Ministry of Higher Education and Scientific Research

Hassiba Benbouali University of Chlef

Faculty of Science and Technology

Department of Electrical Engineering

Memory

Presented for graduation from

MASTER IN: Electrical Engineering

Specialty :Electric control

By

MAKHALDI MARWA

AISSA BOKHTACHE BOTHAYNA

Theme :

Remote Control of a GDS Oscilloscope Via Internet Dedicated to Educational Lab work

Defended on 25/06/2025 in front of the jury composed of:

Adil YAHDOU	MCA/ UHB-Chlef	President
Maamar SOUAIHIA	MCB/ UHB-Chlef	supervisor
Hakima MOSTEFAOUI	MCB/ UHB-Chlef	Examiner
Omar MAAROF	MAA/ UHB-Chlef	Examiner

Academic Year 2024-2025

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Hassiba Benbouali de Chlef

Faculté des Sciences et technologie

Département d'électrotechnique

Mémoire

Présenté pour l'obtention du diplôme de

MASTER EN : électrotechnique

Spécialité : commande électrique

Par

MAKHALDI MARWA

AISSA BOKHTACHE BOTHAYNA

Thème :

Remote Control of a GDS Oscilloscope Via Internet Dedicated to Educational Lab work

Soutenue le ../06/2025 devant le jury composé de :

Adil YAHDOU	MCA/ UHB-Chlef	Presedent
Maamar SOUAIHIA	MCB/ UHB-Chlef	encadreur
Hakima MOSTEFAOUI	MCB/ UHB-Chlef	Examinateur
Omar MAAROF	MAA/ UHB-Chlef	Examinateur

Année Universitaire 2024-2025

THANKS

We would like to express our deep gratitude to Mr. **Maamar SOUAIHIA**, our supervisor, for his exemplary support throughout this project.

His scientific rigor, constant availability, and insightful advice greatly contributed to the success of this project.

Beyond his role as supervisor, we thank for the trust he placed in us, his attentive listening skills, and, above all, for creating an environment conducive to learning, curiosity, and excellence.

This project would not have been possible without his sincere commitment, his patience, and the wealth of his always constructive feedback.

We express our sincere and respectful gratitude to him.

DEDICATION

Alhamdoulilah always and forever.

Not as I first dreamed and not at the time I thought it would be, but I've arrived. I dedicate this work and my deep gratitude to all those who sacrificed themselves to provide me with the conditions conducive to my success:

Firstly, To myself:

You were on the verge of discouragement, you experienced failure, fatigue, injustice, the Are you sure it's meant for you?

But you didn't give up, you started again, you started again. You chose yourself, and today you are here. This diploma is your war trophy.

To my father:

My first-ever hero, my unwavering source of strength. Your compassion, wisdom, and everlasting love continue to shape who I am. once made you a promise… and today, I fulfil it……. Withall my love and deep gratitude for everything you've given me—thank you, Dad.

To my mother:

The blessing in my life, the twin of my soul. Your unconditional love, boundless selflessness, and nurturing spirit are unmatched. I am endlessly grateful for your prayers, your countless sacrifices, and your patient heart.

To my brothers – Imad, Chaima, and Abdelwadoud

Thank you for being my unwavering support, my joy through both calm and storm.
You made the good times even brighter and helped carry me through the difficult ones.
I treasure every moment we've shared and every word of encouragement you've given.
Thank you for always walking beside me—with love, strength, and unity.

My friends: AYA,MARWA, ZINEB

Nabil, Abderahman and all who stood by me:

Your friendship has been a source of encouragement and inspiration. Through shared challenges and victories, your support never wavered. Thank you for your kindness, loyalty, and belief in me. I am truly grateful to have walked this path alongside you.

MARWA MAKHALDI

DEDICATION

In the Name of Allah, the Most Gracious, the Most Merciful

This is not just an academic achievement —

It is the echo of two years lived in dua, in heartbreak, in helpless giving, and in endless hope.

I dedicate this work, wholly and humbly,

to Gaza.

To the people who have taught the world that resilience is not a slogan, but a way of breathing under siege.

To the youth who fought with stones, with pens, with silence — and with belief.

Through every lecture, every formula, every sleepless night,

Your strength echoed in my soul.

Your names filled the margins of my notebooks.

Your courage was the motivation behind every line I wrote.

And when I almost gave up, I remembered you did not.

This degree may bear my name,

but it carries your story.

To Gaza —

You are not a memory.

You are the reason I write,

the reason I raise my hand in prayer before I hold my pen,

and the proof that truth, though still stands tall.

This is for you,

with every heartbeat, every word, and every prayer.

AISSA BOKHTACHE Bothayna

Summary

List of figures

List of tables

List of symbols

AJAX: Asynchronous JavaScript and XML.

CSS: Cascading Style Sheets.

DOM: Document Object Model.

DRM: Digital Rights Management.

HTML: Hypertext Markup Language.

HTTP: Hypertext Transfer Protocol.

GPIB: General Purpose Interface Bus.

IP: Internet Protocol.

LAN: Local Area Networks.

MVC: Model-View-Controller.

My SQL: Structured Query Language Database.

PHP: Hyper Texte Preprocessor.

P2P: Peer-to-peer.

RS232: Serial Port.

SCPI: Standard Command for programmable instruments.

SSH: Secure Shell.

TCP: Transmission Control Protocol.

TLS: Transport Layer Security.

UML: Unified Modelling Language.

UP: Unified Process.

USB: Universal Serial Bus.

XML: Extensible Markup Language.

IoT: Internet of Things

FTP: File Transfer Protocol

UDP: User Datagram Protocol

XHR : XMLHttpRequest

ملخص:

تقدم هذه المذكرة نظام تحكم عن بُعد لراسم الاهتزازات المهبطي GDS عبر الإنترنت، وهو مصمم للاستخدام العملي في مختبرات التدريس. الهدف الرئيسي هو تمكين الطلاب من الوصول إلى المنظار وتشغيله عن بُعد، مما يُسهّل تعلمهم وإجراء التجارب العملية دون الحاجة إلى التواجد الفعلي في المختبر.

ولتحقيق هذا الهدف، طُوّرت واجهة ويب تجمع بين تقنيات مثل Hypertext Markup Language(HTML) وJavaScript وHyper Texte Preprocessor(PHP) وPython، لضمان اتصال سلس مع المنظار عبر الشبكة عبر بروتوكول Standard Command for programmable instruments(SCPI). بالإضافة إلى ذلك، يتضمن النظام ميزات إدارة الأوامر واسترجاع البيانات في الوقت الفعلي لتوفير تجربة مستخدم مثالية.

تُظهر النتائج أن هذا النظام يُتيح تحكمًا عن بُعد فعالًا وسهل الاستخدام في المنظار، مما يُسهم في تحديث تدريس العلوم التجريبية وتلبية احتياجات التعلم عن بُعد، لا سيما في ظل تزايد رقمنة التعليم.

Résumé:

Ce mémoire présente un système de contrôle à distance d'un oscilloscope GDS via Internet, conçu pour les travaux pratiques en laboratoire d'enseignement. L'objectif principal est de permettre aux étudiants d'accéder à l'oscilloscope et de l'utiliser à distance, facilitant ainsi leur apprentissage et la réalisation d'expériences pratiques sans avoir à être physiquement présents au laboratoire.

Pour atteindre cet objectif, une interface web a été développée, combinant des technologies telles que le Hypertext Markup Language (HTML), JavaScript, Hyper Texte Preprocessor (PHP) et Python, afin d'assurer une communication fluide avec l'oscilloscope sur le réseau via le protocole Standard Command for programmable instruments(SCPI). De plus, des fonctionnalités de gestion des commandes et de récupération des données en temps réel sont incluses pour offrir une expérience utilisateur optimale.

Les résultats démontrent que ce système permet un contrôle efficace et intuitif de l'oscilloscope à distance, contribuant ainsi à moderniser l'enseignement des sciences expérimentales et à répondre aux besoins des formations à distance, notamment en contexte de digitalisation croissante de l'éducation.

Abstract:

This thesis presents a system for remote control of a GDS oscilloscope via the Internet, designed for practical work in the teaching laboratory. The main goal is to allow students to access and use the oscilloscope remotely, making it easier for them to learn and perform hands-on experiments without having to be physically present in the lab.

To achieve this goal, a web interface was developed, combining technologies such as Hypertext Markup Language (HTML) , JavaScript, Hyper Texte Preprocessor (PHP) and Python, to ensure smooth communication with the oscilloscope over the network via the Standard Command for programmable instruments(SCPI) protocol. In addition, order management and real-time data retrieval features are included to provide an optimal user experience.

The results show that this system allows efficient and intuitive control of the oscilloscope remotely, thus contributing to modernizing the teaching of experimental sciences and meeting the needs of distance learning, especially in the context of the increasing digitalization of education.

GENERALE INTRODUCTION

General introduction

The rise of information and communication technologies, including the Internet of Things (IoT) and embedded systems, has profoundly transformed the way electronic devices and measuring instruments are controlled and used. This digital revolution has paved the way for the remote operation of laboratory equipment, allowing users to interact with physical instruments remotely, in a flexible, efficient and secure way.

In this context, controlling a digital oscilloscope via the Internet represents both a technological and educational challenge. From an industrial point of view, the remote control of the instruments allows for continuous supervision, preventive maintenance, and a significant reduction in downtime. On the teaching side, it offers students and researchers remote access to experimental tools, promoting independent learning and the democratization of online experimentation.

However, the implementation of such a system is not without its difficulties. remote control of an oscilloscope, via SCPI commands, raises significant challenges in terms of network communication, latency, signal synchronization, and security of data exchanges. Therefore, essential to design a solution that is reliable, efficient and accessible.

The present thesis is part of this dynamic and has as its main objective to design and develop a web application to remotely control a GDS-2204A oscilloscope connected locally to a computer, through a Python server. The user is able to interact with the device from their phone or any other terminal connected to the Internet, without having to install a complex dedicated server.

To achieve this objective, the project is structured around the following axes ,analysis of the operation of the GDS-2204A oscilloscope and its SCPI interface ,design of a communication server capable of relaying commands between the web interface and the instrument ,development of an interactive user interface based on web technologies. Evaluation of system performance in terms of response time, stability, and ease of use.

In the content of digital and automation, the command at a distance from the equipment and systems that use the latest technology. The pilot allows for dispositifs à distance via the logic interfaces, and it is suitable for communication protocols, microcontrollers, and programming languages. This project will be written in this login and will enable the development of a system to complete the command at a distance.

This work is structured into three chapters, each addressing a specific stage of the project:

The first chapter presents the communication technologies, client-server architecture, and network protocols used in the project.

The second chapter focuses on the system design, including hardware selection and identification of the required software components. It also covers the development of the control application, implementation, testing of outputs, and the user interface.

Finally, the last chapter analyses the results obtained to evaluate system performance and suggest possible improvements. [1]

CHAPTER I:

Theoretical Foundation

I.1 Introduction

In recent years, the ability to control physical systems remotely via the Internet has transformed industries, research laboratories, education, and daily life. This approach, known as Internet-based remote control, allows users to monitor, configure, and operate devices and instruments from virtually anywhere in the world using a computer, smartphone, or tablet.

Remote control systems combine network technologies, embedded systems, and software platforms to provide real-time access to hardware such as oscilloscopes, power supplies, robotic systems, and smart appliances. This paradigm not only increases flexibility and efficiency, but also significantly reduces the need for physical presence, thus saving time, costs, and resources.

