CRC)
//        return (rawTemp);
    temp=rawTemp*(175.72/65536.0);
    temp=temp-46.85;
    return temp;
}

u8 SHT20_checkCRC(u16 message_from_sensor, u8 check_value_from_sensor)
{
    u8 i;
    u32 divisor = (u32)SHIFTED_DIVISOR;
    u32 remainder = (u32)message_from_sensor << 8;

    remainder |= (u32)check_value_from_sensor;

    for(i = 0 ; i < 16 ; i++) {
        if(remainder & (u32)1 << (23 - i))
            remainder ^= divisor;
        divisor >>= 1;
    }
    return (u8)remainder;
}

```

5.7 Conclusion

The remote Embedded laboratory offers a convenient and interactive platform for students and enthusiasts interested in embedded systems. By providing virtual access to the real hardware and comprehensive documentation, this laboratory enables its users to gain practical experience in microcontroller programming and further develop their understanding of embedded systems.

The key advantages of the remote embedded laboratory are:

- **Virtual Access:** allowing users to access the embedded system setup from anywhere as if they are physically using them in the laboratory.
- **Real-time interaction:** sending commands, reading sensor data and observing the system response in real time.
- **Experiment library:** the laboratory has a set of pre-designed experiments covering various topics that can be combined to form a practical embedded system project.
- **Comprehensive documentation:** a detailed documentation is provided for every experiment, including circuit diagrams, code examples, step-by-step instructions and explanations of the fundamental concepts.

Overall, the remote embedded laboratory offer an innovative and effective approach to remote learning and experimentations in the field of embedded systems. It provides a flexible and accessible platform for individuals to explore and expand their knowledge of embedded systems and microcontroller programming.

Laboratory manual

Virtual Instrumentation Laboratory Platform



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



6 Virtual Instrumentation Laboratory Platform

6.1 Introduction

The Remote Virtual Instrumentation Laboratory Platform (RVILP, given in Figure 6-1) provides engineers with a versatile platform for conducting a wide range of experiments using standard instrumentation. These include multimeters, power supplies, oscilloscopes, function generators, data acquisition cards, etc. The RVILP operates on an event-driven state machine programming architecture, enabling engineers to initiate specific virtual instruments (VIs) for carrying out dedicated tasks and updating the platform using global variables. This programming approach allows for efficient control and coordination of the experiments. To facilitate seamless communication among the virtual instruments, functional global variables are utilized. These variables serve as a means of sharing information pertaining to physical quantities, measurement units, status and control messages, as well as error control. Remote access to the VIs is made possible through a standard clientless remote desktop gateway, such as Apache Guacamole. This ensures that engineers can conveniently access and control the RVILP from any location, further enhancing the flexibility and usability of the platform.

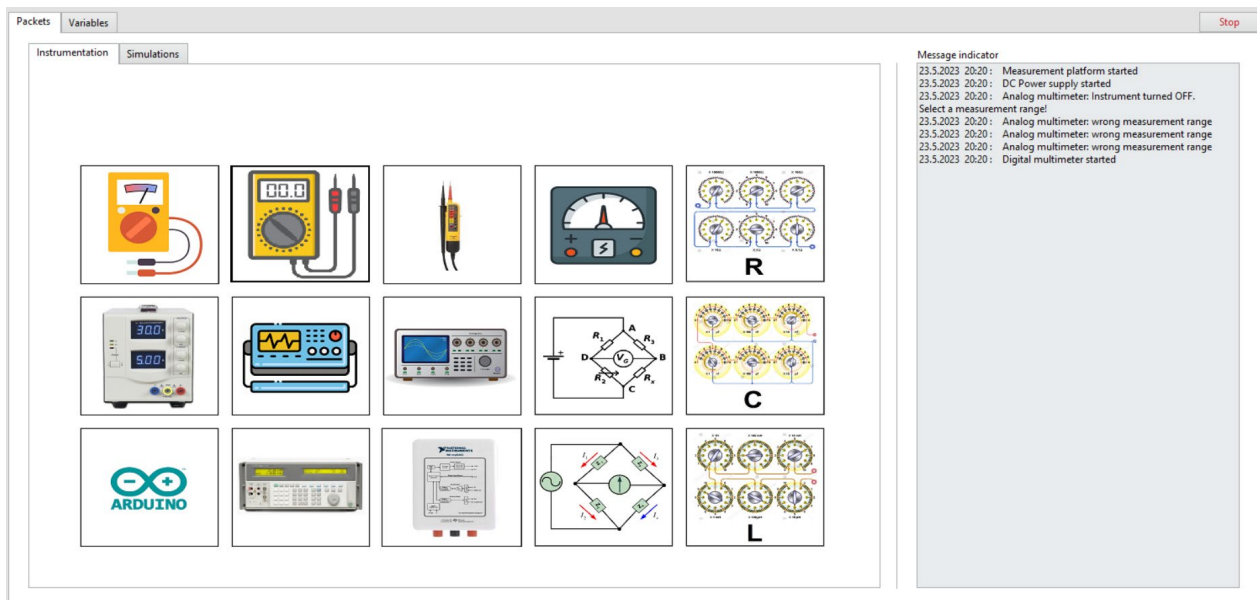


Figure 6-1. Front panel of the virtual instrumentation laboratory platform.

The RVILP consists of two main components: Instrumentation and simulation. In the Instrumentation section, users can access a comprehensive collection of standard instruments commonly found in engineering laboratories. These virtual instruments are equipped with the ability to communicate with one another, enabling the interfacing of real physical signals. This functionality allows users to replicate and interact with genuine measurement scenarios. On the other hand, the simulation tab provides virtual instruments specifically designed to simulate various instruments or operational principles commonly encountered in the field of measurement science and technology. These simulation tools offer users the opportunity to explore and understand theoretical concepts and experimental setups in a virtual environment. Additionally, the platform offers a variables tab where users can conveniently view, export, or store the signals generated by the global variables. This feature allows for easy access to and management of the data associated with the global variables used throughout the experiments.

The primary objective of the RVILP platform is to create a virtual laboratory environment that replicates real-world experiments, enabling the monitoring of measurement parameters using data acquisition cards through the virtual instrumentation laboratory platform. This concept has broad applicability across various engineering domains since virtual instruments are versatile tools essential for conducting a wide range of measurements. The platform offers hardware support for integrating National Instruments-related hardware or Arduino boards into the experiments. When utilizing Arduino development boards, users need to upload a specific open-source firmware to enable their compatibility with the virtual platform. Once the firmware is uploaded, the Arduino board functions as a standard data acquisition card, capable of reading analog channels or performing read/write operations on digital channels. Moreover, the platform supports popular digital protocols like I2C, SPI, UART, and PWM.

Beyond providing remote access to a virtual laboratory setup, the virtual instrumentation laboratory platform offers several advantages compared to traditional physical laboratory setups. One significant benefit is the ease of sharing and replicating experiments. Users can access the same virtual instruments and setups from anywhere in the world, facilitating collaboration among researchers and enabling students to access laboratory resources regardless of their location. Additionally, the platform serves as a valuable teaching tool to enhance engineering education. By offering a virtual instrumentation laboratory environment that simulates real-world experimentation, it provides students with an interactive and engaging learning experience. Furthermore, students can experiment freely without concerns about damaging physical equipment or consuming expensive resources.

This part of the laboratory manual comprises three laboratory exercises that utilize the virtual instrumentation laboratory platform. The purpose of these exercises is twofold: to establish a proof of concept and to inspire other users to leverage the potential of the platform. To showcase its versatility, the laboratory exercises are implemented in three distinct scenarios:

1. Complete virtual instrumentation utilization,
2. Utilizing virtual measurement instruments in conjunction with a specific hardware setup,
3. Excitation and response measurement employing the Remote Virtual Instrumentation Laboratory Platform.

6.2 Laboratory experiment 1: Voltmeter accuracy verification (calibration)

To ensure accurate measurements of physical quantities, it is essential to understand the accuracy of the measuring instrument. These characteristics can be obtained from the technical datasheet of the instrument and is typically represented as the maximum error occurring within a specified measurement range. In the case of digital instruments, accuracy is commonly expressed as follows:

$$\Delta X = \pm(A\% + B\% + C),$$

where A% is the error of the reading, B% is the error of the measurement range, and C is the resolution error expressed in digits.

For the analog instruments, the accuracy is expressed through the accuracy class by the following formula:

$$\text{a. c.} = \frac{|\Delta x_i|_{\max}}{x_{mr}} \cdot 100,$$

where "a.c." is the accuracy class of the instrument, Δx_i is the maximum absolute error in the measurement range x_{mr} .

Throughout its lifespan, an instrument is subjected to various conditions that can impact its accuracy. Therefore, periodic verification (calibration) becomes necessary to either maintain its existing accuracy or establish a new level of accuracy that best reflects the instrument's current condition. In this particular exercise, students are introduced to the process of verifying voltmeters, which serves as the initial step in the calibration process. Generally, there are three methods available for testing measuring instruments during calibration: the compensation method, the comparison method using a more precise instrument, and the utilization of a specialized calibration device (calibrator). In this exercise, the calibration method employs a virtual calibrator. The process involves measuring the voltage generated by the calibrator using both an analog and digital voltmeter. It is crucial that all measurements with the instrument are conducted within the same measuring range, while the calibrator's measuring range can be adjusted during the process to achieve the most accurate readings. The accuracy of the instrument is then assessed by calculating the verification error. The value obtained through adjustment with the calibrator represents the "true" value, while the measured value corresponds to the reading obtained from the measuring instrument.