With the emergence of the IoT and cloud-based control systems, remote operations are becoming more secure, scalable, and intelligent. Whether it's a researcher controlling a laboratory device from home, an engineer tuning equipment on a remote site, or a technician diagnosing faults in industrial machinery across borders, remote control over the Internet has become an essential capability.

This report/project explores the technologies and methods used in implementing remote control systems over the Internet, highlighting practical examples, communication protocols, and the integration of user interfaces with backend services (e.g., Python + Serial communication).

I.2 History and evolution

The concept of remote control over long distances has fascinated engineers and scientists for over a century. One of the earliest documented demonstrations of wireless remote control was performed by Nikola Tesla in 1898, when he presented a radio-controlled boat at Madison Square Garden. This groundbreaking invention illustrated the feasibility of sending commands wirelessly to manipulate machines, a principle that underpins many modern remote-control systems [2].

With the exponential growth of internet technologies in the late 20th century, researchers began exploring how the internet could be used as a medium for remote control. This shift allowed systems to be accessed and manipulated not just across short distances via radio or infrared, but globally over the web. A landmark project in this evolution was the "Telegarden," developed in 1995 by Ken Goldberg and his team. The Telegarden enabled users from all over the world to remotely control a robotic arm via a web interface, allowing

them to plant seeds, water them, and observe their growth in real time [3]. This project combined the fields of robotics, internet communication, and user interface design, and is often cited as one of the first public experiments in web-based remote physical interaction.

The success of the Teegarden highlighted both the technical feasibility and the societal potential of internet-based remote control. It also laid the foundation for modern applications such as remote laboratories, smart homes, industrial automation, and telemedicine. By enabling users to control real-world systems through standard web browsers, this early example demonstrated a new paradigm in human-machine interaction—one that transcended physical location and opened the door to global, collaborative, and real-time control environments [3].

I.3 Industrial and educational applications

Communication technologies and client-server architectures are widely used in both industrial and educational fields. They enable system control, data exchange, and remote interaction, making them essential tools for automation, learning, and experimentation.

I.3.1 Industrial applications

In modern industrial settings, remote control and monitoring of equipment have become increasingly essential. One such application is the remote operation of laboratory instruments like the GDS-2204A oscilloscope via the internet. By integrating mobile or web-based interfaces with the oscilloscope, users can perform essential functions such as starting measurements, adjusting voltage/div settings, switching channels, and retrieving waveform data without being physically present in the lab [4].

Furthermore, the remote system can include user authentication, real-time signal display, and SCPI command-based communication between a mobile client and a server connected to the oscilloscope.

Some key applications include:

- Industrial Automation.
- Predictive Maintenance.
- Energy and Utilities.
- Remote Troubleshooting.
- Safety Monitoring.

I.3.2 Educational Applications

In education, remote access to laboratory equipment plays a vital role in enhancing the learning process, particularly in electronics and engineering programs. One key application is remote or virtual laboratories, which allow students to perform real experiments on physical devices through internet interfaces, regardless of their geographic location [5].

This approach promotes self-directed and interactive learning, giving students the freedom to schedule their own practical sessions and reinforcing their understanding of theoretical concepts through real-time experimentation [5].

In education, remote control technology has revolutionized laboratory access and hands-on learning, especially in science and engineering:

- Virtual Laboratories.
- Distance Learning.
- Collaborative Research.
- STEM Education.

I.4 Design Architecture

The progression of the client-server architecture will not be linear. Indeed, this technology will develop at a very accelerated pace. It can be said that it has grown exponentially in all areas of activity:

- Database management.
- Transactional systems.
- Messaging systems, web, Internet.
- Data sharing systems.
- Scientific calculations.

I.4.1 Client/ server

Many applications operate in a client/server environment, which means that client machines (machines that are part of the network) contact a server, a machine that is usually very powerful in terms of input-output capabilities, which provides services to them. These services are programs that provide data such as time, files, connection, etc. Services are operated by programs, called client programs, that run on client machines [6]. (Figure.I.1)

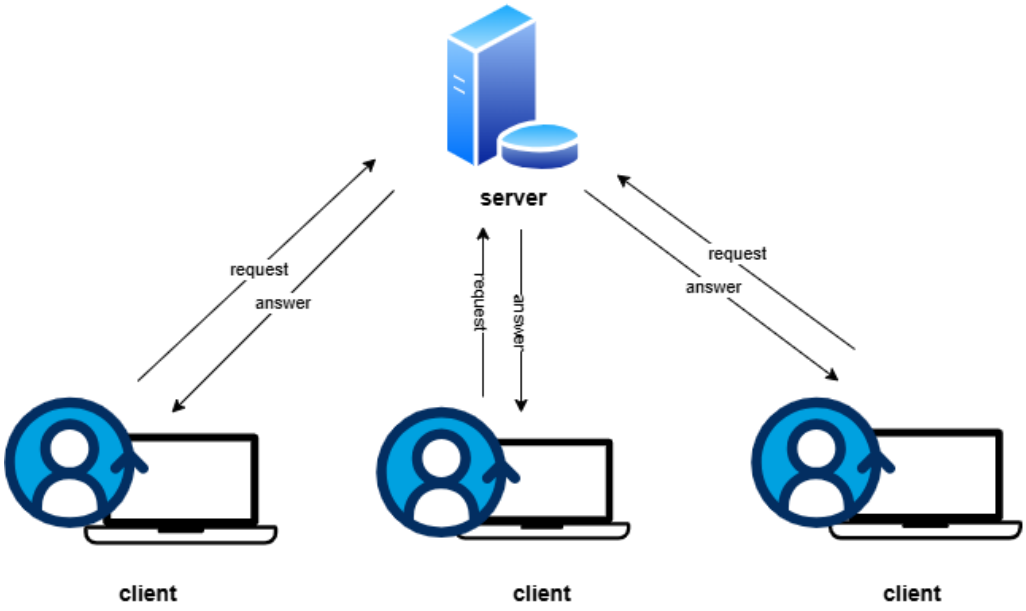

Figure I.1: Client-server architecture

The-Server:

Computers specialize in the provision and storage of shared resources of network users. Characteristics of a server: Service provider:

• It listens, ready to respond to requests sent by customers.

• As soon as a request reaches him, he processes it and sends a response.

• Processing multiple clients simultaneously.

The Client:

computers that access shared resources provided by a server on the network. Characteristics of a customer:

• Service consumer.

• Establishes a connection to the server.

• Sends requests to the server.

• Waits for and receives responses from the server.

The Query

A message sent by a client to a server describing the operation to be performed on behalf of the client.

Answer:

A message transmitted by a server to a client after an operation has been performed, containing the result of the operation [6].

I.4.2 Client-server models

The client-server model is a key concept in networked computing, where the client requests services and the server provides them. This structure supports centralized control, modular design, and scalability across local network. The client sends a request, and the server responds accordingly.

Variants of this model include:

I.4.2.1 Two-Tier Architecture

The two-tier architecture is one of the simplest models in distributed systems. It consists of two main components: the client and the server. The client handles the user interface and sends requests to the server, which processes the requests and communicates directly with the data base [7].

This architecture is often used in local networks or small-scale systems where the number of users is limited. It is easy to implement and provides fast performance in basic environments. However, the lack of separation between the business logic and data access logic results in reduced scalability and flexibility. Moreover, updates and maintenance are more challenging as changes to one part of the system often affect the whole (Figure I.2).[7]

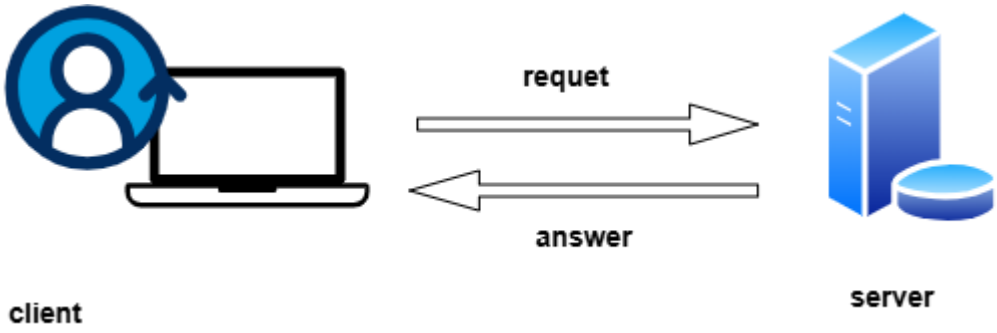

Figure I.2: two-tiers client server

I.4.2.2 Three-tiers Client/Server Model

To overcome the limitations of the two-tier architecture, an additional tier was introduced. The purpose of the additional tier (called as "middle" or "rule" tier) is handling application execution and database management, as with the two-tier model, the tiers can either be implemented on different physical machine or multiple may be co-hosted on a single machine.

In the three-tier architecture the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platform (Figure.I.3) [8].

Figure I.3: Three-tiers Client/Server.

I.5 Communication protocols

There are several types of protocols depending on their function:

I.5.1 Transmission Control Protocol / Internet Protocol (TCP/IP)

TCP/IP is the set of communications protocols used for the Internet and other similar networks.

These protocols are the backbone of modern digital communication, enabling reliable data exchange across both global Internet and local area networks (LANs). This makes them essential for applications requiring remote control and monitoring, such as the one implemented in our project.

It is named from two of the most important protocols in it: the (TCP) and the (IP), which were the first two networking protocols defined in this standard. IP networking represents a synthesis of several developments, namely the internet and LAN (Figure.I.4) [9].

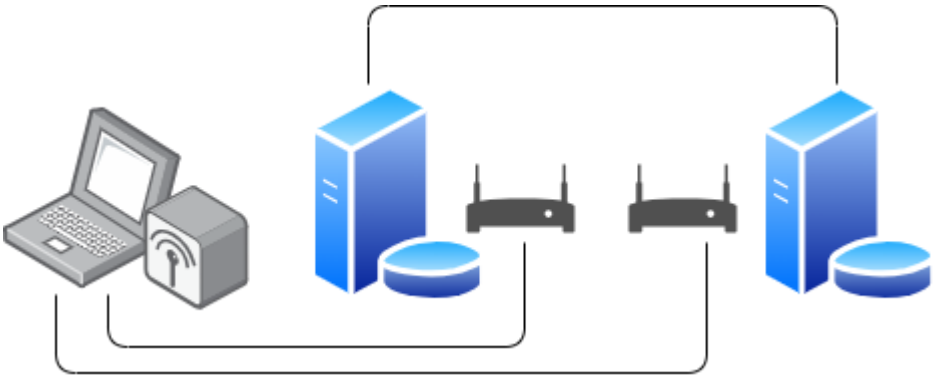

Figure I.4: general model of TCP/IP.

The Internet Protocol Suite, like many protocol suites, may be viewed as a set of layers. Each layer solves a set of problems involving the transmission of data and provides a well-defined service to the upper layer protocols based on using services from some lower layers **The TCP/IP model** is composed of four layers**:**

Application layer: Interfaces directly with user applications , e.g., Hypertext Transfer Protocol(HTTP), File Transfer Protocol (FTP).

Transport layer: Provides reliable data transport (e.g., TCP, User Datagram Protocol(UDP).

Internet layer: Handles logical addressing and routing (e.g., IP).

Network access layer: Manages hardware addressing and access to the physical network (e.g., Ethernet).[9]

I.5.1.1 TCP protocol

TCP is a required TCP/IP standard defined as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks .

It achieves reliability through mechanisms such as sequence numbers, acknowledgments (ACK), and retransmission of lost or corrupted data. These features make TCP a connection-oriented protocol, ensuring that data is delivered accurately and in the correct order, which is especially important in applications like remote oscilloscope control. [9]

I.5.1.2 The IP Addresses version 4

TCP/IP uses 32-bit binary addresses as universal machine identifiers. Called Internet Protocol addresses or IP addresses, the identifiers are partitioned into two parts: a prefix identifies the network to which the computer attaches, and the suffix provides a unique identifier for the computer on that network. The original IP addressing scheme is known as classful, with each prefix assigned to one of three primary classes. Leading bits define the class of an address; the classes are of unequal size [9].

For example, the IP address 192.168.1.1 belongs to class C. The classful addressing system is divided as follows:

Class A: 0.0.0.0 to 127.255.255.255 : used for very large networks.

Class B: 128.0.0.0 to 191.255.255.255 : used for medium-sized networks.

Class C: 192.0.0.0 to 223.255.255.255 : used for small networks. [10]

In our project, this addressing format allowed us to assign a unique IP address to the oscilloscope on the LAN, enabling direct and stable communication between the client interface and the instrument through TCP/IP and SCPI commands.

I.5.1.3 Protocol model

The Internet Protocol Suite, the official name for TCP/IP stack, consists of four layers. We have sat in this (Table I.1):

Application layer	FTP HTTP TELNET DNS RIP
Transport layer	TCP UDP
Internet layer	IPV4 IPV6 OSPF EIGRP
Link layer	ETHERNET FRAME RELAY IS-IS

Table I.1: protocol model [11]

I.5.2 HTTP

HTTP is an application protocol that runs on top of the TCP/IP protocol suite. It defines how messages are formatted and transmitted, and how web servers and browsers should respond to various commands. One of the concepts of the HTTP protocol includes the idea that files can contain references to other files (hence the notion of hypertext), whose selection will solicit other transfer requests. HTTP operates as a stateless protocol, meaning each request is independent, and the server does not retain information about previous interactions. To manage user sessions or preferences, additional mechanisms like cookies are often used.

Moreover, HTTP supports different request methods such as GET, POST, PUT, and DELETE, allowing a wide range of interactions with web resources. For example, the GET method retrieves information, while POST submits data to the server for processing. HTTP typically uses port 80 for communication, while its secure version, HTTPS, runs over port 443 and uses encryption protocols like Secure Sockets Layer / Transport Layer Security (SSL/TLS) to ensure data confidentiality.

In addition to the web page files they serve, all web servers contain an HTTP daemon, which is a program designed to wait for HTTP requests and process them when they arrive. These daemons handle requests from clients, locate the requested resources, and send appropriate responses. The notion of hypertext allows web pages to contain embedded links to other pages or resources, creating a dynamic and interconnected structure that is central to the web's functionality [12].

I.6 Security of remote communication

Security plays a critical role in ensuring reliable exchanges between two remote systems. Many protocols incorporate protection mechanisms to ensure the confidentiality, integrity, and authentication of the data transmitted. For example, HTTPS, which is a secure version of HTTP, uses Transport Layer Security (TLS) to encrypt communications between a client and a web server. Similarly, protocols such as Secure Shell (SSH) ,for secure remote access or IP offer robust solutions against attacks such as interception or tampering.

Thus, in any modern communication architecture, the integration of security at the protocol level is essential to protect sensitive information exchanged over public or private networks [13].

I.6.1 Safety scheme

Securing communication over a distance is critical in any system where data is transmitted between devices over a network, especially in remote control applications. Without adequate security measures, data can be intercepted, altered, or misused by unauthorized parties.

The following diagram summarizes these key aspects of secure communication (Figure I.5):

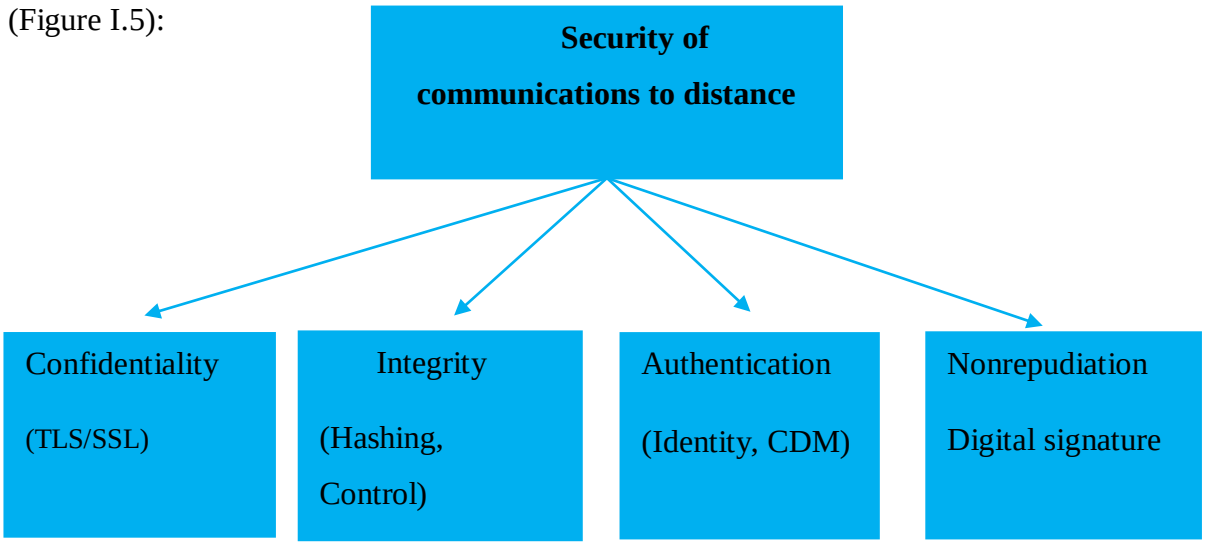

Figure I.5: Safety scheme.

I.7 Languages and tools

In the spirit of challenge, and with the desire to put into practice the knowledge we acquired during our training, we chose HTML, PYTHON, Cascading Style Sheets (CSS), JAVASCRIPT, asynchronous JavaScript and XML(AJAX)and PHP, languages that are

widely used particularly by many professional programmers, making them high-level languages [14].

I.7.1 HTML

HTML is a format for defining the various elements of a "page" viewed with web browsing software: a browser (Netscape, Mozilla Firefox, Internet Explorer) [14].

HTML defines:

- Text.
- The formatting and layout of the text.
- The placement and size of images.
- The positioning of animations and sounds.
- The placement on a page of any other static or dynamic element
- Managed by the browser.
- Hyperlinks to other pages.

Figure I.6: logo of HTML5

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Figure I.7 : Exemple de code HTML

HTML is therefore the language spoken behind the scenes of the web to describe what you see there. HTML was designed at CERN, by scientists far from any commercial considerations, to exchange information across the boundaries of different computers, networks, etc [14].

I.7.2 CSS

CSS is a computer language used on the internet to format HTML or Extensible Markup Language (XML) files. Style sheets, also called CSS files, contain code that manages the design of an HTML page. Although HTML can be formatted using specific tags, nowadays it's more appropriate to use CSS and only use HTML for the content of the web page [15].

Figure I.8: Logo of CSS

```
body {
  background-color: lightblue;
}
h1 {
  color: white;
  text-align: center;
}
p {
  font-family: verdana;
  font-size: 20px;
}
```

Figure I.9 : Exemple de code CSS

I.7.3 PHP

PHP is a computer language used on the internet. The term PHP is a recursive acronym for "Personal Home Page". This language is mainly used to produce a dynamic website.

It is common for this language to be associated with a database, such as Structured Query Language (MySQL), run on the server side (the place where the site is hosted) there is no need for visitors to have any software or plugins. However, web masters who want to develop a site in PHP must make sure that the host takes this language into account [13].

Figure I.10: Logo of PHP

I.7.3.1Why Choose PHP

- A large community of developers sharing hundreds of thousands of PHP script examples

. • The free and available source code (PHP is distributed under the GNU GPL).

- The simplicity of writing scripts.
- The ability to include the PHP script within an HTML page.
- The simplicity of interfacing with databases (many DBMSs) is supported, but the most used with this language is MySQL).
- Integration into many web servers (Apache, Microsoft IIS, etc.).[14]

I.7.3.2 General Formula of a PHP Script

```
PHP

CopyEdit

<?php

// 1. Variable reporting

$variable = "value";

// 2. Processing (conditions, loops, functions, etc.)

if ($variable == "value") {

    // instructions

}

// 3. Output (display or return to customer)

echo "Result: " . $variable;

?>
```

Figure I.11 : Exemple de code PHP

I.7.4 Python

Python is considered one of the text languages that does not require a structure to execute the program written in it. It is also one of the easiest and most strictly organized languages, making it a top choice among academic languages adopted by universities. [16].

Figure I.12: Logo of python

I.7.4.1 General Structure of a Python Program

```
import serial

def send_command(cmd):

    with serial.Serial(port="COM1", baudrate=115200, timeout=1) as ser:

        ser.write((cmd + '\n').encode())

        response = ser.readline().decode()

        return response

if __name__ == "__main__":

    result = send_command("*IDN?")

    print("Oscilloscope says:", result)
```

Figure I.13 : Exemple de code Python

I.7.5 Flask

Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led an international team of Python enthusiasts called Pocock. Flask is based on the WSGI Werkzeg toolkit and the Jinja2 Template engine [15].

Figure I.14:logo of flask

I.7.6 Node js

Node.js is an event-driven JavaScript runtime. Node has countless potential uses for JavaScript development, including being a great environment for building high-performance web applications [17].

Figure I.15: logo of node js

I.7.7 JAVASCRIPT

It is an object-oriented computer programming language commonly used to create interactive effects in web browsers, alongside HTML and CSS. JavaScript is one of the three core technologies of the World Wide Web. This language, created in 1995 by Brendan Eich [17].

Figure I.16: Logo of JavaScript

I.7.8 XML

XML is a universal text document format that has become an important IT standard. It is used to store documents and transfer data between applications. Its simplicity, flexibility, and scalability make it suitable for various fields, from geographic data to vector design and business.

Figure I.17: Logo of XML

I.8 Communication interfaces with measuring instruments

Modern measuring instruments such as oscilloscopes, mustimeters, function generators, and spectrum analysers are increasingly equipped with communication interfaces that enable remote control, data acquisition, and automation. These interfaces are essential in industrial automation, research, and remote laboratories.

I.9 Common Communication Interfaces

Communication interfaces are essential for enabling data exchange between electronic devices and systems. They define both the physical connection, and the protocols used to transfer data. we site Common Types of Communication Interfaces.

I.9.1 General Purpose Interface Bus (GPIB)

GPIB products offer three key benefits to help us save money. over the life of our instrument control system, the development through to production and maintenance.

The GPIB is a standard published by the IEEE under ANSI/IEEE Standard 488. It defines the electrical, mechanical, functional and software specifications for interfacing the instruments programmable to PCs. Let's set up the device's GPIB address and connect the device to the computer using a GPIB cable [15].

Figure I.18: Cable GPIB

I.9.2 Ethernet (RG45/WIFI)

To complete the installation of the system, you will need an internet connection using an RJ45 cable or a Wi-Fi available [15].

Figure I.19 : Cable Ethernet RJ45 et receptor Wi-Fi.

I.9.3 Serial Port (RS 232) interface

The RS-232 port on many of the devices is driven by a Universal Asynchronous Receiver Transmitter (UART). The UART performs conversion of outgoing data from parallel form to serial form and incoming data from serial to parallel form. In addition, it is also equipped with special inputs and outputs, which are used to co-ordinate the flow of data with an external device.

These special purpose inputs and outputs are referred to as the hardware hand-shaking line. Remote control of instruments with RS232, Universal Serial Bus (USB) and GPIB interfaces [9].