In the realm of metrology, it is recognized that the quantitative evaluation of a physical quantity should be complemented with a qualitative description encompassing the unit of measurement and a statement concerning the reliability of the measurement. This statement is known as *measurement uncertainty* and should be based on objective facts derived from the technical or scientific understanding of the measurement process. The concept of measurement uncertainty was established by the ISO/BIMP "Guide to the expression of uncertainty in measurement," commonly referred to as GUM, in 1992. Since then, this concept has gained international acceptance as the foundation for determining measurement uncertainty in metrology. According to the GUM, the total measurement uncertainty (y) is determined by applying the law of propagation of measurement uncertainty, which states:

$$u(y) = \sqrt{u_1^2(y) + u_2^2(y) + \dots + u_n^2(y)},$$

where $u_1(y) - u_n(y)$ represents the contribution of each individual input quantity to the total measurement uncertainty of the measurement.

6.2.1 Experimental setup

The experiment setup in this exercise consists of a virtual calibrator and two voltmeters, analog (AVM) and digital (DVM). The simplified electrical circuit of the laboratory exercise is given in Figure 6-2.

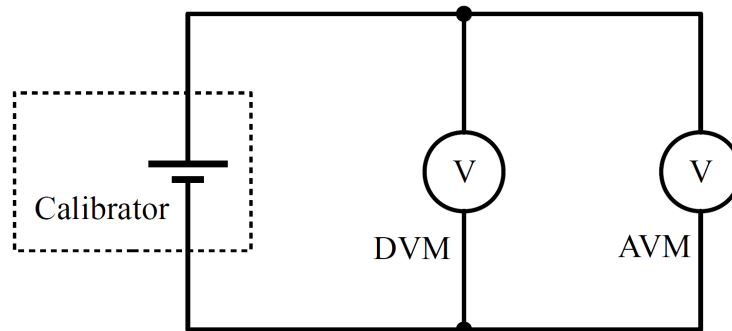
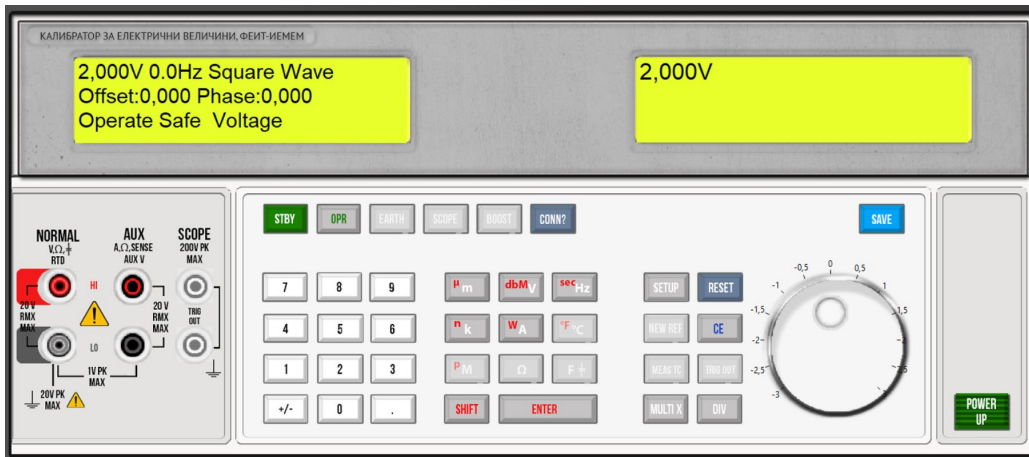


Figure 6-2. Experiment setup for verification of analog and digital voltmeters.

The measurand in this scenario refers to the direct current (DC) voltage produced by the calibrator, effectively functioning as a DC voltage source. This voltage is measured by both the analog and digital voltmeters, which are connected in parallel and communicate via the same virtual channel, such as CH1. All instruments are activated and controlled using the virtual instrumentation laboratory platform. The front panels of the virtual instruments can be observed in Figure 6-3.



a) Virtual calibrator



b) Virtual digital voltmeter



c) Virtual analog voltmeter

Figure 6-3. Front panel of the virtual instruments.

The readings obtained from both the analog and digital voltmeters are directly proportional to the predetermined voltage set on the virtual calibrator. The accuracy verification process is conducted at multiple points along the measurement range, such as -100%, -80%, ..., 0%, 20%, 40%, ..., and 100%. However, before taking measurements, it is essential to properly configure the voltmeters for DC voltage measurements. This entails ensuring the correct virtual channel is selected (e.g., CH1), specifying the measurement quantity as DC voltage, and choosing the appropriate measurement range. These adjustments are necessary to ensure accurate and reliable readings during the calibration process.

6.2.2 Experimental results

The primary objective of the laboratory exercise is to validate the claimed accuracy provided by the manufacturer. To achieve this, we assess the accuracy of both analog and digital voltmeters within a specified measurement range, following the procedure outlined in the previous chapter. It is crucial to meticulously record the voltmeter readings and appropriately input them into the result tables. Subsequently, we calculate the absolute and relative measurement errors using the virtual calibrator, comparing these values against the maximum allowable error of the voltmeters. The final column in the results tables should reflect the computation of the measurement uncertainty associated with the verification process. Please refer to the following data for detailed technical specifications of all measuring instruments utilized in the verification procedure.

6.2.2.1 Virtual calibrator DC voltage specifications

Measurement range	Absolute uncertainty ± [% of output + resolution]	
	% of output	Resolution
100 mV	0.15	0.1 mV
1 V	0.1	1 mV
10 V	0.1	10 mV
100 V	0.15	0.1 V
1000 V	0.2	0.2 V

6.2.2.2 Virtual digital voltmeter accuracy specifications

Measurement range	Digital voltmeter accuracy ± [% of reading + resolution]	
	% of reading	Resolution
400 mV	1	1 mV
4 V	1	3 mV
40 V	0.5	30 mV
400 V	0.5	300 V
1000 V	1	3 V

Fill in the following tables with the measurement data obtained from the measurement process. The notations used in the tables are as follows:

U_{cal} – virtual calibrator output voltage

U_{instr} – measured voltage with the instrument

$\Delta U = U_{instr} - U_{cal}$ – verification absolute error

$$\delta U = \frac{\Delta U}{U_{cal}} \cdot 100 - \text{verification relative error}$$

$\pm \Delta u_{max}$ – voltmeter accuracy

$\pm u$ – standard absolute measurement uncertainty

Table 6-1. Accuracy verification of digital voltmeter.

No.	% U_{mr}	U_{cal} [V]	U_{instr} [V]	ΔU [V]	δU [%]	$\pm \Delta U_{max}$ [V]	$\pm u$ [V]
1.	100	4	3,973	-0,027	-0,67	0,043	0,014
2.	90	3,6	3,575	-0,025	-0,69	0,039	0,013
3.	80	3,2	3,177	-0,023	-0,71	0,035	0,012
4.	70	2,8	2,779	-0,021	-0,75	0,031	0,011
5.	60	2,4	2,381	-0,019	-0,79	0,027	0,001
6.	50	2	1,983	-0,017	-0,85	0,023	0,001
7.	40	1,6	1,585	-0,015	-0,93	0,019	0,011
8.	30	1,2	1,187	-0,013	-1,08	0,015	0,012
9.	20	0,8	0,789	-0,011	-1,37	0,011	0,013
10.	10	0,4	0,391	-0,009	-2,25	0,007	0,014

Voltmeter measurement range: $U_{mr}=4\text{ V}$

Plot the obtained results from the verification procedure of the DVM (absolute errors, accuracy limits, calibrator output voltage, and uncertainty bars):

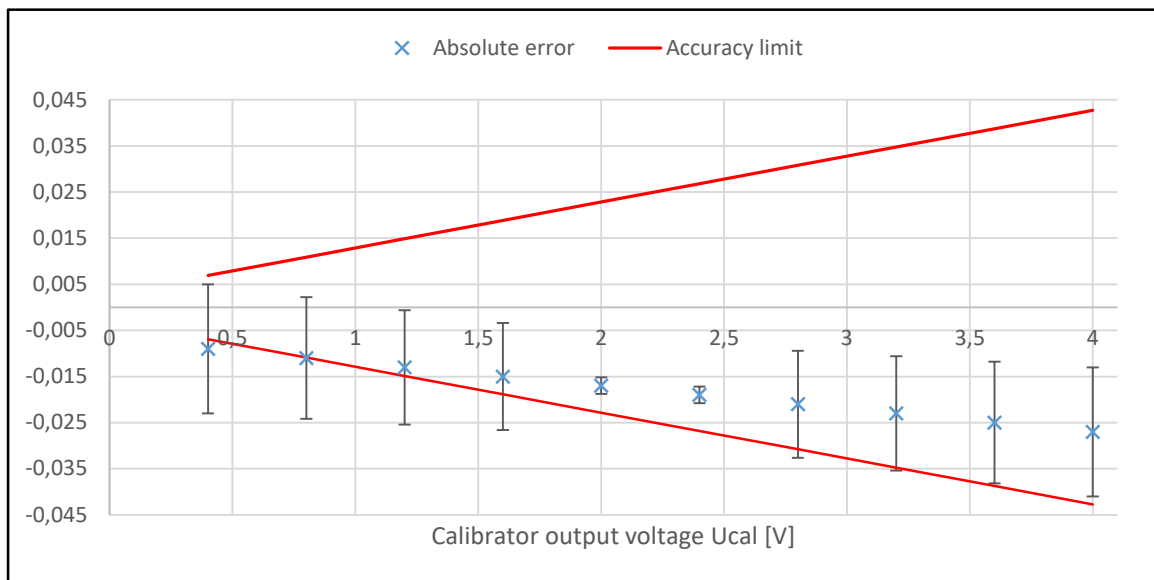


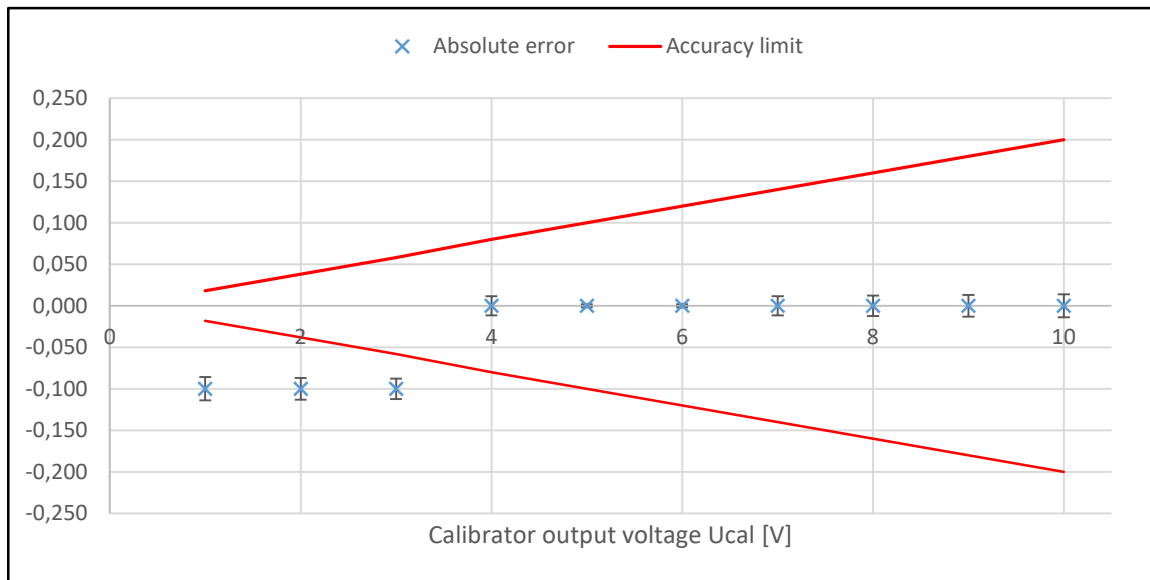
Table 6-2. Accuracy verification of analog voltmeter.

No.	$\%U_{mr}$	U_{cal} [V]	U_{instr} [V]	ΔU [V]	δU [%]	$\pm\Delta U_{max}$ [V]	$\pm u$ [V]
1.	100	10	10,0	0,0	0	0,20	0,014
2.	90	9	9,0	0,0	0	0,18	0,013
3.	80	8	8,0	0,0	0	0,16	0,012
4.	70	7	7,0	0,0	0	0,14	0,011
5.	60	6	6,0	0,0	0	0,12	0,001
6.	50	5	5,0	0,0	0	0,10	0,001
7.	40	4	4,0	0,0	0	0,08	0,011
8.	30	3	2,9	-0,1	-3,3	0,05	0,012
9.	20	2	1,9	-0,1	-5	0,03	0,013
10.	10	1	0,9	-0,1	-10	0,01	0,014

* the accuracy class of the analog voltmeter is 2

Voltmeter measurement range: $U_{mr} = 10\text{ V}$

Plot the obtained results from the verification procedure of the AVM (absolute errors, accuracy limits, calibrator output voltage, and uncertainty bars):



What conclusions can be drawn from the results obtained during the verification of the analog and digital multimeters?

6.3 Laboratory experiment 2: Verification of the first Kirchoff law

The aim of this exercise is confirmation of the first Kirchoff law by simulation and realization of simple electrical circuit. The first Kirchoff law states "the sum of currents that enter or leave a given junction of the electrical circuit is equal to zero". **Junction** (Figure 6-4) is a point where at least three branches of the electrical circuits are joined.

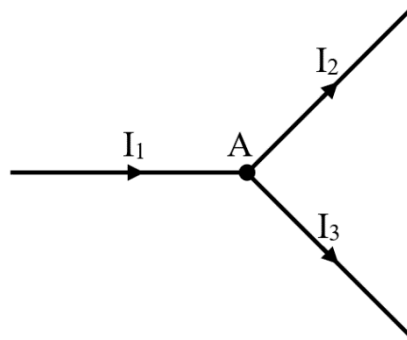


Figure 6-4. Simplified representation of three branches of electrical circuit forming the junction "A".

The part of the electrical circuit given in the figure above consists of three branches. The branches are joined in the junction A, and the following electrical currents are defined: I₁, I₂ and I₃. The first Kirchoff law for the junction A of the electrical circuit is:

$$(+I_1) + (-I_2) + (-I_3) = 0$$

In the mathematical formulation of the first Kirchoff law it is assumed that the currents entering the junction (I₁ in this case) are positive, while the currents leaving the junction (I₂ and I₃) are negative. It is clear that the first Kirchoff law also holds in case of opposite current directions. This laboratory exercise demonstrates the exploitation of the RVILP for verification of the first Kirchoff law in three steps: theoretical calculation, simulation and experimental verification.

We are analyzing a simple electrical circuit formed by three branches (Figure 6-5). The first branch consists of a voltage generator E and resistor R_1 , while the remaining two branches contain the resistors R_2 and R_3 wired as in the figure below. The task is to determine the electrical currents I_1 , I_2 and I_3 and to check the first Kirchoff law for the junction J. In order to solve the electrical circuit, the following parameters are given: $E=10\text{ V}$, $R_1=470\ \Omega$, $R_2=300\ \Omega$, $R_3=200\ \Omega$.

Your task is to calculate the electrical currents in each branch of the circuit by using the recommendations given in this exercise.

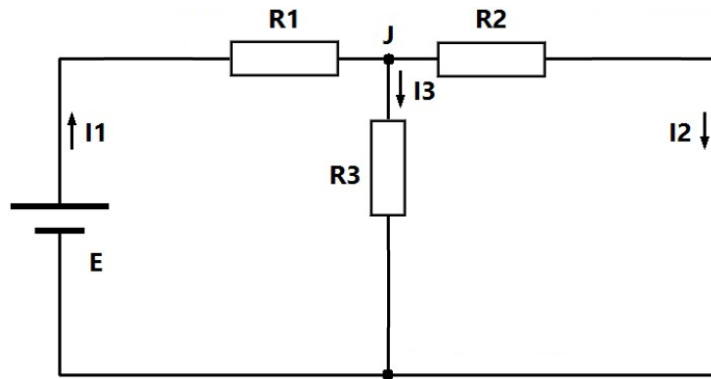


Figure 6-5. A simple electrical circuit for verification of the first Kirchhoff law.

$I_1 =$ _____

$I_2 =$ _____

$I_3 =$ _____

Question: what is the first Kirchhoff law for junction J?

Question: is the first Kirchhoff law fulfilled? If it is NOT fulfilled, check the calculations and solve the electrical circuit again.

6.3.1 Simulation of ideal electrical circuit

In this part of the exercise we perform a simulation of the ideal electrical circuit given in the previous section. Under "ideal" electrical circuit we assume a circuit where all elements have exact and time invariable parameters. The aim of the simulation is to check the theoretical calculations of the electrical currents I_1 , I_2 and I_3 . The simulation of the electrical circuit is realized with the RVILP. Run the application for the first Kirchhoff law, which is located under the "simulations" tab of the RVILP platform. At this point, a virtual instrument intended for the first Kirchhoff law will appear. The simulation of the electrical circuit is realized by selection of the tabulator "Calculation". The front panel of the virtual instrument is given in Figure 6-6.