Figure I.20: Cable RS232

I.9.4 USB cable

The USB Bus was born of the alliance in 1994 of seven industrial partners (Compaq, DEC, IBM, Intel, Microsoft, NEC and Northern Telecom). Operational specification or standard USB version 1.0 was released in January 1996.

The advantages of USB are several: low-cost interface, power devices possible via cable, independence from host machines, Hot Plug & Play (connection and disconnection without the need of shut down the PC**)** [18].

Figure I.21: Cable USB

I.9.5 Common interface types

In electronic test and measurement systems, communication interfaces is a vital role in enabling data exchange between instruments and controlling devices. This section summarizes the most used interface types in laboratory equipment, understanding these interfaces is essential for designing the oscilloscope and remote control.

This table (Table I.2) summarizes the most common communication interface types:

Interface	Type	Common Application
RS-232	**Series**	**Mustimeters, old oscilloscopes**
USB	**Serial Bus**	**Instruments modern, PC**
GPIB	**Parallel**	**High-end laboratory instruments**
Ethernet	**TCP/IP network**	**Connected instruments, remote access**

Table I.2: Comparison of Common Interfaces

I.10 SCPI Command

SCPI is a universal command language developed to standardize communication and control of test and measurement instruments from different manufacturers.

I.11 Conclusion:

This chapter highlights the ability to remotely explore and control systems through command platforms. These platforms are designed to be efficient, secure, and user-friendly. They enable users to monitor instruments, perform measurements, and analyse data in real time.

Remote control increases flexibility and reduces operational costs. It also enhances safety, especially in sensitive or hard-to-reach environments. Such platforms are widely used in industrial, educational, and scientific settings.

In the following chapter, we will present the practical design of the system architecture, detailing the hardware and software components used to implement remote control of the oscilloscope.

CHAPTER II:

System Architecture Design

II.1 INTRODUCTION

The first part of this work represents the foundational phase in the development of complex systems, where the overall structure and organization of the system are defined. It involves identifying the key components, their roles, the way they interact, and the technologies that will be used. This design serves as a blueprint that guides the development process, ensuring that the system meets both functional requirements—such as performance and usability—and non-functional requirements like scalability, security, and maintainability. By outlining how data flows between components and establishing clear interfaces, system architecture helps reduce complexity, minimize development risks, and make future enhancements more manageable. Whether applied to software systems, embedded systems, or large-scale enterprise solutions, a well-structured architecture is essential for building reliable and efficient systems.

II.2 Choice of communication architecture

The design of an efficient communication architecture is essential for enabling remote interaction in a reliable, secure, and scalable manner. Several architectures are commonly used in modern systems, each offering specific advantages depending on the context of use, the required performance, and the technical constraints.

In what follows, we present four different communication architectures that could be applied to our system:

II.2.1 Client-Server Network (direct)

In a client–server network, there is at least one dedicated central server that controls the network, and a few clients connect to the server to carry out specific tasks. A client-server network can have more than one central server, each performing a specific function. Functions may include user access, data storage, internet connection management, and network traffic monitoring, among others.

Multiple clients connect to one central server. A client is a computer or computer-controlled device that provides users with access to data on the remote server. Types of clients include smartphones, desktop computers, and laptop. This network architecture facilitates

efficient communication and resource sharing between the central server and connected clients, enabling seamless task execution and data access across various devices [19].

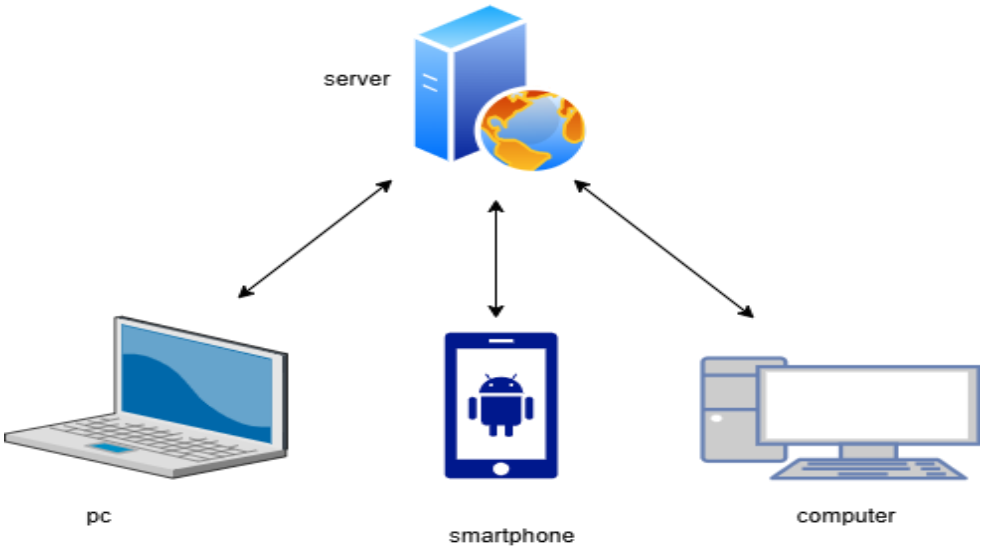

Figure II.1: Model client-server

II.2.2 Peer to peer(P2P)

Peer to peer systems, direct file sharing systems among the peers, are one of the most attractive file sharing systems. P2P architectures have high scalability and high performance because its architectures which have characteristics of distributed file processing. However, P2P Architecture is infamous for distribution channel of illegal contents. So, we must apply the Digital Rights Management (DRM)system to the P2P architecture, and we should keep advantages of P2P even if after DRM system applied. server.

In this paper, we propose new type of DRM applied P2P system architecture that keeps existing P2P system's advantages [18].

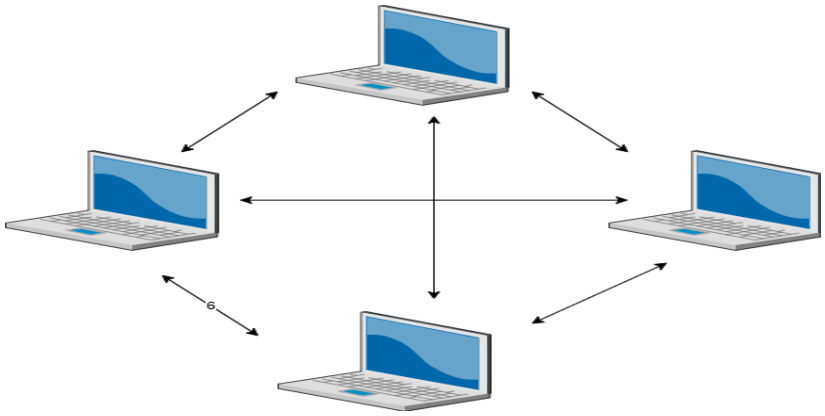

Figure II.2: Architecture P2P

II.2.3 WebSocket

The WebSocket architecture is based on a model of bidirectional and real-time communication between a client (browser or mobile application) and a WebSocket server.

Unlike HTTP which follows a request/response pattern, WebSocket establishes a persistent connection, allowing the server and client to exchange data at any time without having to reopen a new connection [21].

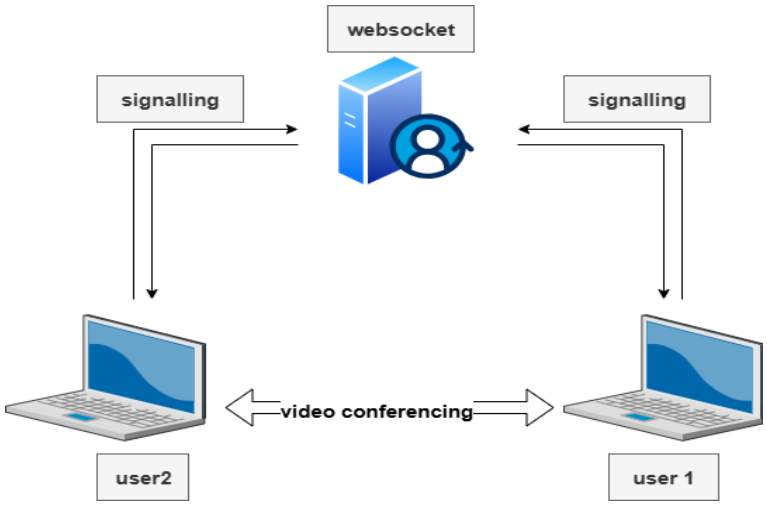

FIGURE II.3: Architecture WebSocket

II.2.3.1 WebSocket server

WebSocket server is the core part of the interface program, as building a stable server for the whole system is particularly important. WebSocket server needs to analyse the WebSocket

protocol, including the analysis of sending packets, handshake connection, data encryption [21].

II.2.3.2 WebSocket client

WebSocket client The WebSocket client uses the JavaScript as programming language. The WebSocket client is used to relate to the server and receive data from the server, what's more, it transfers data to the remote ATS system as an array. The WebSocket client listens for WebSocket connection status and processes events on the connection, message receiving, disconnections, and erroneous connections [21].

II.2.4 Cloud

The Cloud architecture makes it possible to centralize the processing, storage and management of data on remote servers accessible via the Internet. In a remote-control application, such as an oscilloscope, this architecture offers great flexibility, global accessibility, and simplified management. Cloud is gaining popularity as means for saving cost of IT ownership and accelerating time to market due to ready-to-use, dynamically scalable computing infrastructure and software services offered on Cloud on pay-per-use basis. Design of software solution for delivery as a shared service over Cloud requires specific considerations [22].

The user (client) interacts with a web or mobile interface. This interface sends requests to a cloud platform, where an application server processes the commands, redirects them to the connected oscilloscope, and returns the measured data to the client.

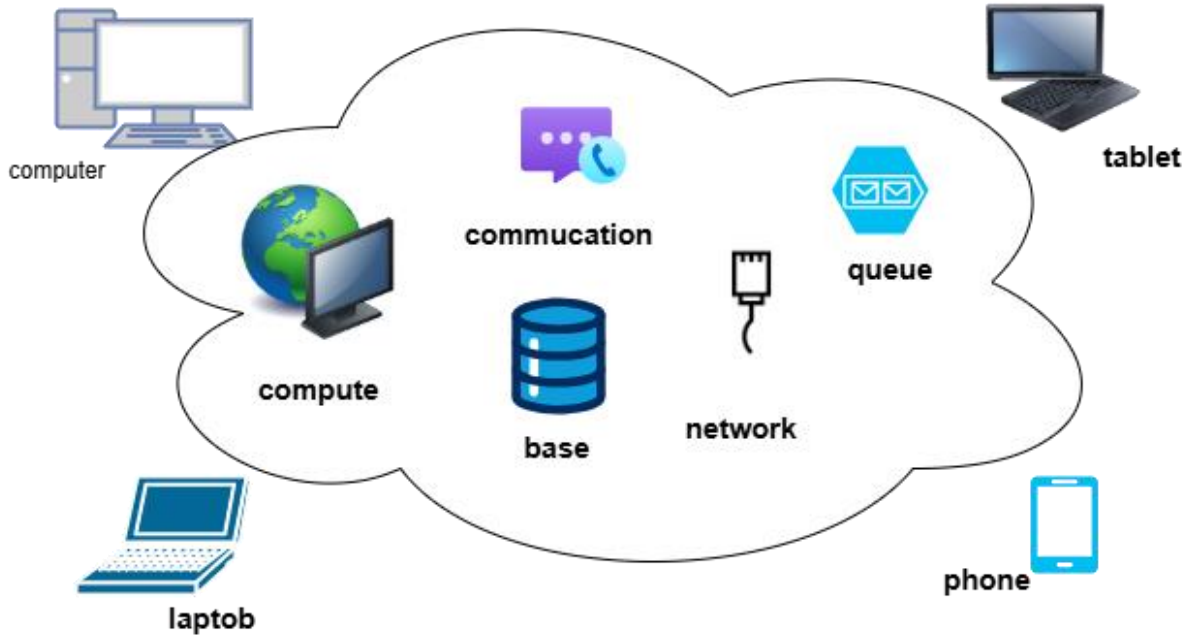

Figure II.4: Architecture cloud

II.2.5 Comparison of the different possible architectures

We represent in this table (Table II.1) a comparative overview of different communication architectures commonly used in networked control system