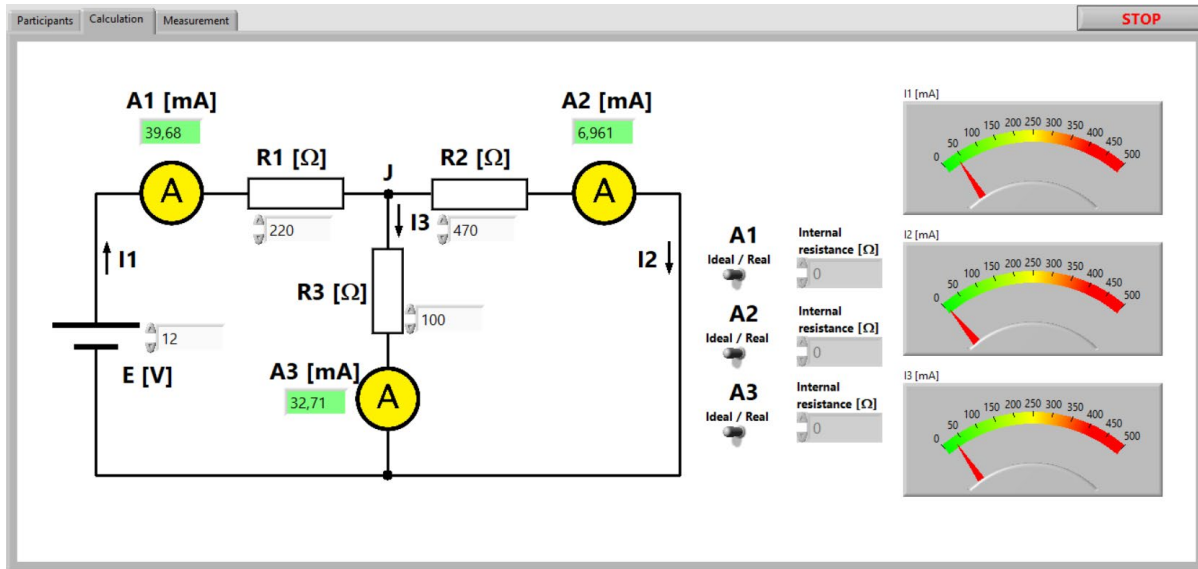


Figure 6-6. Front panel of the VI for simulation of the first Kirchoff law.

The electrical current is being measured with instrument called amperemeter. In this part of the exercise we assume that the amperemeter is an ideal instruments, i.e. its internal impedance is equal to zero. Hence, the electrical circuit will remain identical if we replace the amperemeters with wires (short circuit). The configuration of the ideal amperemeter (A1, A2 and A3) is realized with the controls "ideal/realistic", by turning them to the "ideal" position.

Each amperemeter is wired in series in the branch whose current is being measured. Hence, the amperemeter A1 is used to measure the electrical current I_1 , the amperemeter A2 for the current I_2 , and the amperemeter A3 for the current I_3 . The measured electrical currents are shown on the indicators A1, A2 and A3.

Your task is to simulate the electrical circuit from section 2 and determine the electrical currents in all branches.

The simulation of the electrical circuit is realized by entering the following parameters: $E=10\text{ V}$, $R_1=470\ \Omega$, $R_2=300\ \Omega$, $R_3=200\ \Omega$. Write the amperemeter readings fir the electrical currents I_1 , I_2 and I_3 :

$$I_1 =$$

$$I_2 =$$

$$I_3 =$$

Question: are the values from the simulation identical to those obtained by theoretical calculations in the previous section? If they are NOT, check the simulation settings and theoretical calculations all over again.

Question: what is the behavior of the electrical currents I_1 , I_2 and I_3 in the electrical circuit if the voltage source decreases (from 10 V to 5 V)?

6.3.2 Experimental setup

This part of the exercise is related to the practical realization of the electrical circuit and performing realistic measurements. The aim is to test the first Kirchoff law once again, but this time in realistic conditions. To perform the experiment we use the experimental board given in Figure 6-7.a, and its practical realization in Figure 6-7.b.

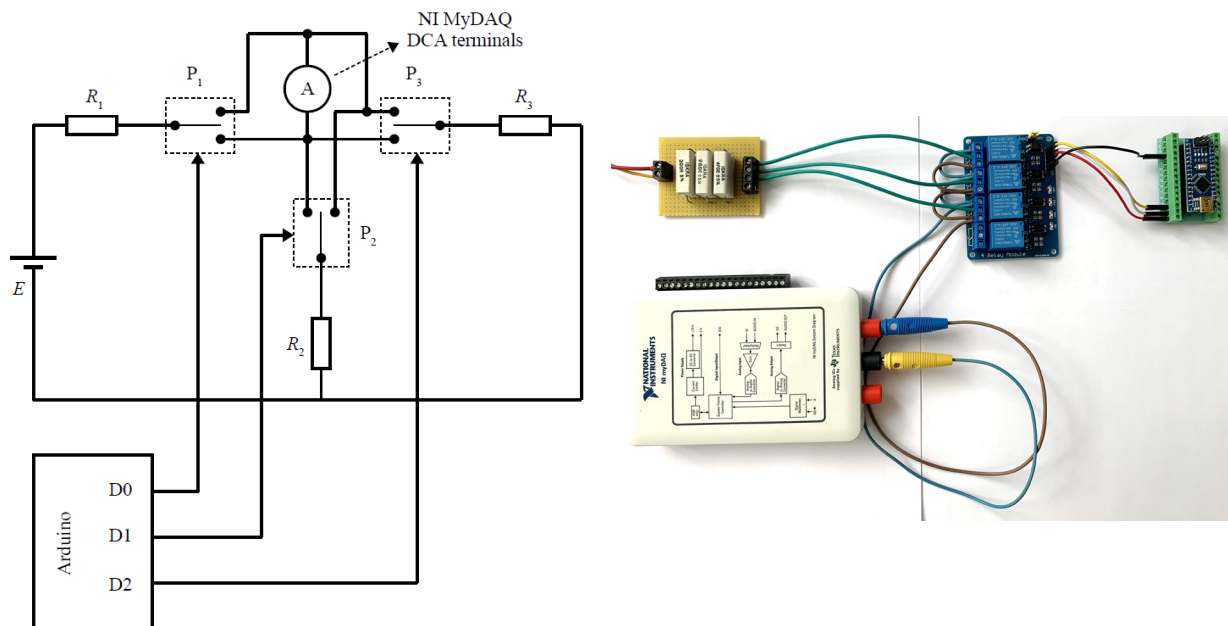


Figure 6-7. Experimental verification of the first Kirchoff law, a) Electrical circuit, and b) Practical realization.

Your task is to realize the electrical circuit given in Figure 6-7 and measure the electrical currents in all branches of the circuit.

The following hardware is used to perform the experiment:

- Experimental board
- Variable DC power supply
- Data Acquisition Card NI-myDAQ
- Arduino board
- Relay modules (P_1 - P_3)

We notice that the electrical circuit on the board is identical to those from the previous sections. The resistors $R_1=470\ \Omega$, $R_2=300\ \Omega$ and $R_3=200\ \Omega$ are integrated into the experimental board, while the voltage source and the amperemeters are externally connected. Actually, the experiment uses the current input channel of the NI MyDAQ card to measure direct current. The Arduino board controls which current is being measured by appropriate switching the relay modules (P_1 - P_3).

Procedure for realization of the electrical circuit and performing experimental measurements:

1. Note that the DC power supply is already connected to the experimenter board and it is adjusted to 10 V.
2. Activate the tabulator "Measurement" from the virtual instrument. The front panel from Figure 6-8 appears:

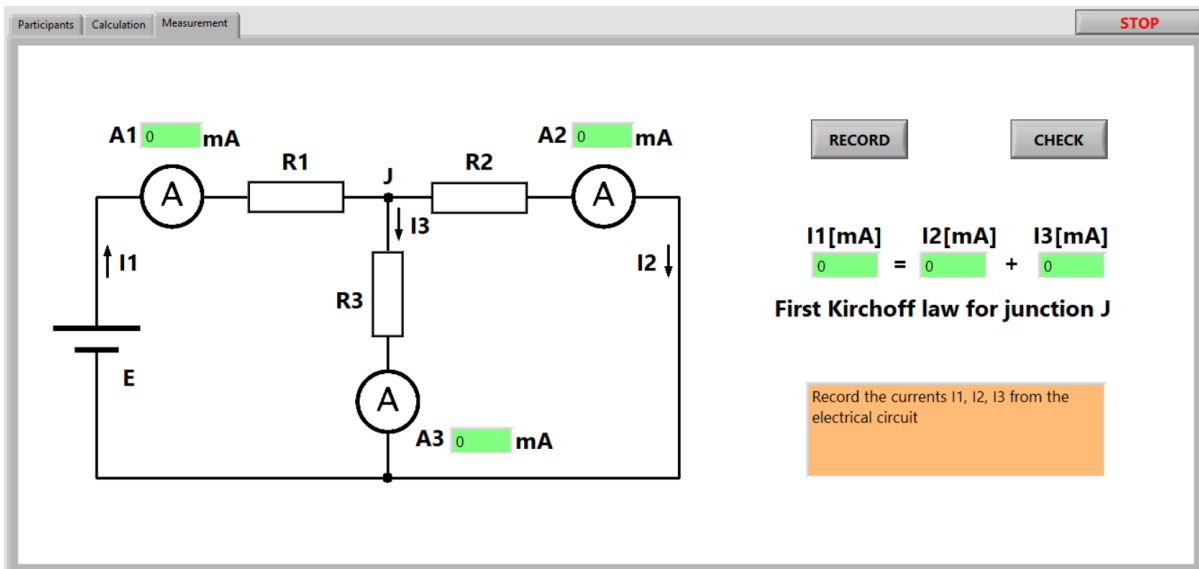


Figure 6-8. Front panel of the VI for experimental verification of the first Kirchof law.

3. Set the digital output ports of the Arduino board (D0-HIGH, D1-LOW, D2-LOW) through the Arduino VI of the RVILP platform and press the symbol of the amperemeter A1 (the symbol will turn yellow). Then, press the control button "RECORD" on the virtual instrument.

If this step is performed correctly, the measured current in the first branch will appear on the digital indicator I1[mA].

4. Repeat the step 3 for the amperemeters A2 и A3 for measurement of the electrical currents I_2 and I_3 .

Write down the measured values for the electrical currents by using the NI-myDAQ instrument:

$$I_1 =$$

$$I_2 =$$

$$I_3 =$$

5. To check the first Kirchoff law for the junction J press the control button "CHECK" on the virtual instrument.

The orange text indicator delivers a text message concerning the first Kirchoff law for the junction J. In case the first Kirchoff law is not fulfilled, the indicators are cleared and the experiment must be repeated again from the step 2.

6. Compare the measurements for the electrical currents with the NI-MYDAQ with the theoretical calculations and the simulations from the previous sections.

Question: do the measured values completely match the results from the theoretical calculations and the simulations?

Note: Examine using realistic instruments (enter the amperemeter internal resistance) in the "simulation" tab. Are the simulation results getting closer to the results from the practical measurements?

6.4 Laboratory experiment 3

In this laboratory exercise, we will explore the fundamental principles and practical aspects of measuring the transfer characteristic of a PIN photodiode, a crucial device widely used in optoelectronic applications. Photodiodes are semiconductor devices that convert light into an electrical current. Among the different types of photodiodes, the PIN photodiode stands out due to its unique structure and characteristics. The PIN photodiode consists of three distinct regions: a P-type semiconductor layer sandwiched between two N-type semiconductor layers. The "P" stands for positive, indicating an excess of positive charge carriers, while the "N" represents an excess of negative charge carriers.

When light strikes the depletion region, the region where the P and N layers meet, electron-hole pairs are generated. The electric field within the depletion region separates these charges, creating a photocurrent proportional to the incident light intensity. The transfer characteristic of a PIN photodiode describes the relationship between the output current and the input optical power or light intensity.

In this laboratory exercise, we will explore the experimental techniques and procedures necessary to measure the transfer characteristic of a PIN photodiode when using the signal conditioning

circuit given in Figure 6-9. We cover topics regarding the signal conditioning, data acquisition and linear approximation by using the least squares method. Additionally, the students can observe the key parameters and considerations that impact the transfer characteristic, including dark current, and linearity.

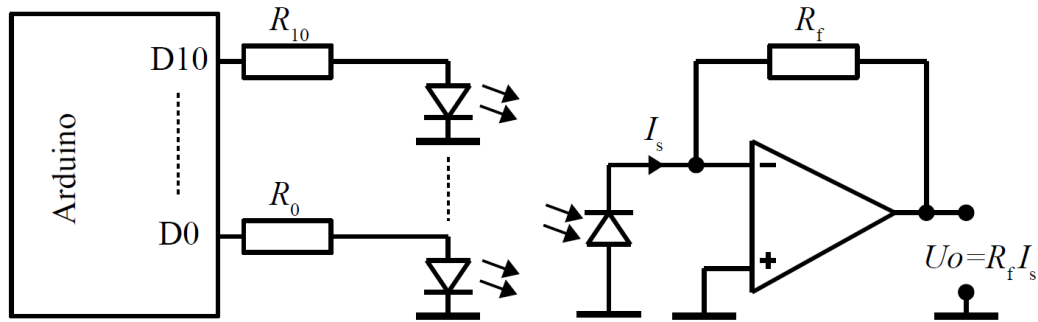


Figure 6-9. Experimental setup including the Arduino board, excitation LEDs, PIN photodiode, and signal conditioning circuit.

6.4.1 Experimental setup

In order to set up the experimental setup, the following equipment is required:

- Firstly, a particular PIN photodiode is needed in the visible spectrum of the light. This is a type of photodetector that operates by converting light into an electrical current. It will serve as the primary sensor for detecting light in the experiment.
- Next, an Arduino Uno or Nano board is necessary. This microcontroller board will act as a control and data acquisition module, responsible for receiving inputs from the photodiode and controlling the output of the LED diodes. The Arduino board will process the data from the photodiode and perform the required signal conditioning.
- Ten green LED diodes with a wavelength in the middle of the spectral range of the PIN photodiode (approximately 555 nm for visible region) are also required. These LEDs will be used to emit light at a specific wavelength, and particular illumination, which will be detected by the photodiode. The LEDs should be of the same type and specifications to ensure consistency in the experiment.
- To condition the signal from the photodiode, a signal conditioning circuit is needed. This circuit will process the electrical output from the photodiode to make it suitable for further analysis. It involves amplification in order to use the entire measurement range of the Arduino input analog to digital (AD) converters.
- Lastly, a Remote Virtual Instrumentation Platform (RVILP) is required. This platform provides a virtual interface or software that enables remote control and data acquisition from the experimental setup. It allows users to monitor and interact with the Arduino board, photodiode, and other connected devices remotely, providing a convenient way to collect and analyze data.

Experimental Procedure:

Step 1: Setting Up the Arduino and RVILP

Assemble the necessary equipment, including the Arduino board, PIN photodiode, green LED diodes, and signal conditioning circuit. Connect the Arduino board to your computer using a USB cable. Launch the RVILP software on your computer and establish a connection with the Arduino board through the RVILP platform (Figure 6-10).

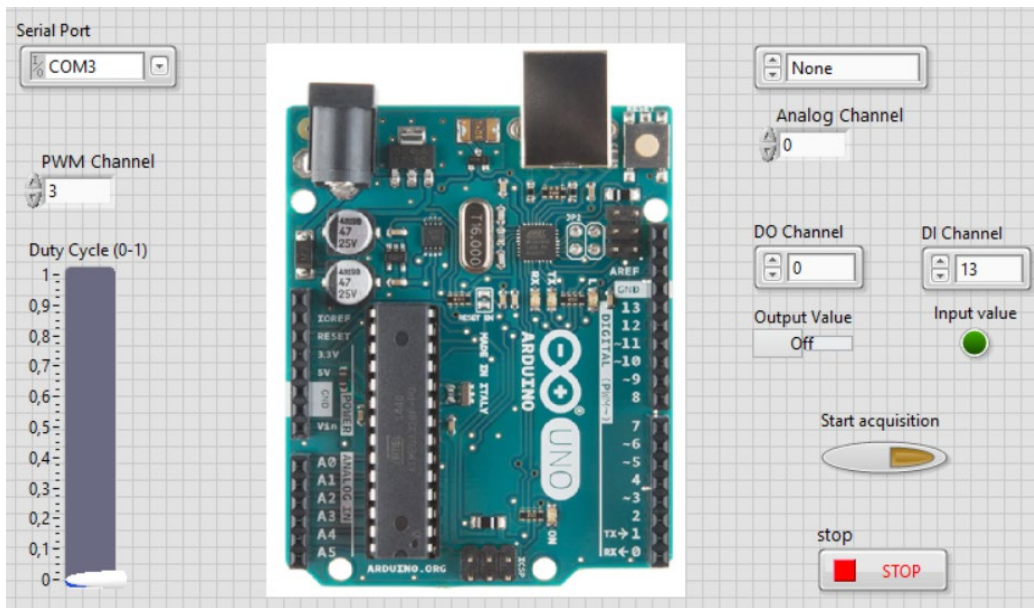


Figure 6-10. Front panel of the Arduino virtual instrument embedded into the RVILP platform.

Step 2: Configuring the LED Diodes

Connect the green LED diodes to the Arduino board, positioning them at a distance of 5 cm normal to the PIN photodiode. Utilize the RVILP software to control the Arduino board, allowing for the sequential switching on of the LED diodes.

Step 3: Signal Conditioning Circuit

Set up the signal conditioning circuit to ensure accurate measurement of the PIN photodiode's output voltage. Connect the output of the PIN photodiode to the input of the signal conditioning circuit. Then, connect the output of the signal conditioning circuit to one of the analog input ports of the Arduino board (e.g. A0).

Step 4: Conducting the Measurements

Ensure that the PIN photodiode is appropriately positioned and aligned with the LED diodes. Measure the output voltage of the PIN photodiode without any light illumination to determine the dark current. The measurements are performed by using a virtual voltmeter from the RIVLP platform. Use the RVILP software to control the Arduino board, causing the LED diodes to illuminate sequentially. Measure the output voltage of the signal conditioning circuit for each

illuminated LED diode. Record the corresponding LED diode number and the measured output voltage in a table. Repeat this process for all ten LED diodes, resulting in ten discrete points for the transfer characteristic.

6.4.2 Experimental results

Table 6-3. Transfer characteristics of the PIN photodiode and signal conditioning circuit.

No.	Illumination [%]*	OP amp output voltage [V]
1.	0	0.0
2.	10	0.21
3.	20	0.51
4.	30	0.81
5.	40	1.12
6.	50	1.42
7.	60	1.72
8.	70	2.03
9.	80	2.33
10.	90	2.73
11.	100	3.03

* The Illumination is expressed in percent normalized to the total Illumination from all 10 LEDs

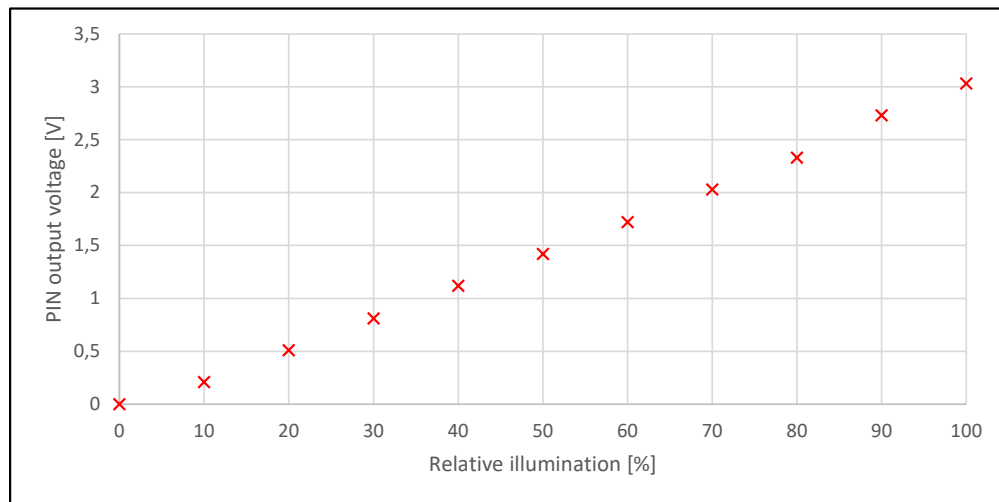


Figure 6-11. Transfer characteristics of the PIN photodiode.

6.5 Least squares approximation

The least squares approximation is a widely used method for finding the best-fitting curve or line that minimizes the overall squared distance between the observed data points and the predicted values from the model. It provides a robust and efficient way to estimate parameters and make predictions based on limited or noisy data.

In this laboratory practicum, we explore the principles and applications of least squares approximation through PIN photodiode hands-on experiments. As input parameters we are using the experimental results obtained from the measurements (given in Table 6-3 and Figure 6-11), whereas for the least squares approximation we are using a simulation virtual instrument from the RVILP. The virtual instruments and the obtained linear approximation coefficients are given in Figure 6-12.

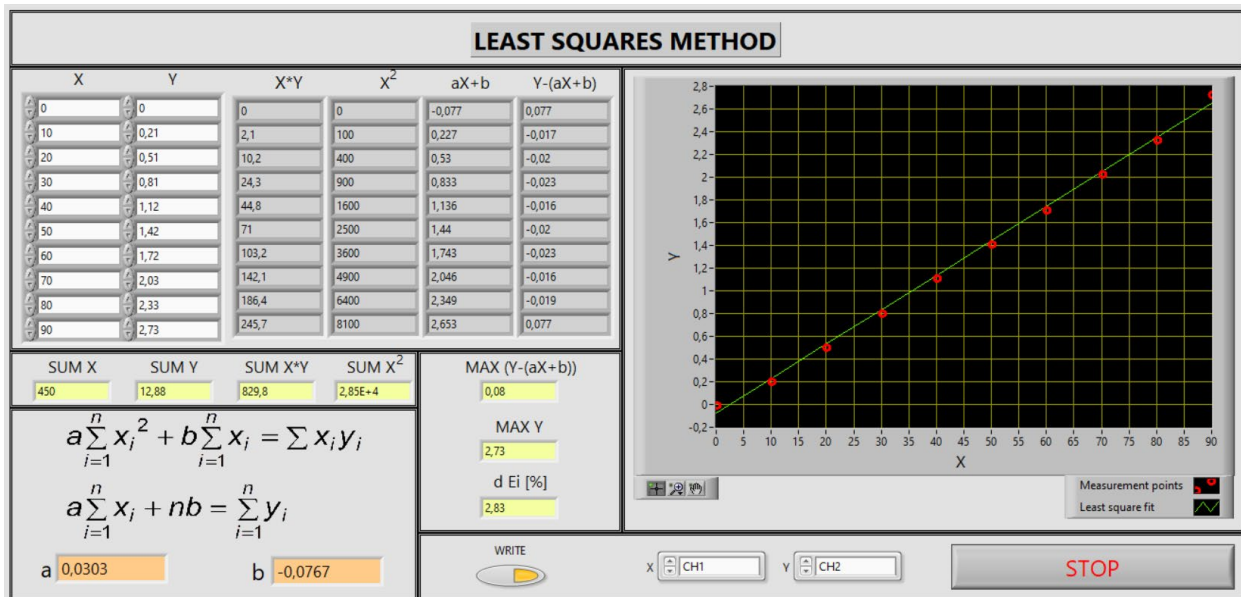


Figure 6-12. Virtual instrument for least squares approximation by using linear function.

6.6 Conclusion

Through these experiments, it can be seen that the Remote Virtual Instrumentation Laboratory Platform is a powerful and flexible environment for experiments and measurements using virtual versions of standard instrumentation, with easy and effective options for convenient remote access and control.

Laboratory manual

System Administration Laboratory

UbiLAB



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



Erasmus+



NATIONAL AGENCY
for European Educational
Programmes and Mobility

7 System Administration Laboratory

7.1 Introduction

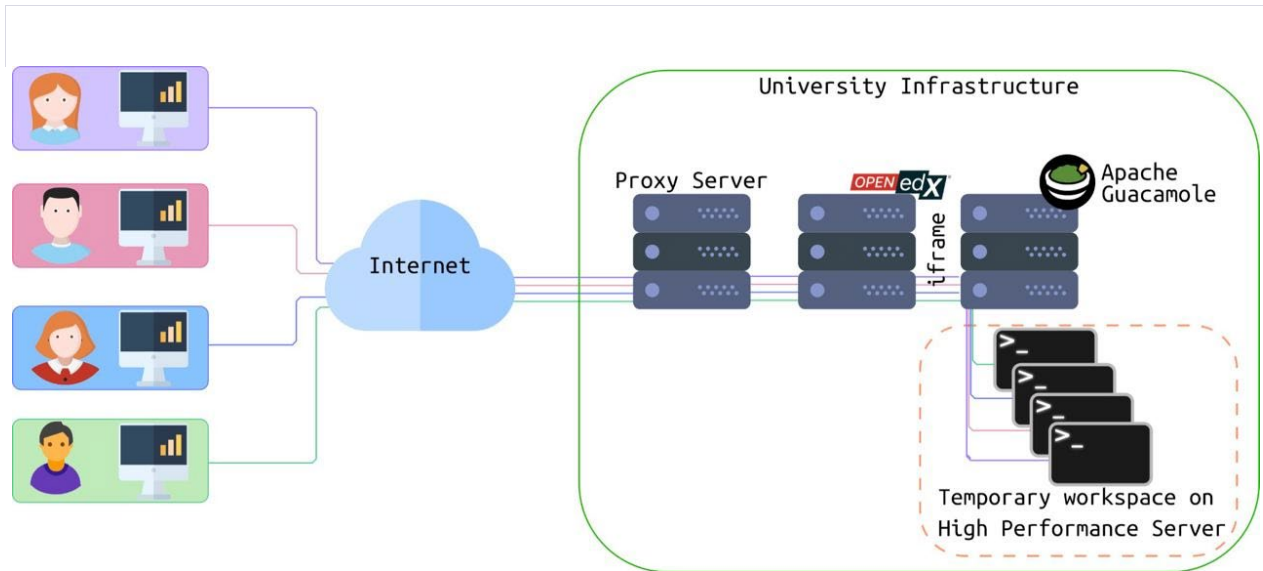


Figure 7-1. The architecture of remote access to the laboratory experiments.

The setup shown in Figure 7-1 that enables remote access to the laboratory experiments involves several stages of development and integration of various technologies. Here's a detailed description of each component:

1. **Apache Proxy Server:** The first line of communication with the outside world, the Apache proxy server, is the backbone of our setup. It's strategically placed between the internet and our internal network, serving as the border point. This configuration not only maintains our internal network's security but also manages and controls the flow of traffic. Any requests from users first hit this server, where they are verified and then routed to the appropriate service in our internal network.
2. **OpenEdx and Guacamole Services:** Both these services are deployed within the secure boundaries of our internal network and are responsible for the core functionality of our laboratory setup. OpenEdx is a feature-rich platform for online learning, serving course material, quizzes, and providing a user-friendly interface for learners to interact with the course content. Guacamole is used to facilitate the connection to our high-performance server. It provides a Virtual Network Computing (VNC) service that allows users to access the lab environment directly from their web browsers.

3. **iFrame Integration:** To provide a seamless user experience, Guacamole is integrated into OpenEdx using an iFrame. This allows the laboratory exercises to be loaded directly within the OpenEdx platform, offering a unified interface to the students. The students can access the lab exercises and OpenEdx course materials simultaneously, leading to a more interactive and efficient learning experience.
4. **High-Performance Server and Docker Containers:** The laboratory environment is hosted on a high-performance Linux server. The server uses Docker, a platform-as-a-service product that delivers software in packages known as containers. These containers are temporary instances that contain all the necessary elements to run the desired application, in this case, the laboratory exercises. Each student gets their Docker container, providing an isolated environment for their work. This ensures that students' experiments do not interfere with each other and can be scaled as per the needs.
5. **Apache Guacamole VNC:** Apache Guacamole's VNC is used to allow users to interact with their Docker containers. It provides a remote desktop interface, directly accessible from a web browser, eliminating the need for users to install any software or navigate complex network configurations.
6. **Automatic Exercise Checking Script:** The grading of exercises is automated through a script. Upon completion of an exercise, this script packages the student's work into a tar file and computes its MD5 checksum for verification purposes. The tar file is then removed, ensuring that only the checksum remains for assessment. This setup allows for efficient, scalable, and consistent grading of student work.
7. **Integration with Moodle and OpenEdx Platforms:** The entire laboratory setup is designed to connect seamlessly with Moodle and OpenEdx learning management systems. These integrations allow us to deliver laboratory exercises and feedback directly through these platforms, making it easier for students to access, complete, and receive feedback on their work. This setup provides a comprehensive learning experience to the students, right from accessing the course material to getting their exercises evaluated.

This setup, with a combination of Apache Proxy Server, OpenEdx, Guacamole, Docker, and integration with popular learning management systems, ensures a scalable, accessible, and interactive environment for students to carry out their laboratory experiments remotely. The focus on security, usability, and seamless integration provides a superior learning experience for students.

7.2 Gaining remote access

First time here? [Create an Account.](#)

Sign In

Email

The email address you used to register with My Open edX

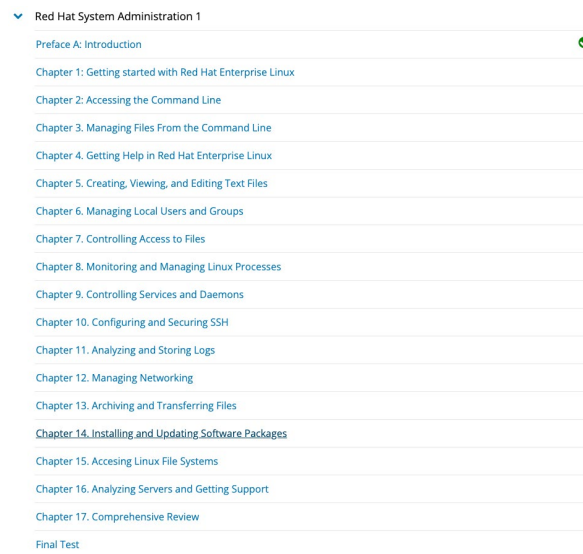
Password

[Need help logging in?](#)

Sign in

Figure 7-2. OpenEdx Login Screen: Enter your credentials to begin your learning journey.

1. Login: First, navigate to the OpenEdx platform using the domain name `openedx.fila-lab.de` in your web browser. You will be greeted by a login screen. Enter your credentials to proceed. If you don't have an account yet, you'll need to create one.



Red Hat System Administration 1	
Preface A: Introduction	✓
Chapter 1: Getting started with Red Hat Enterprise Linux	
Chapter 2: Accessing the Command Line	
Chapter 3: Managing Files From the Command Line	
Chapter 4: Getting Help in Red Hat Enterprise Linux	
Chapter 5: Creating, Viewing, and Editing Text Files	
Chapter 6: Managing Local Users and Groups	
Chapter 7: Controlling Access to Files	
Chapter 8: Monitoring and Managing Linux Processes	
Chapter 9: Controlling Services and Daemons	
Chapter 10: Configuring and Securing SSH	
Chapter 11: Analyzing and Storing Logs	
Chapter 12: Managing Networking	
Chapter 13: Archiving and Transferring Files	
Chapter 14: Installing and Updating Software Packages	
Chapter 15: Accessing Linux File Systems	
Chapter 16: Analyzing Servers and Getting Support	
Chapter 17: Comprehensive Review	
Final Test	

Figure 7-3. Course Dashboard: A wide range of courses are at your disposal. Select one to proceed.

2. Course Selection: After logging in, you will be directed to the main dashboard, where you can see all the available courses. Browse through the list and select the course you want to participate in.

Lab: Accessing the Command Line

Bookmark this page

Lab: Accessing the Command Line

Performance Checklist

In this lab, you will use the Bash shell to execute commands.

Outcomes

- Successfully run simple programs using the Bash shell command line.
- Execute commands used to identify file types and display parts of text files.
- Practice using some Bash command history "shortcuts" to more efficiently repeat commands or parts of commands.

Log in to **Apache Guacamole** as **student** using **student** as the password.

Log in to **workstation** as **praktikum** using **praktikum** as the password.