Criterion	Client–Server	P2P (Peer to Peer)	WebSocket	Cloud
Communication Model	Centralized: one server, multiple clients	Peer-to-peer distributed	Bidirectional, real-time (client-server)	Centralized with global access via the Internet
Connection	Client initiates queries	Each peer can be client and server	Persistent connection via single socket	Requests via API/remote server
Scalability	Medium to High (Adding Servers)	Very high (distributed architecture)	Medium (limited by server capacity)	Very high (elastic cloud)
Security	Easy to control	Difficult (risk of illegal content)	Good, requires SSL/TLS	Very good with advanced management tools

Table II.1: Comparison of the different possible architectures

- In summary

1. Client-server: ideal for structured systems with few clients and simple exchanges.
2. P2P: excellent for massive data sharing but not very suitable for instrument control.
3. WebSocket: A powerful solution for real-time exchanges, such as controlling an oscilloscope from a browser.
4. Cloud: Recommended for systems that can be accessed remotely at any time and integrate centralized services.

we use in our project adopts a client-server model for its simplicity, reliability, and centralized control, enabling efficient communication between the user interface and the oscilloscope while enhancing security and maintainability.

II.3 Control System Components

A control system is composed of multiple elements achieve precise regulation and interaction with a target device. In the context of this project, the objective is to implement a remote-control solution for the GDS-2204A oscilloscope via a client-server structure. The system involves a graphical user interface, a communication server, and the physical oscilloscope. Each component plays a vital role in ensuring seamless real-time control and monitoring over a local network.

II.3.1 System to be controlled

The system to be controlled in this project is one of the basic devices in electronics laboratories, which is the digital oscilloscope.

II.3.1.1 Digital Oscilloscope

The digital oscilloscope is a modern device that implements digital signal processing techniques. It thus has several advantages that have caused the gradual disappearance of the analogy oscilloscope; these are some of the advantages: - Real-time signal analysis. The storage of digital data, acquired in external devices (memory) for later visualization [23]. Figure (Figure II.5) show the interface of GDS2204a.

Figure II.5: Digital Oscilloscope 2204a

II.4 Communication Manager (hardware and software interface)

In the project to remotely control the GW Instek GDS-2204A oscilloscope, the Communication Manager plays a fundamental role. It provides the interface between the hardware (oscilloscope) and the software (control application) using an Local Area Networks link(LAN) . This software component centralizes the management of commands, responses, and the communication status between the PC and the instrument.

II.4.1 Hardware Interface

The hardware interface is the physical interface between the digital control system and the oscilloscope. It is responsible for passing commands and data between the server and the device. This interface varies depending on the type of connection supported by the device.

II.4.1.1 LAN communication

LAN is a communication network that interconnects various data communication devices within a small geographic area, typically a building or campus. It supports high data transfer rates (up to 1 Gbps or more) with very low error rates. LANs are widely used in commercial and academic environments to facilitate access to shared resources such as file servers, printers, and instrumentation devices. In the context of laboratory equipment, a LAN allows for centralized control and monitoring, including remote access to oscilloscopes, such as the GDS-2204A, via SCPI commands over TCP/IP. Due to its secure and isolated nature, a LAN

ensures that sensitive data and controls are kept within the internal network. Data Communication Over LAN [24].

1. LAN uses the TCP/IP model, which includes:

- IP (Internet Protocol): Identifies devices (like address of oscilloscope) TCP (Transmission Control Protocol): Ensures reliable connection
- Port: Identifies application/service (e.g., SCPI server on the oscilloscope)

2. Data Communication Over LAN

- LAN uses the TCP/IP model, which includes:
- IP (Internet Protocol): Identifies devices (like address of oscilloscope)
- TCP (Transmission Control Protocol): Ensures reliable connection
- Port: Identifies application/service (e.g., SCPI server on the oscilloscope).

II.4.2 Software Interface

The software interface is designed using advanced languages such as Python or C and relies on the use of standardized SCPI commands to communicate with the instrument via USB or RS-232. This interface allows parameters to be configured, data acquired and displayed in numerical or graphical form., the power of modern computers, data exchange takes place in near real time, offering smooth and precise control of the oscilloscope. The intuitive HMI facilitates interaction with the system, making it accessible to students and researchers without direct physical contact with the device. This, the software interface ensures efficiency, automation and synchronization in laboratory experiments at the same time. This (Figure II.6) represents the software communication.

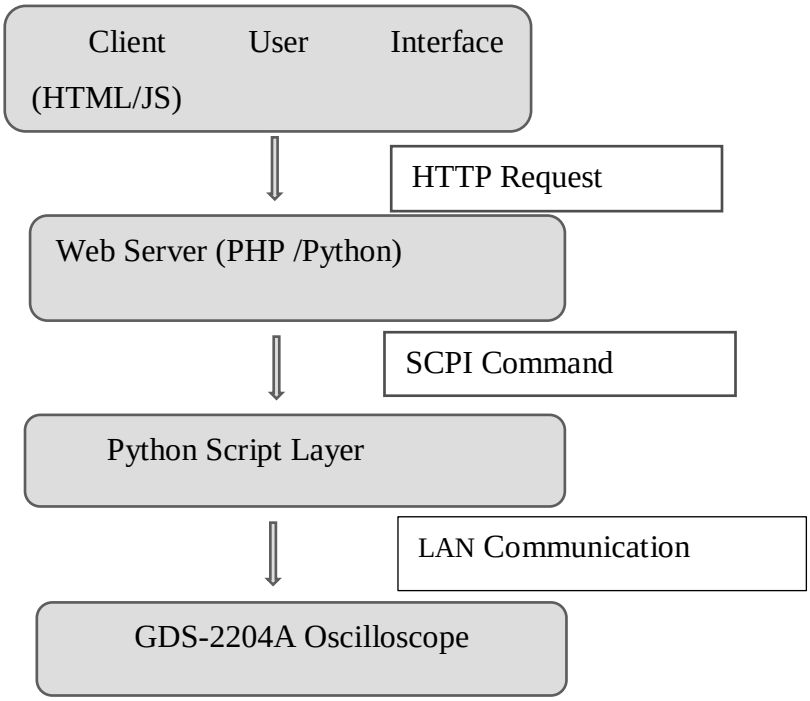

Figure II.6: Software Interface Architecture

II.5 Web Server and Application (Apache):

XAMPP is open-source free software, it contains Apache distributions for the Apache server, MariaDB, PHP and Perl, and it is essentially a local host or local server, possible to test your website by XAMPP before uploading it to the web server or remote computer. This XAMPP server software provides you with a suitable environment to test MySQL, PHP, Apache and Perl projects on your computer [17].

We Using Apache through XAMPP ensures cross-platform compatibility and facilitates a smooth transition from local development to a live server, as it uses the same components commonly deployed in production. This makes Apache via XAMPP an ideal choice for developing and testing the oscilloscope control system's server-side application efficiently.7 The figure below (Figure II.7) represent the page of server XAMPP.

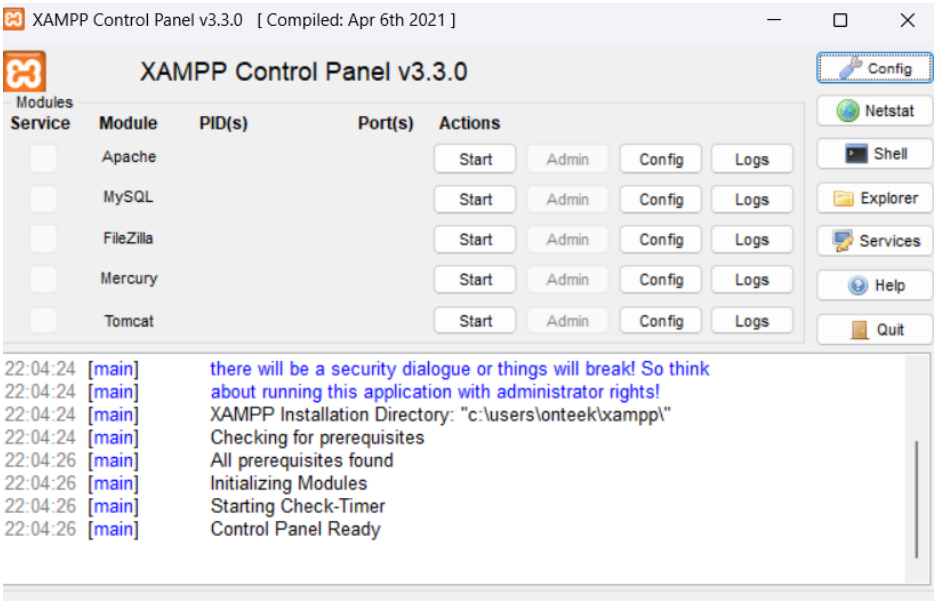

Figure II.7: Page server XAMPP

II.6 User Client (GUI for Remote Access)

The user client is a graphical application that allows an end user to interact with the GDS-2204A oscilloscope remotely. It is the "Vue" layer in the system's Model-View-Controller(MVC)architecture. It connects to the Communication Manager (installed on the local PC or server) via a software interface and allows the sending of SCPI commands as well as the display of responses [25].

II.6.1 Objective

Enable remote control of the oscilloscope without physical access, providing a simple and intuitive interface for configuration and reading of measurements.

II.7 Technologies Utilizable

Choosing the appropriate technologies is a crucial step in developing the control system, as it directly impacts usability, performance, and maintainability. This section explores different technology options for implementing the user interface and application layers, considering various use cases such as web access, local desktop applications, and optional mobile support.

This table (table II.2) represents an overview of the main technology options considered for developing the control system’s user interface. Each option targets different usage scenarios and platform

Option	Technologies	Description
Web interface	HTML, CSS, JavaScript + Python (Flask or Fast API)	Interface accessible from a web browser. Ideal for remote access over LAN or Wi-Fi.
Local application	Tkinter, PyQt, (Python)	Simple desktop application works directly on the PC controller. Faster and localized.
Mobile (optional)	JavaScript (WebView) or PWA	Mobile version to access via phone or tablet.

Tableau II.2: Technologies Utilisable

After analysing the oscilloscope's operating principles and identifying the functional requirements, this part focuses on the development of a remote-control application through an interactive web interface. A structured methodology based on the MVC architecture (Model–View–Controller) has been adopted to ensure a clear separation of concerns between the user interface, data processing, and hardware communication.

This section is divided into three key components:

II.8 Software Modelling and Design:

The design and modelling of the remote-control application for the GDS oscilloscope follow structured software engineering practices to ensure maintainability, scalability, and clear separation of concerns. Two essential approaches were adopted: the (MVC) architectural pattern and the use of Unified Modelling Language (UML) diagrams [26].