```
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-132-generic)
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Mon 30 Jan 2023 06:31:36 PM UTC

System load:                0.26
Usage of /:                  75.1% of 19.52GB
Memory usage:               72%
```

Figure 7-4: Interactive Laboratory Exercise: Perform your exercise in the embedded Linux environment through the Apache Guacamole VNC interface.

3. Sections and Subsections: After selecting a course, you will be presented with the course outline. This includes various sections and subsections, each representing different topics and subtopics within the course. Navigate through these sections and subsections to find the laboratory exercises.
4. Accessing Laboratory Exercises: The laboratory exercises are embedded within the course using an iframe. This iframe contains the Apache Guacamole VNC interface, which provides remote access to a Linux environment where the exercises are to be performed. Click on the specific laboratory exercise, and it will be loaded within the iframe.

Please remember to save your work regularly while performing the exercises to avoid data loss. Once you have completed your exercise, it will be automatically graded and the feedback will be available on the OpenEdx platform. Enjoy your learning experience!

7.3 Laboratory experiment 1: Accessing the Command Line

This laboratory experiment introduces students to the Bash shell, a popular command-line interpreter for Unix-based systems. In the scope of this lab, students will execute a variety of commands, manipulate text files, and experience the efficiency of Bash command history shortcuts.

7.3.1 Experimental setup

The experiment is conducted on a Linux environment, accessed remotely via Apache Guacamole, integrated into the OpenEdx platform. Students log in to the environment as 'praktikum' with the same as the password, allowing them to execute commands in the Bash shell.

7.3.2 Goals of the experiment

The objectives of this experiment are to:

1. Familiarize students with the Bash shell command line.
2. Execute commands to identify file types and display parts of text files.
3. Learn and practice using Bash command history shortcuts for command repetition and modification.

7.3.3 Experimental results

Upon completion of this experiment, students will gain:

1. A clear understanding of how to navigate the Bash shell command line.
2. Experience with executing commands to manipulate and inspect files.
3. Proficiency in using Bash command history shortcuts to repeat and modify previous commands.
4. After each step, students can input their command results into the lab's interactive grader for immediate feedback and validation.

7.3.4 Task

Step 1: Login

Firstly, log in to Apache Guacamole using the credentials: username - 'student', password - 'student'. Then, log in to the workstation with the credentials: username - 'praktikum', password - 'praktikum'.

Step 2: Display Current Date and Time

Execute the following command in the Bash shell to display the current time and date:

```
date
```

Step 3: Display Current Time in 12-Hour Clock Format

To display the current time in a 12-hour clock format, use the following command:

```
date +%r
```

Step 4: Identify File Type

Determine the file type of /home/praktikum/zcat and whether it is readable by humans using:

```
file /home/praktikum/zcat
```

Step 5: Display the Size of the File

Use the wc command along with Bash shortcuts to display the size of zcat:

```
wc -c /home/praktikum/zcat
```

Step 6: Display the First 10 Lines of the File

To display the first 10 lines of zcat, execute:

```
head /home/praktikum/zcat
```

Step 7: Display the Last 10 Lines of the File

Display the last 10 lines of the zcat file using the following command:

```
tail /home/praktikum/zcat
```

Step 8: Repeat the Previous Command

To repeat the previous command exactly with three or fewer keystrokes, use Bash history command:

```
!!
```

Step 9: Display the Last 20 Lines of the File

Edit the previous command to display the last 20 lines of the file with a minimal number of keystrokes:

```
!-1 -n 20
```

Step 10: Run the date +%r Command Again

Use the shell history to run the date +%r command again:

```
!date
```

Please remember, after each step, to input your command results into the fields provided for verification. This will help you ensure the correctness of your work.

Remember to follow each step carefully and good luck with your experiment!

7.4 Laboratory experiment 2: Managing Files from the Command Line

In this lab, you will efficiently create, move, and remove files and directories by using the shell and a variety of file name matching techniques.

7.4.1 Experimental setup

1. Log into Apache Guacamole as student using 'student' as the password.
2. Log into the workstation as 'praktikum' using 'praktikum' as the password.

7.4.2 Goals of the experiment

The aim of this experiment is to:

1. Familiarize you with file and directory manipulation from the command line.
2. Teach you to use wildcards for locating and handling files.

7.4.3 Task

1. Before you create project files, use the `mkdir` command with brace expansion to create empty project planning documents in the `/tmp/answer/Documents/project_plans` directory. (Hint: if `/tmp/answer/Documents` does not exist, the `-p` option for the `mkdir` command will create it.) Create two empty files in the `/tmp/answer/Documents/project_plans` directory: `season1_project_plan.odf` and `season2_project_plan.odf`.
2. Create sets of empty practice files to use in this lab. If you do not immediately recognize the intended shell expansion shortcut, use the solution to learn and practice. Use shell tab completion to locate file path names easily. Create a total of 12 files with names `tv_seasonX_episodeY.ogg`. Replace X with the season number and Y with that season's episode, for two seasons of six episodes each.

3. As the author of a successful series of mystery novels, your next bestseller's chapters are being edited for publishing. Create a total of eight files with names `mystery_chapterX`. Replace `X` with the numbers 1 through 8.
4. Use a single command to create two subdirectories named `season1` and `season2` under the `Videos` directory, to organize the TV episodes.
5. To check the task `[student@workstation ~]$ bash /tmp/check.sh /tmp/answer/`
6. Paste the output of the script into the text box
7. Move the appropriate TV episodes into the season subdirectories. Use only two commands, specifying destinations using relative syntax.
8. Create a 2-level directory hierarchy with a single command to organize the mystery book chapters. Create `my_bestseller` under the `Documents` directory, and `chapters` under the new `my_bestseller` directory.
9. Create three more subdirectories directly under the `my_bestseller` directory using a single command. Name these subdirectories `editor`, `changes`, and `vacation`. The `-p` option (create parents) is not needed because the `my_bestseller` parent directory already exists.
10. Change to the `chapters` directory. Move all book chapters to the `chapters` directory, which is now your current directory. What is the simplest syntax to specify the destination directory?
11. To check the task: `bash /tmp/check.sh /tmp/answer/`
12. Paste the output of the script into the text box
13. You sent the first two chapters to the editor for review. Move only those two chapters to the `editor` directory to avoid modifying them during the review. Starting from the `chapters` subdirectory, use brace expansion with a range to specify the chapter file names to move and a relative path for the destination directory.
14. While on vacation you intend to write chapters 7 and 8. Use a single command to move the files from the `chapters` directory to the `vacation` directory. Specify the chapter file names using brace expansion with a list of strings and without using wildcard characters.
15. Change your working directory to `/tmp/answer/Videos/season2`, and then copy the first episode of the season to the `vacation` directory.
16. Use a single `cd` command to change from your working directory to the `/tmp/answer/Documents/my_bestseller/vacation` directory. List its files. Use the previous working directory argument to return to the `season2` directory. (This will succeed if the last directory change with the `cd` command was accomplished with one command rather than

several `cd` commands.) From the `season2` directory, copy the episode 2 file into the vacation directory. Use the shortcut again to return to the vacation directory

17. To check the task: `bash /tmp/check.sh /tmp/answer/`
18. Paste the output of the script into the text box
19. The authors of chapters 5 and 6 want to experiment with possible changes. Copy both files from the `/tmp/answer/Documents/my_bestseller/chapters` directory to the `/tmp/answer/Documents/my_bestseller/changes` directory to prevent these changes from modifying original files. Navigate to the `/tmp/answer/Documents/my_bestseller` directory. Use square-bracket pattern matching to specify which chapter numbers to match in the filename argument of the `cp` command.
20. Change your current directory to the `changes` directory. Use the `date +%F` command with command substitution to copy `mystery_chapter5` to a new file which includes the full date. The name should have the form `mystery_chapter5_YYYY-MM-DD`. Make another copy of `mystery_chapter5`, appending the current time stamp (as the number of seconds since the epoch, 1970-01-01 00:00 UTC) to ensure a unique file name. Use command substitution with the `date +%s` command to accomplish this.
21. After further review, you decide that the plot changes are not necessary. Delete the `changes` directory. If necessary, navigate to the `changes` directory and delete all the files within the directory. You cannot delete a directory while it is the current working directory. Change to the parent directory of the `changes` directory. Try to delete the empty directory using the `rm` command without the `-r` recursive option. This attempt should fail. Finally, use the `rmdir` command to delete the empty directory, which will succeed.
22. When the vacation is over, the vacation directory is no longer needed. Delete it using the `rm` command with the recursive option. Return to `/tmp/answer` directory.
23. Create a hard link to the `/tmp/answer/Documents/project_plans/season2_project_plan.odf` file named `/tmp/answer/Documents/backups/season2_project_plan.odf.back`. A hard link will protect against accidental deletion of the original file and will keep the backup file updated as changes are made to the original.
24. On workstation, run the `check.sh` script to confirm success on this lab. `$ cd /tmp/ ; bash check.sh answer/`
25. Paste the output of the script into the text box

7.4.4 Experimental results

You should be able to use wildcards to locate and manipulate files.

7.5 Laboratory experiment 3: Editing Files Using Vim's Visual Mode

In this lab, you'll learn how to simplify repetitive edits using Vim's visual mode. You'll manipulate a file containing a list of selected files and tabular data, which will help you become more familiar with the required utilities and techniques for file editing.

7.5.1 Experimental setup

Log into the server as 'student' and begin in the student's home directory.

7.5.2 Goals of the experiment

The objective of this experiment is to enhance your skills in:

1. Redirecting output to a file or program.
2. Editing text files from the shell prompt using Vim.
3. Editing text files with a graphical editor.

7.5.3 Experimental results

At the completion of this lab experiment, you should have achieved the following outcomes:

1. File Redirection: You should have successfully redirected the long listing of all content in the student's home directory into a file named `editing_final_lab.txt`. This redirection helps in logging the directory structure for future reference or for tracking changes over time.
2. Vim Editing: You should be able to demonstrate your proficiency in using Vim's visual mode. This was achieved by editing `editing_final_lab.txt` and manipulating lines and columns within the file. Your skills in using the line-based, single-line, and block selection modes in Vim should have been honed.
3. File Manipulation: By removing specific lines and columns from the file, you have demonstrated an understanding of how to selectively manipulate file content. This includes modifying file permissions and removing certain rows.
4. File Backup and Emailing: You created a backup of your file using a timestamp to ensure a unique filename. You then emailed the contents of this file to 'student'. This showcases your ability to safeguard your work and to share your work via email.

5. File and Process Appending: You added a dashed line to the file to denote the beginning of new content. Subsequently, you appended a process listing to the file and validated its presence at the end of the file. This demonstrates your ability to dynamically add content to a file, including system process information.

Overall, these outcomes signify that you've acquired valuable skills in redirecting output to a file, editing text files from the shell prompt using Vim, and using an editor in a graphical desktop environment to modify file content.

7.5.4 Task

Log in to workstation as student using student as the password.

On workstation, run the edit-review start command.