II.8.1 MVC Architecture:

The design pattern we're going to choose and that meets our needs is MVC is a design method that (HMI) of a software application. Below are the model general of MVC(Figure II.8)

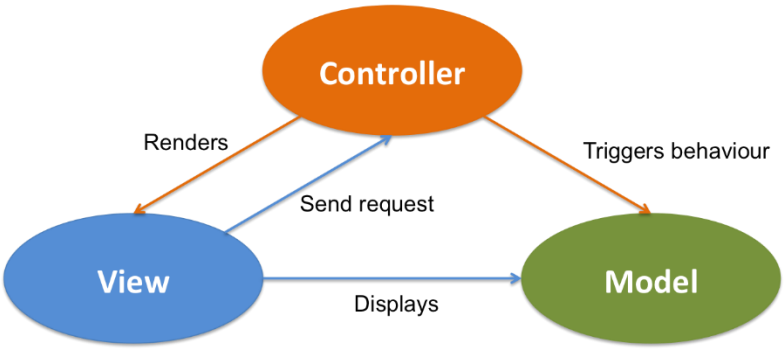

Figure II.8: model general MVC

This paradigm divides the HMI into:

II.8.1.1 The Model:

This part manages the site's data. Its role is to retrieve the "raw" information from the database, organize it, and assemble it so that it can then be processed by the controller [26]. In our case, it includes:

1. Management of booking sessions.
2. The classes of communication with the oscilloscope via supported protocols (e.g., USB, Ethernet, or SCPI).
3. Processing of acquired data (signals, measurements, and screenshots).
4. Recording session histories or logs.

II.8.1.2 The View:

This part focuses on display. It performs almost no calculations and simply retrieves variables to determine what to display [26]**.** In our project, this includes:

1. Login and booking forms.
2. The graphical interface for remote control of the oscilloscope (control buttons, real-time display of measurements).
3. Visualization of experimental results (curves, measured values).

II.8.1.3 The Controller:

This part manages the logic of the code that makes decisions [26]. It is responsible for:

1. Authenticate users.
2. Validate reservations.
3. Transmit commands to the oscilloscope via the model.
4. Refresh the view according to the model's answers (e.g., display of results).

II.8.2 Advantages of the MVC model in this project

1. **Clear separation of responsibilities**, facilitating testing and maintenance.
2. **Modularity**: Ability to modify the interface without affecting the business logic.
3. **Reusability** of code in other interfaces (e.g., mobile, desktop).
4. **Easier collaborative work**: Backend and frontend developers can work independently.

The (Figure II.9) below shows the general shape of the MVC model:

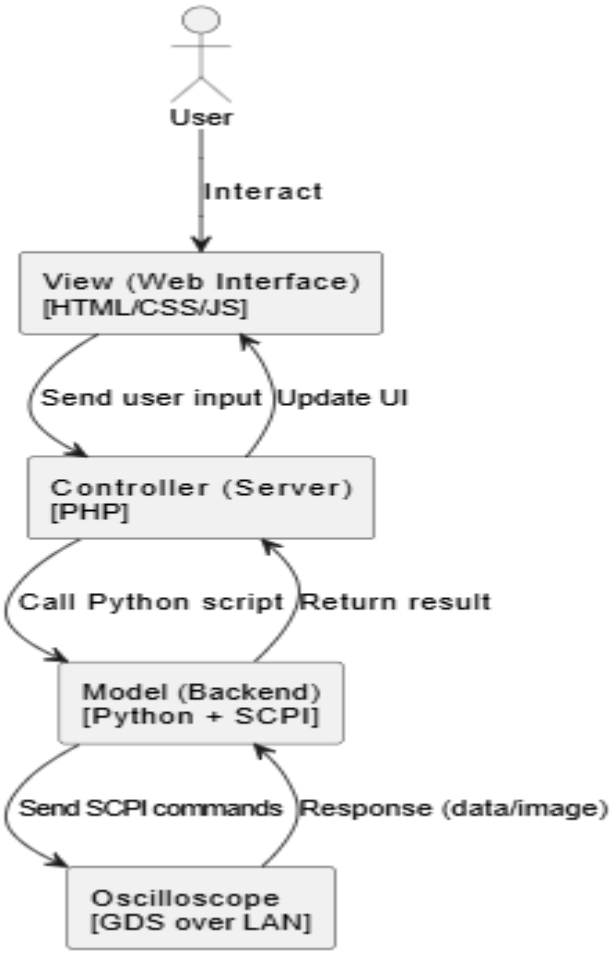

Figure II.9: Model MVC

II.9 UML diagrams

One of the most important steps in programming is the design phase. It must not be skipped otherwise serious errors can occur.

The object-oriented approach adopted in our project has several advantages, the most notable of which are modularity, extensibility and reusability.

For our application, we chose modelling based on the UML programming language and the Unified process (UP). We used use case diagrams, sequence diagrams and class diagrams.

Among the many software tools we used to draw diagrams, we chose plant UML because it is free and open source.

II.9.1 Introduction to UML

UML is a formalized, standardized language for object-oriented modelling. It is independent of programming languages, application domains, and processes, and has versatility and flexibility that make it a general-purpose language. In addition, UML is essentially a communication medium that facilitates the representation and understanding of object-oriented solutions. Graphical symbols can visually represent object-oriented solutions, thereby facilitating the comparison and evaluation of solutions. Graphical symbols reduce ambiguity and misunderstanding.

UML provides an ingenious way to represent different projections of the same representation using views.

II.9.2 Types of UML

UML provides a standardized way to visualize the design and structure of a software system. It is used to represent both the static and dynamic aspects of a system through a variety of diagram types. In our project, UML diagrams help how data flows, and how responsibilities are distributed across the system.

The most relevant types of UML diagrams for this application include:

II.9.2.1 Sequence diagram

A sequence diagram is UML diagram that represents the sequence of messages between objects during an interaction. A sequence diagram consists of a group of objects, represented by lifelines, and the messages that these objects exchange during the interaction [12].

Sequence diagrams represent the sequence of messages transmitted between objects. They can also represent the control structures between objects. For example, the lifelines in a sequence diagram for a banking scenario might represent a customer, a teller, or a branch manager [12].

The communications between the customer, teller, and manager are represented by the messages between them. The sequence diagram represents the objects and the messages

between these objects [12].

The (Figure II.10) below represent the diagram of sequence:

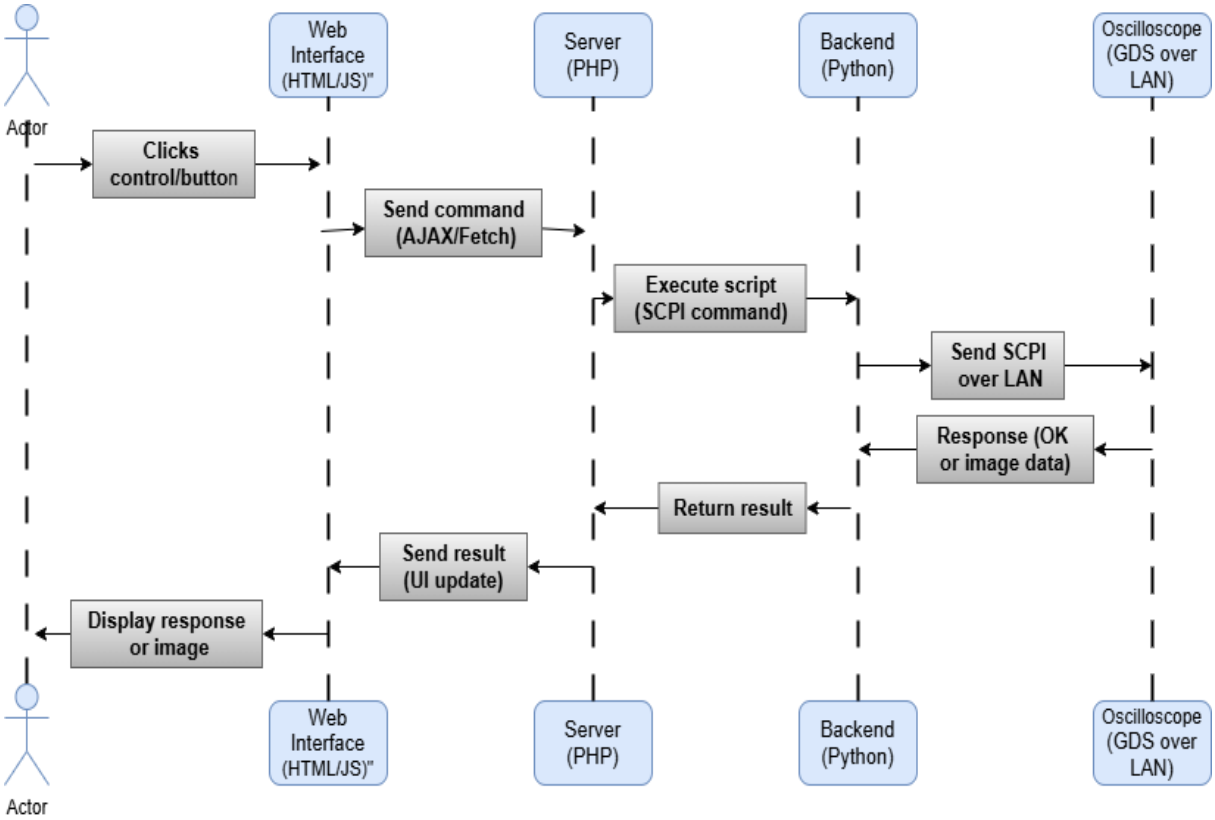

Figure II.10: diagram sequence

II.9.2.2 Class Diagram:

Class Diagram: The most used UML diagram and the basis of any object-oriented solution. It includes the classes, attributes and operations of the system and the relationships between the various classes. When modelling large systems, classes are grouped into class diagrams. they represent the static structure of the system, including its classes, attributes, operations and objects [16].

They can represent statistical data or organizational data in the form of implementation classes or logical classes. There may be overlap between these two groups.

The following diagram shows (Figure II.11) diagram of class:

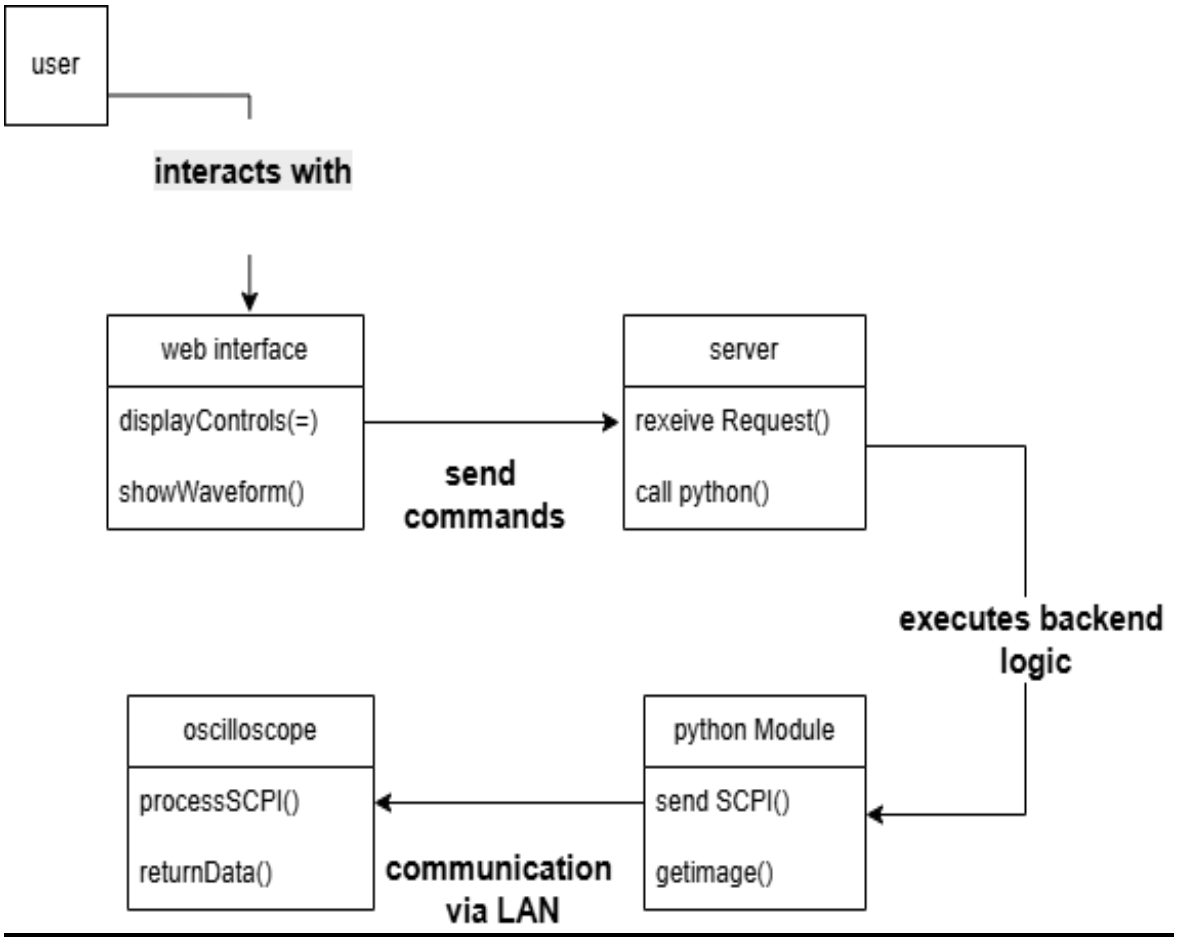

Figure II.11: Class Diagram

II.9.2.3 Use Case Diagram

The use case diagram provides a user perspective, not an IT perspective. It requires no IT knowledge and ideally should be created by the client [12].

The use case diagram is not an exhaustive inventory of all system functions. It only lists essential and main general functions without going into detail.

This is a diagram (Figure II.11) that presents use case:

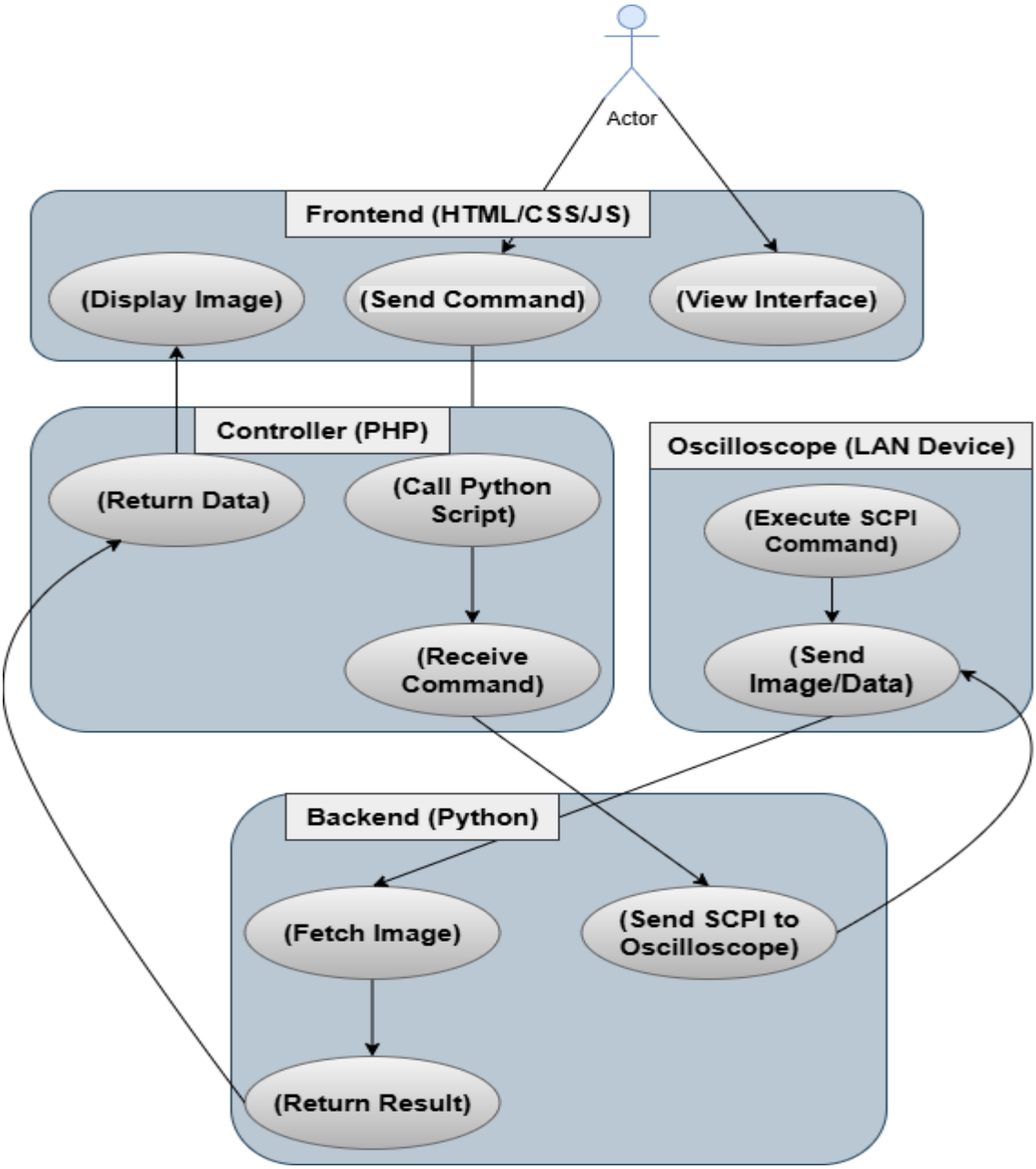

Figure II.12: Use Case Diagram

II.9.2.4 Activity diagram

An activity diagram is used to model the behaviour of the system, including the sequence of actions and their conditions of execution. Stocks are the basic units of the system behaviour. An activity diagram allows you to group and ungroup actions. If an action can be split into multiple actions in sequence, you can create an activity representing them [12].

Figure II.13: Diagram Activity

II.10 Introduction to the Document Object Model (DOM) in HTML

The DOM is like a blueprint of the web page. It represents the HTML content as a tree of objects, allowing developers to access, modify, and interact with elements on the page dynamically. Whether you're hiding a paragraph, changing a color, or creating interactive features with JavaScript—you're working with the DOM.

This DOM (Figure II.14) represents the structure of an HTML document as a tree-like model, where each HTML element is a node connected in a hierarchical way:

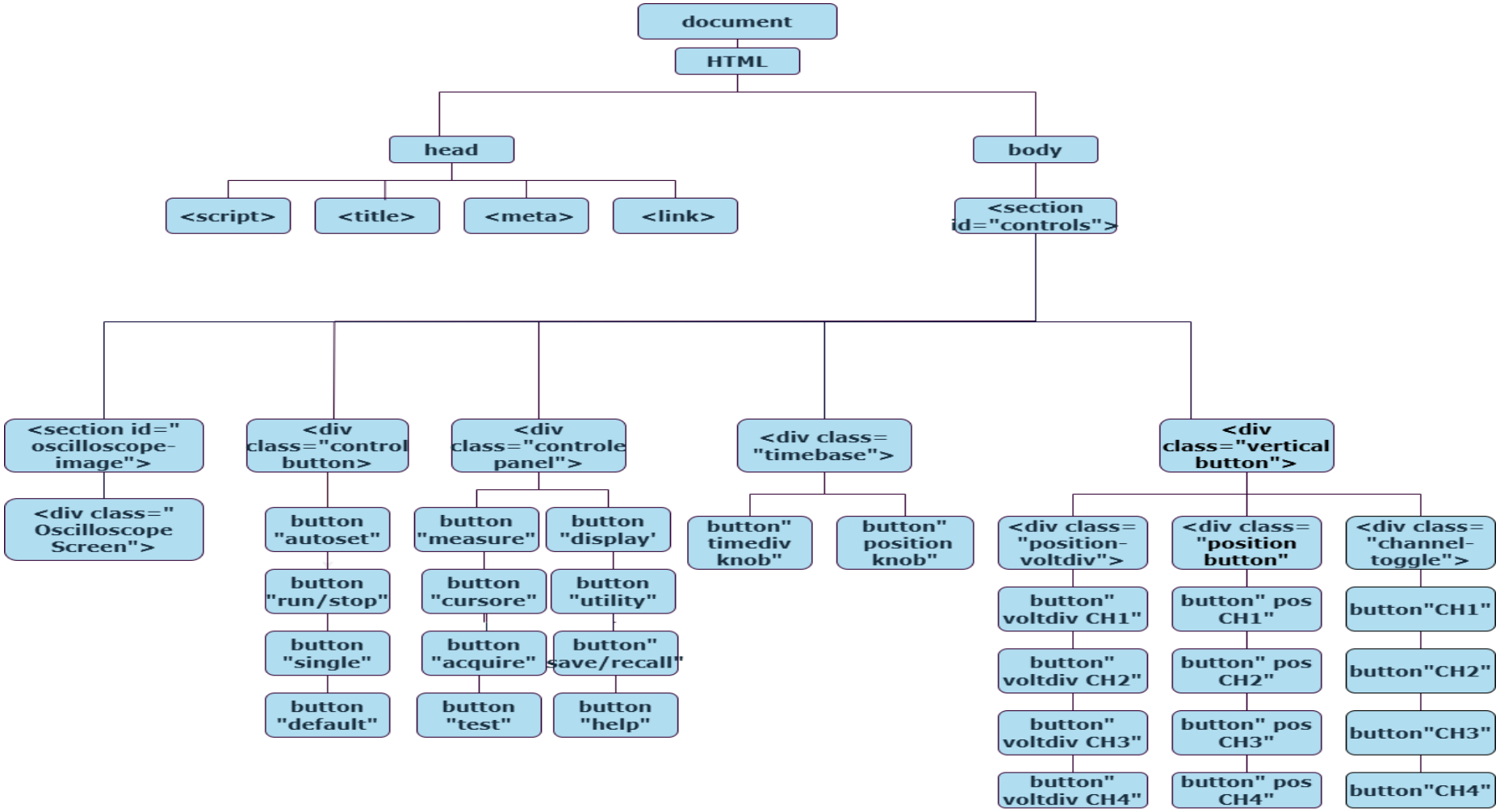

Figure II.14: Representation of the HTML DOM

II.11 Web client Development

The development of the web client focused on creating an intuitive and responsive user interface, allowing remote control of the oscilloscope. The use of AJAX technology, data exchanges with the server are done without a complete reload of the page, providing smooth navigation and instant updating of the information displayed [27].

This approach improves the user experience by making the application faster and more interactive, while ensuring efficient communication with the server for SCPI order processing.