```
[student@workstation ~]$ lab edit-review start
```

1. Redirect a long listing of all content in the student's home directory, including hidden directories and files, into a file named `editing_final_lab.txt`.
2. Edit the file using Vim.
3. Remove the first three lines. Enter line-based visual mode with uppercase V.
4. Remove columns on the first line. Enter visual mode with lowercase v. Lowercase v selects characters on a single line only. The columns after `-rw-` should be deleted.
5. Remove columns, and the subsequent dot (".") on the remaining lines. Use the visual block mode. Enter visual block with the control sequence `Ctrl+V`. Use this key sequence to select a block of characters on multiple lines. The columns after `-rw-` should be deleted.
6. Use visual block mode to remove the fourth column.
7. Use visual block mode to remove the time column, leaving the month and day on all lines.
8. Remove the Desktop and Public rows. Enter visual line mode with uppercase V.
9. Use the `:wq` command to save and exit the file. Make a backup, using the date (in seconds) to create a unique file name.
10. Append a dashed line to the file. The dashed line should contain at least 12 dashes
11. Append a directory listing of the Documents directory. List the directory listing on the terminal and send it to the `editing_final_lab.txt` file with one command line.
12. Confirm that the directory listing is at the bottom of the lab file.

7.6 Conclusion

In conclusion, this lab exercise provided a hands-on experience with fundamental operations in Unix/Linux command line environments, including redirecting output to a file, editing text files using Vim, and utilizing a graphical editor for altering file content.

The practice of using Vim's visual modes allowed for enhanced understanding and skill development in manipulating file data effectively. The selective removal of lines and columns within a file served as an example of how file content could be manipulated for specific needs, such as refining file permissions or modifying specific rows.

Moreover, the experiment emphasized the importance of good file management practices, particularly in creating backups and ensuring files are shared securely via email.

Appending a process listing to a file demonstrated how system process information could be logged and accessed, underlining the versatility of Unix/Linux commands in gathering system insights.

Ultimately, the lab has equipped the participants with essential command line skills that will be beneficial in a variety of scenarios, particularly in system administration, software development, and data management. These skills are not only crucial in dealing with Unix/Linux environments but also are transferrable to other computing contexts, which makes them valuable in today's increasingly digital and data-driven world.

Laboratory Manual

Automation System Control Laboratory

- Dual temperature control system
- Closed loop control of dual temperature control system
- Wind levitation system

UbiLAB



Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ



Erasmus+



NATIONAL AGENCY
for European Educational
Programmes and Mobility

8 Automation System Control Laboratory

8.1 Introduction

The laboratory experiments are intended for students to learn basic and advanced feedback theory concepts. The exercises cover basic feedback design techniques with graphical tools and modern design approaches with more complex and sophisticated tools. All the experiments can be executed remotely over the web graphic user interface. The developed web interface based on HTML and JavaScript can be easily portable to the different MOOC platforms and used for distance learning.

This document describes the exercises on two different setups, where the goal is to design feedback control algorithms and perform activities remotely. The basis of both experiments is a connection bridge developed on the ARM32F7 controllers. The main board has two possible links to share the data over the network. The first link is based on serial communication, where the data is collected from the experiment and provided to the server, which ports the data to the web. A second connection is based on the LWIP stack, which enables a direct connection over the TCP/IP protocol. All data can be provided over both connection links and are managed to speed up the package transportation and multi-platform connection.

The main board of the experiment collects the data from the existing system and processes the data from the sensors. Safety and connectivity are ensured with a multilevel watchdog timer, which distinguishes between non-activity from the user and safety protocols to maintain the system's safe operation. If the system detects the user's absence of network activity, restart the connection and re-establish a new link to the other user. The safety protocols of the experiment activity maintain the proper functionality and avoid unwanted malfunctions in the system.

The exercises are divided into three subsections, where each previous experiments are starting point for the next practice. All the activities are closely related to the feedback control theory. Feedback control theory is a branch of engineering and mathematics that analyzes and designs systems governed by feedback loops. It provides a framework for understanding and manipulating the behaviour of dynamic systems to achieve desired objectives. Feedback control theory aims to design controllers that stabilize the system, reject disturbances, and achieve desired performance criteria such as stability, accuracy, speed, and robustness. The design process typically involves mathematical modelling of the system, analyzing its behaviour, and applying control techniques to achieve the desired objectives. Feedback control theory finds applications in various fields, including robotics, aerospace, automotive, process control, and electronics. It enables engineers to design control systems that can effectively control and optimize the behaviour of dynamic systems.

8.2 Laboratory experiment 1 - Dual Temperature Control System

The laboratory experiment is based on a dual heater system. A dual temperature control system comprises two power transistors and two temperature sensors. A dual temperature control system is a specific feedback control system that controls two different temperatures simultaneously. It is commonly employed in applications where maintaining two different temperatures is critical, such as in industrial processes, environmental control systems, or HVAC (Heating, Ventilation, and Air Conditioning) systems.

A dual temperature control system typically has two separate control loops, each responsible for regulating one temperature. Designing and tuning a dual temperature control system involves setting appropriate control parameters for each temperature loop to achieve stable and accurate temperature regulation. It also requires considering any interactions or cross-couplings between the two control loops to ensure proper coordination and prevent interference.

The dual temperature control system can provide precise and independent control over two different temperatures, allowing optimal operation and energy efficiency in applications where maintaining specific temperature conditions is crucial.

The essential components of the system include:

- **Sensors:** Two temperature sensors measure the temperatures of the two different zones or processes. These sensors provide feedback signals to the control system.
- **Controllers:** Two individual controllers are employed to process the feedback signals and generate the appropriate control actions for each temperature. Each controller compares the measured temperature with a desired setpoint and calculates the necessary control output.
- **Actuators:** Two actuators adjust the inputs to control each temperature separately. These actuators could be heating elements, cooling units, valves, or other devices that can manipulate the temperature.

8.2.1 Experimental setup

A dual temperature control system is developed as a small MIMO system, which is portable and easy to use. The system does not require additional laboratory equipment. For the heaters, two power transistors, TIP31 are used. The resistor does not limit the current to the base port of the transistor, and a larger current is provided through the collector and emitter, therefore the heat dissipation is increased. On each power transistor, the LM60 temperature sensor is attached. The sensor has an analogue output, and the temperature scaling is processed on the microcontroller board. The main three components of the system are:

- **Sensors:** Two analogue sensors LM60.
- **Controllers:** The controller is implemented in the main timer-interrupt routine with a time tap of 1 second.
- **Actuators:** Two power transistors TIP31 driven with the PWM signal from the microcontroller board.

The experiment is presented in Figure 8-1, and the block schematic of the dual temperature control system is given in Figure 8-2.

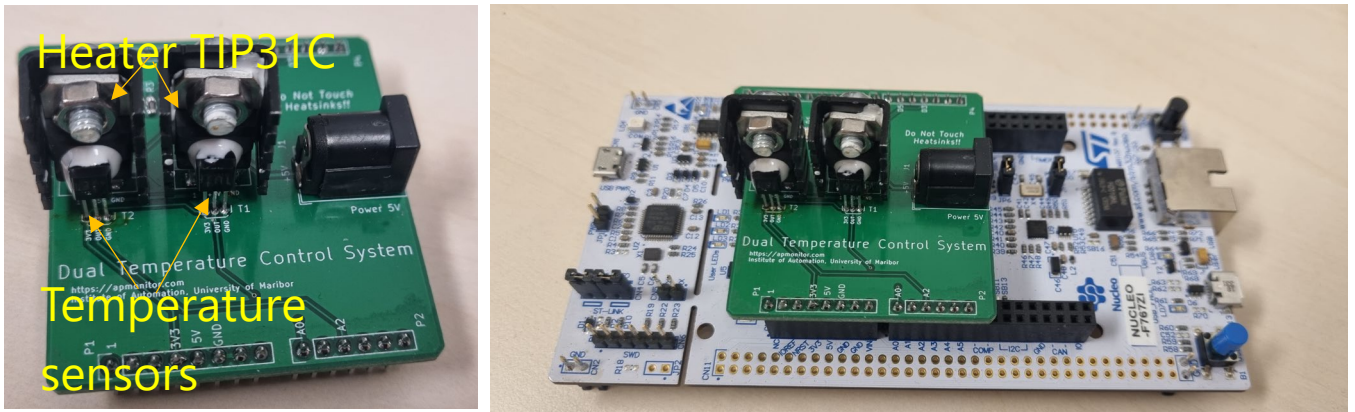
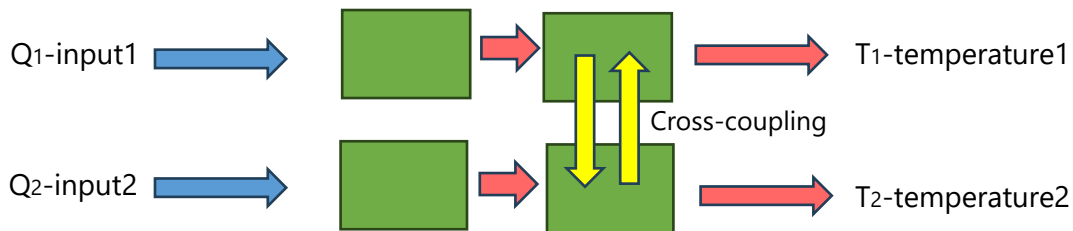


Figure 8-1. A dual temperature control system main board with microcontroller Nucleo F767ZI.



8.2.2 Goals of the experiment

Figure 8-2. Block schematic of a dual temperature control system.

Identify the parameters of the mathematical model. Identifying the parameters of the model is essential for the control design procedure. The mathematical modelling is based on the physics relation given below. The energy balance equation is given as:

$$m c_p \frac{dT}{dt} = \sum \dot{h}_{in} - \sum \dot{h}_{out} + Q$$

For the first input $Q_1 \rightarrow T_1$,

$$m c_p \frac{dT_1}{dt} = U A (T_\infty - T_1) + \epsilon \sigma A (T_\infty^4 - T_1^4) + Q_{C12} + Q_{R12} + \alpha_1 Q_1$$

with the coupling parameters

$$Q_{C12} + Q_{R12} + \alpha_1 Q_1$$

For the second input $Q_2 \rightarrow T_2$,

$$m c_p \frac{dT_2}{dt} = U A (T_\infty - T_2) + \epsilon \sigma A (T_\infty^4 - T_2^4) - Q_{C12} - Q_{R12} + \alpha_2 Q_2$$

with the coupling parameters

$$- Q_{C12} - Q_{R12} + \alpha_2 Q_2$$

The equations are complex, and most parameters are hard to determine. Most of the equation elements have a minor influence on the final temperature value and can be neglected. In this case, the approximation of the first-order transfer function is used.

$$H(s) = \frac{T_1(s)}{Q_1(s)} = \frac{k}{\tau s + 1}$$

Where τ and k are time constant and gain separately defined as,

$$\tau = t(0.63 T_1(\infty))$$

$$k = \frac{T_1(\infty)}{Q_1(\infty)}$$

As presented in Figure 8-3, both parameters can be assigned from the system step response.

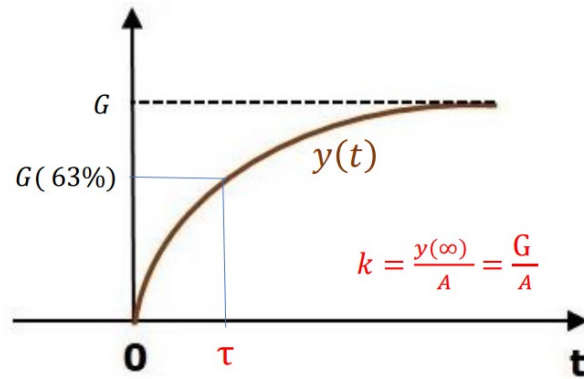


Figure 8-3. Step response of the system.

8.2.3 Experimental results

In the web-GUI (Figure 8-4), adjust the open loop slider to level 40 and read the step response of the system (given in Figure 8-5 for $Q_1 \rightarrow T_1$).

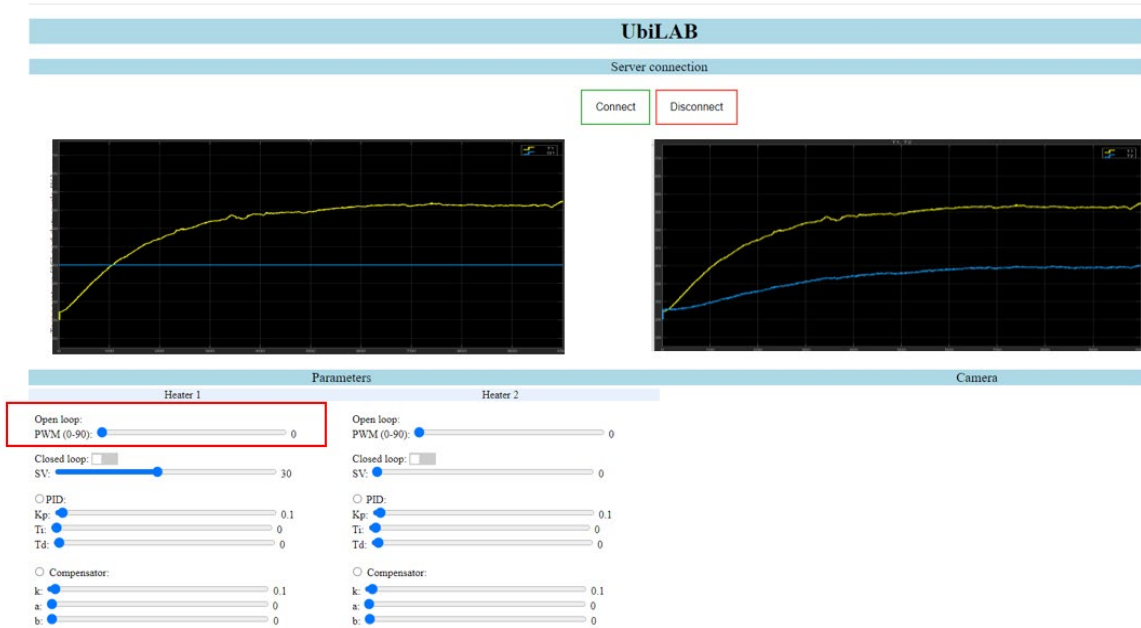


Figure 8-4. Web-GUI for Dual temperature control system.

Step response of the real system on channel one ($Q_1 \rightarrow T_1$) close view,

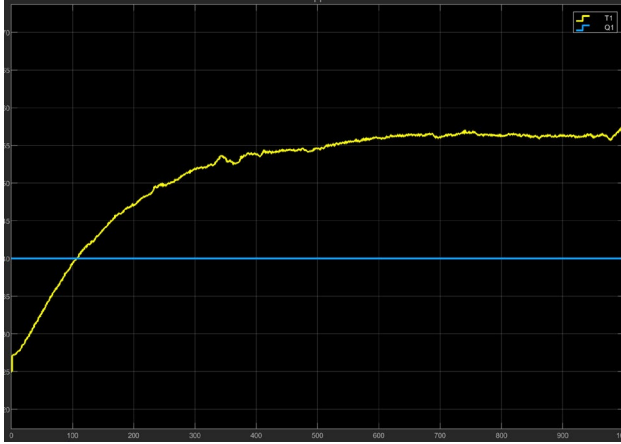


Figure 8-5. Step response of the system Q1->T1.

Assigned data from the step response in Figure 8-5:

$$k = 0.82$$

$$\tau = 165.2s$$

$$H(s) = \frac{T_1(s)}{Q_1(s)} = \frac{0.85}{165.2s + 1} = \frac{0.0049}{s + 0.0061}$$

Model validation is presented in Figure 8-6, close view.

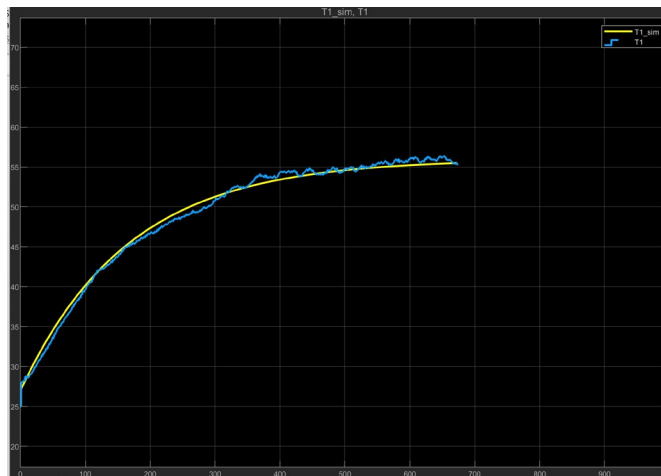


Figure 8-6. Model validation Q1->T1. Blue line: a real measurement, yellow line: simulated model.

8.3 Laboratory experiment 2 - Closed-loop Control

The laboratory experiment aims to design a feedback structure for the dual temperature heater system described in experiment 1.

8.3.1 Experimental setup

In experiment 2, the dual temperature system is used. In this case, the negative feedback (Figure 8-7) is employed.

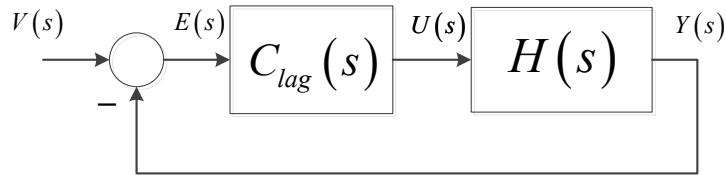


Figure 8-7. Closed-loop system.

8.3.2 Goals of the experiment

The experiment aims to design a feedback control loop with a lag compensator. A lag compensator is a feedback control system designed to improve the stability and performance of a system. It is often used to shape the frequency response of a system to achieve desired characteristics such as better transient response, reduced overshoot, and improved stability margins. The purpose of a lag compensator is to introduce additional phase lag to the system's open-loop transfer function, thereby reducing the system's gain at high frequencies while maintaining or increasing the gain at low frequencies. This helps to mitigate issues like instability or poor response time that may occur in the original system. The transfer function of the lag compensator is,

$$C_{lag}(s) = \frac{E(s)}{U(s)} = \frac{g(s+b)}{(s+a)} \quad g, b, a > 0 \wedge b > a$$

Here g , b , a are compensator gain, zero, and pole, respectively. Designing a lag compensator involves analysing the system's open-loop transfer function and evaluating the desired stability and performance criteria.

8.3.3 Experimental results

The lag compensator regarding the system transfer (exercise 1) function is designed to compensate the pole of the system. The compensator pole located close to the origin achieves transient response behaviour and tracking capability.

The transfer function of the system,

$$H(s) = \frac{T_1(s)}{Q_1(s)} = \frac{0.85}{169.2s + 1}$$

The root locus of the transfer function $H(s)$ is presented in Figure 8-8.

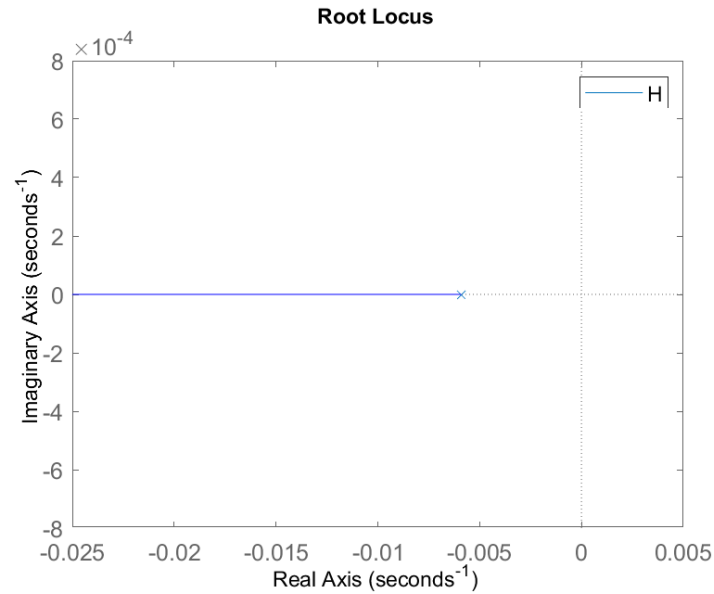


Figure 8-8. Root locus of the transfer function $H(s)$.

Preselect compensator regarding latter design guidelines (Figure 8-9).

$$C_{lag}(s) = \frac{E(s)}{U(s)} = \frac{4.2\left(s + \frac{1}{169.2}\right)}{(s + 0.001)}$$

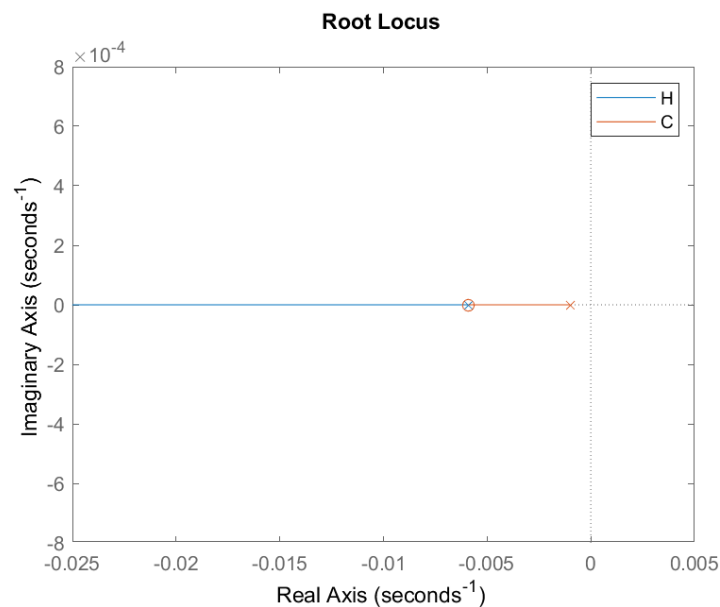


Figure 8-9. Root locus of the transfer function $H(s)$ with lag compensator $C_{lag}(s)$.

Experimental results are given in Figure 8-10, for selected compensator and reference values.

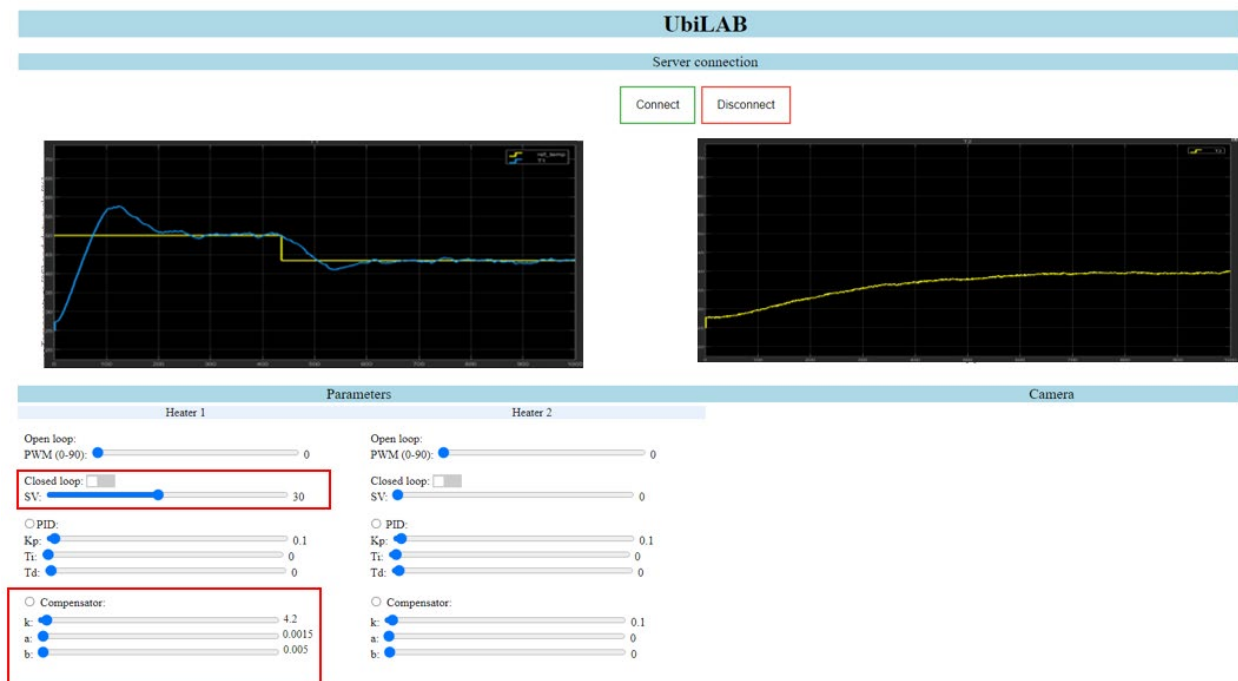


Figure 8-10. Real-time experiment with lag compensator, web-GUI.

8.4 Laboratory experiment 3 - Wind Levitation System

The aim of experiment 3 is to design the feedback structure that stabilizes the floater in the wind tube at a certain height. Height stabilization in the context of a wind tube typically refers to a device or mechanism designed to maintain stability and control the position of an object within the wind tunnel. In wind tunnel testing, a wind tube is used to simulate the effects of airflow on an object or model under controlled conditions. The object or model is typically placed within the wind tunnel, and the air is blown through the tunnel at various speeds to study the dynamic behaviour of the object. The specific design and implementation of a stabilizing floater in a wind tunnel can vary depending on the object being tested, the desired level of control, and the capabilities of the wind tunnel facility.

8.4.1 Experimental setup

The Wind levitation system experiment comprises the wind tube floater and the fan. The fan is placed under the wind tube with an attached sensor to measure the accurate height of the floater. Figure 8-11 presents the experiment system.

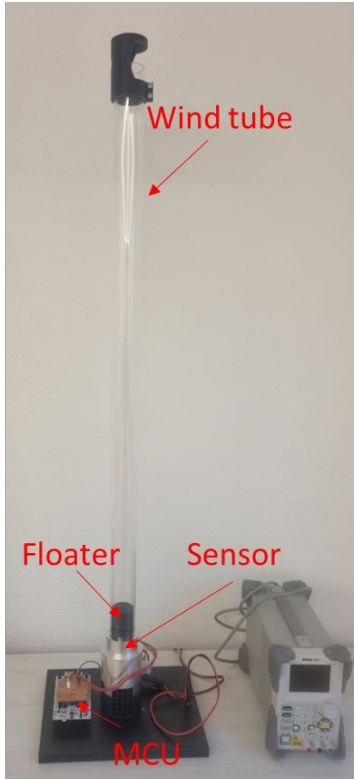


Figure 8-11. Wind levitation experiment system.

All the measurements, data processing, and communication task proceeded on board NUCLE767ZI. The communication protocol is the same as in the previous Dual temperature control system experiment. The main three components of the system are:

- **Sensors:** Time to Flight (ToF) sensor VL53L0X.
- **Controllers:** The controller is implemented in the main timer-interrupt routine with a time tap of 1 second.
- **Actuators:** Fan EDF Ducated 70mm, 3000Kv brushless motor.

The hardware structure and connection between the system are presented in Figure 8-12.

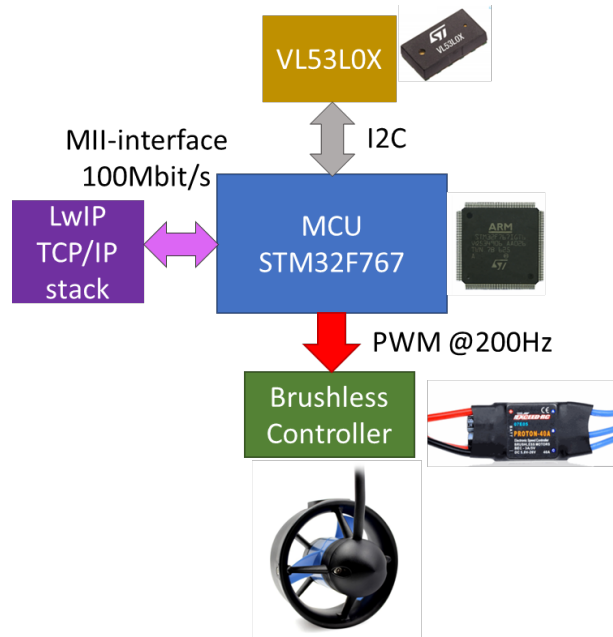
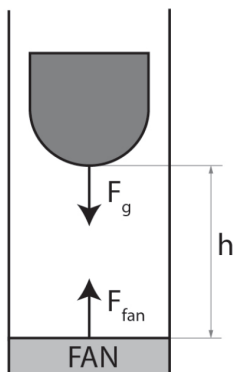


Figure 8-12. Hardware connection of the Wind levitation system.

8.4.2 Goals of the experiment

Design a PID controller for stabilizing the floater at desired height inside the wind tube. Use the following mathematical model.



Second Newton law of floater motion,

$$m\ddot{h} = -mg + F_{fan}$$

Second-order differential equation,

$$\ddot{h} = -g + \frac{1}{m}F_{fan}$$

The transfer function of the system is,

$$H(s) = -g + \frac{H_{height}(s)}{F_{fan}(s)} = -g + \frac{1}{ms^2},$$

$$\frac{H_{height}(s)}{F_{fan}(s)} = \frac{1}{ms^2}$$

The PID controller is designed for the transfer function $\frac{H_{height}(s)}{F_{fan}(s)}$. In the feedback structure presented in Figure 8-13, the gravity constant g can be compensated with additional input at the output of the PID controller.

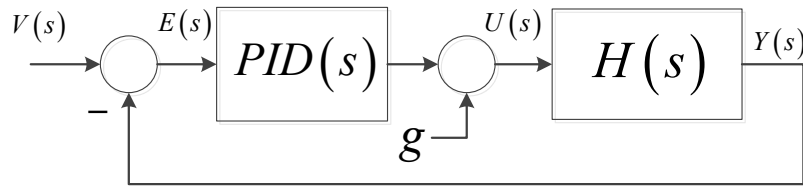


Figure 8-13. Wind levitation feedback structure.

The transfer function of the PID controller is:

$$PID(s) = K_p \left(1 + \frac{1}{T_i} \int e(t) dt + T_d \frac{de}{dt} \right),$$

where K_p , T_i , T_d are proportional gain, integrator, and derivative constant, respectively.

For the controller, the design uses a function 'pidtool' in Matlab software or a similar PID-tuner package (Python script, pidtuner.com, etc.). Tune the PID controller with the given closed-loop performance characteristics:

- Overshoot $\leq 15\%$.
- Settling time $\leq 32s$.
- Rise time $\leq 25s$.
- Steady-state error $\leq 2\%$.

8.4.3 Experimental results

The tuned controller parameters are,

$$K_p = 1.59,$$

$$T_i = 11.2,$$

$$T_d = 0.25.$$

The PID controller is:

$$PID(s) = 1.59 \left(1 + \frac{1}{11.2} \int e(t) dt + 0.25 \frac{de}{dt} \right).$$

The tuned controller is implemented over web-GUI for the Wind levitation system (Figure 8-14).

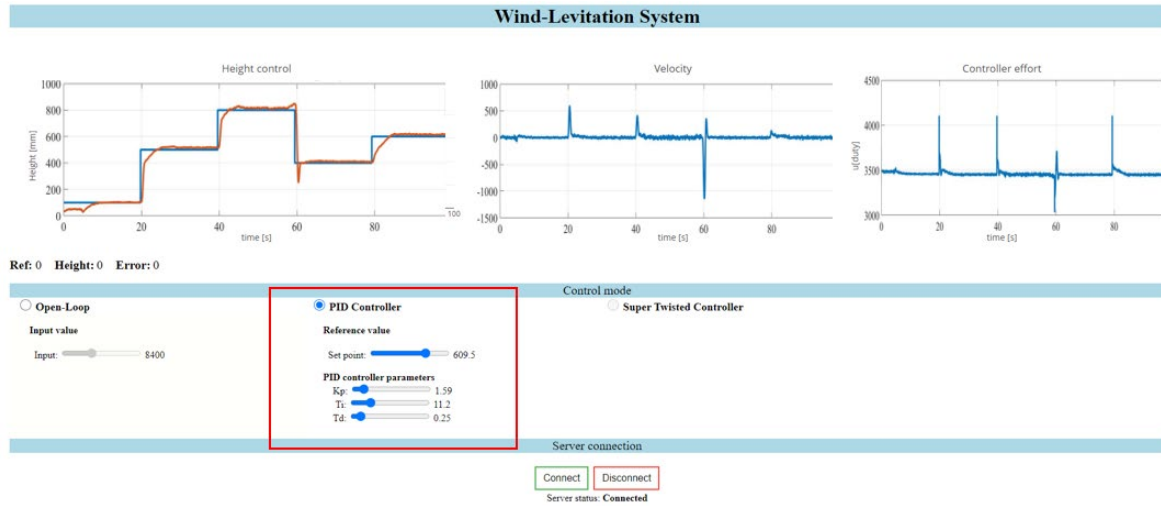


Figure 8-14. Real-time experiment of Wind levitation feedback control with web-GUI.

A close view of the experimental results is presented in Figure 8-15.

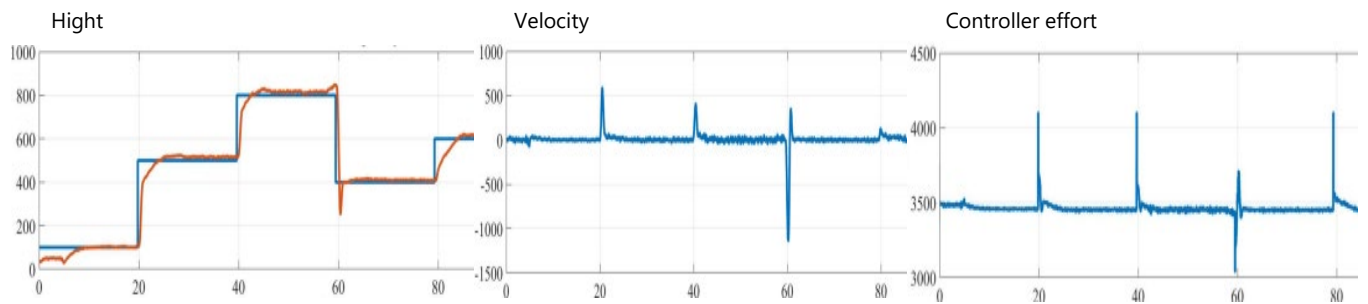


Figure 8-16. Real-time experiment of Wind levitation feedback structure with web-GUI.

8.5 Conclusion

The presented laboratory exercise presents the capability of the developed system. All the practices can be executed remotely, and the data are accessible to the student in real time. Such a system can be ported into vast platforms and web pages. The platform can be used online or offline. All the presented exercises are pilot examples for students with basic knowledge of the control theory and system or first level of study. The exercises are developed gradually.

The first exercise is an example of how to determine the transfer function directly from the measurement. Many industrial control applications (temperature, pressure, flow systems) can be described with first or second order differential equations. With the given exercises, the student can learn how to deal with the transfer function coefficient efficiently before starting with the controller design without a complex modelling procedure and analytical approaches. The second exercise is related to the first. The derived model from the first exercise is used as the plant for the feedback system design with the lag compensator. The lag compensator is designed to improve the tracking capability of the heater system. The design procedure uses the root-locus approach, where the dominant pole of the system is compensated with zero of the lag compensator. The compensator pole is placed close to the origin, which lowers the steady state error and improves the feedback system's tracking capability. The third exercise involves the design of the feedback system for unstable systems. The stability issue is essential in the design procedure, whereby the performance criteria must be met. The design introduced a non-classical feedback structure with direct compensation of the gravity constant. The Wind levitation system is an example of the many practical systems such as hovering drones, helicopter height stabilization, air pressure stabilization in the air piping, air suspension etc.

For further development, the exercises can be developed for students with advanced knowledge and researchers. With the capability of remote execution, there is no need for additional hardware or laboratory equipment. For the advanced exercises, the model predictive control (MPC) can be introduced for the Dual temperature control system or nonlinear controllers such as Sliding mode control (SMC) or Backstepping design for the Wind levitation system.

9 Authors of Virtual Laboratory Manuals

Editors:

Gorjan Nadzinski, Marija Kalendar, Zivko Kokolanski, Branislav Gerazov, University Ss. Cyril and Methodius, Skopje, North Macedonia

Authors:

Programmable Logic Controllers Laboratory:

Filip Doncevski, University Ss. Cyril and Methodius, Skopje, North Macedonia

Scilab Virtual Software Laboratory:

Marija Poposka, University Ss. Cyril and Methodius, Skopje, North Macedonia

Digital Signal Processing Laboratory:

Marija Markovska Dimitrovska, University Ss. Cyril and Methodius, Skopje, North Macedonia

Embedded Systems Laboratory:

Aleksandra Zlatkova, Marko Kamilovski, University Ss. Cyril and Methodius, Skopje, North Macedonia

Virtual Instrumentation Laboratory Platform:

Zivko Kokolanski, University Ss. Cyril and Methodius, Skopje, North Macedonia

System Administration Laboratory:

Anastasiia Sapeha, Kirill Karpov, Eduard Siemens, Anhalt University, Germany

Closed-loop Control and Wind Levitation System Laboratory:

Dusan Gleich, Andrej Sarjas, Blaz Pongrac, University of Maribor, Slovenia