II.12 AJAX

AJAX is a technique that brings together several already existing in known XML and JAVASCRIPT languages, so it consists of sending and receiving data without refreshing the page [15]. (Figure II.15) represent How Ajax technology works

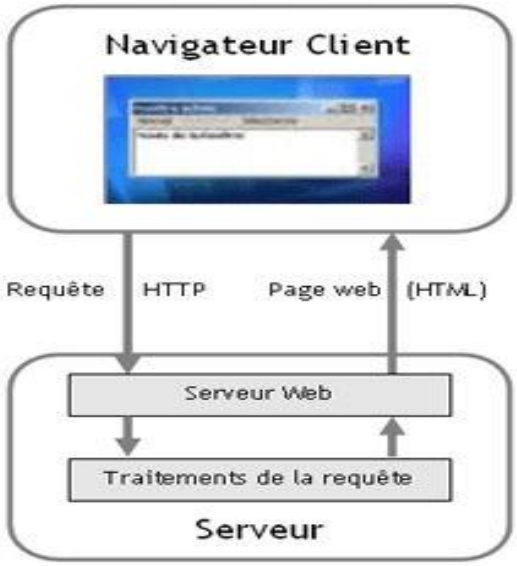

Figure II.15: Ajax technology works

II.12.1 Using AJAX for smooth communication with the server

The Ajax technology is said to make the Internet faster, more interactive and user-friendly. It is spreading rapidly throughout the web. By allowing scripts being executed inside the user's browser to communicate with a remote server, it enables new forms of interaction for webpages and -applications. In this paper we analyse Ajax usability, considering both advantages and imposed problems. Subsequently we present a user study with two typical scenarios in web-applications and compare Ajax-enabled and non-Ajax-enabled versions [27].

This allows us to verify and quantify the assumed effects. As a result, we show that Ajax significantly enhances both user satisfaction and efficiency of use, at least in some scenarios.

II.12.2 Advantages of such smooth communication:

AJAX introduces a very neat way to manage the transfer of information on the web. It is also famous for providing an enhanced user experience, allowing a web page to be updated using data from a web server without the need for a full-page refresh. Now, however, web developers are increasingly requiring AJAX applications to update page data, not only when the user sends a request, but also when the server decides an update is required [28].

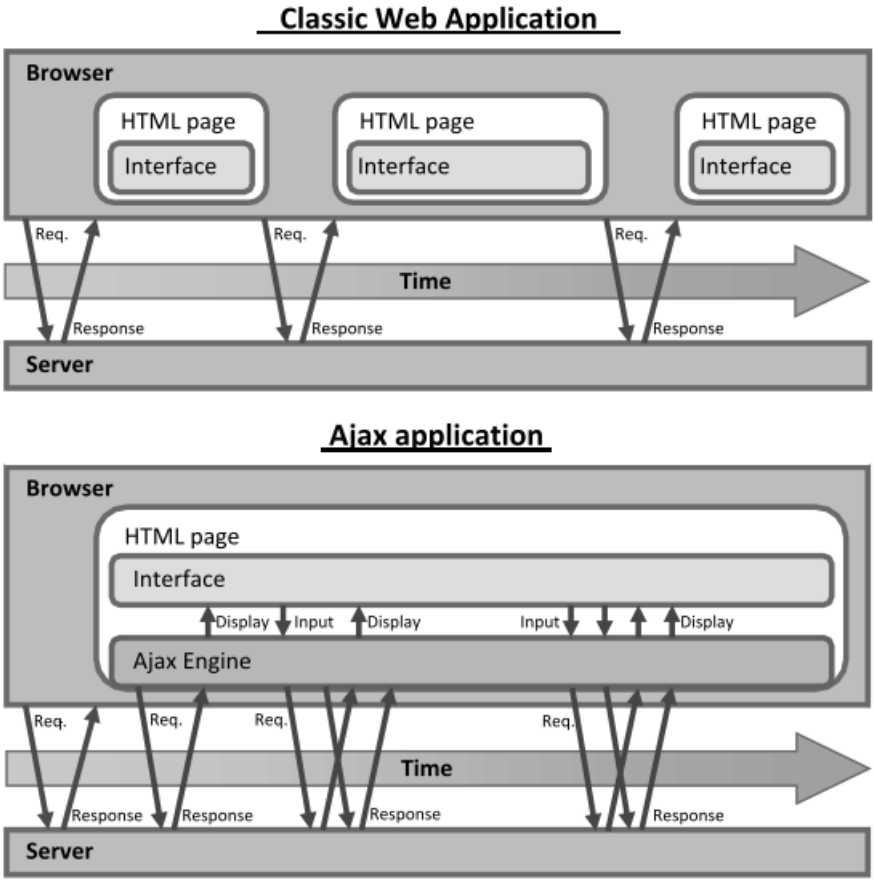

Figure II.16: Comparison of classic web application sand ajax applications.

The importance of AJAX is obvious in some examples of integrated applications, where all users need to be aware of the changes other users make, or monitoring applications with the server needing to let the browser know that its state has changed [28].

II.12.3 Advantages of such smooth communication:

Implementing dynamic, asynchronous communication—particularly using technologies like AJAX—offers several benefits that enhance both the performance and user experience of the web application.

The following (Table II.3) summarizes the main advantages of this smooth communication model:

Advantage	Description
No reloading	The data is dynamically updated.
Reactivity	The interface responds immediately to actions.
Less bandwidth	Only the necessary data is exchanged.
Clear MVC separation	AJAX acts as a bridge between View and Controller.

Table II.3: Advantages of such smooth communication

II.12.4 Server development and communication management

This project consists of creating a web interface that allows a user to send SCPI commands (via HTTP) to interact with a device (e.g. an oscilloscope) via an LAN. The web server (in PHP) will receive the user's commands, pass these commands to a Python script, which will send them to the device via LAN. Then, the server will send the response back to the user.

This figure (Figure II.17) represents the general architecture of this communication flow, the interaction between the user, the web interface, the server, and the device.

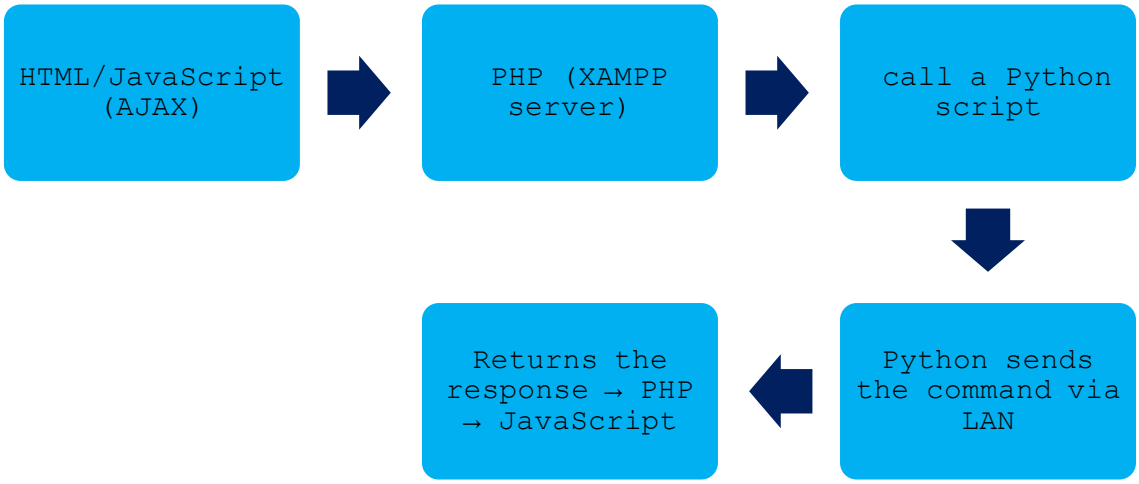

Figure II.17: General architecture of communication

II.12.4.1 Data Decoding

XMLHttpRequest (XHR) objects are used to interact with servers. One can retrieve data from a URL without having to refresh the page completely. This allows a web page to be updated without disrupting the user's actions. XHR is used a lot by the AJAX approach [29]. This picture (Figure II.18) shows the model of XHR:

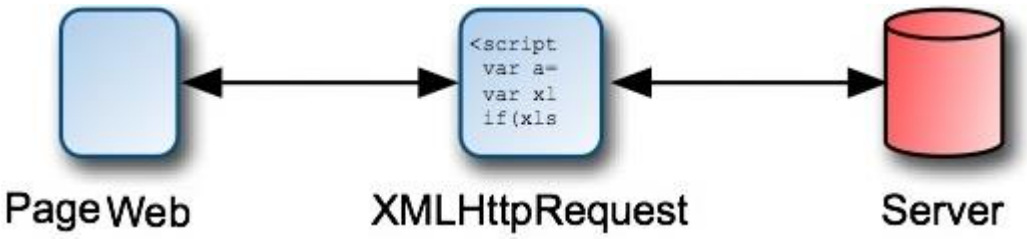

Figure II.18: Model of XHR

II.12.4.2 Decoding steps

The decoding process describes how a user action is translated into a corresponding response from the oscilloscope. This flow involves several key components working together to handle input, communication, and output. Below are the main steps involved in decoding a command.

In this scheme (Figure II.18) represents the schematic view of this decoding step, illustrating how the raw data is transformed and delivered back to the user via the web interface.

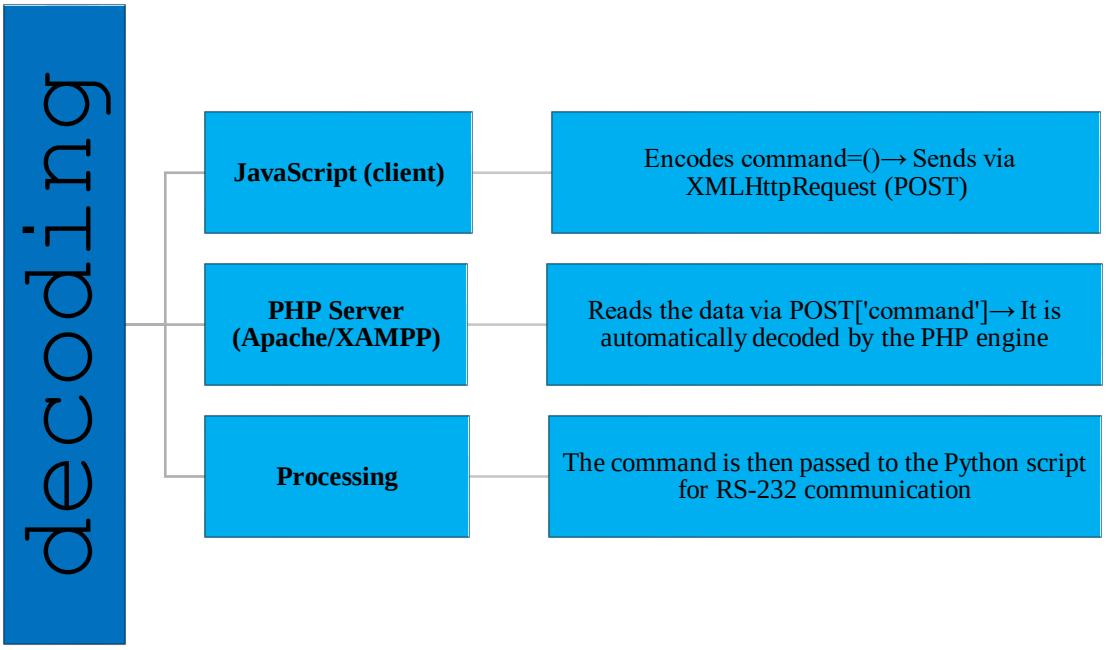

Figure II.19: Schematic Decoding Step

II.13Server-side processing

Once the SCPI command is received by the server via the HTTP POST request, server-side processing consists of transmitting this command to the target device (oscilloscope, measuring instrument, etc.) via a serial interface, typically RS-232. To do this, the PHP server executes an intermediate [30]

Python script, which is responsible for managing serial communication through the serial library. This script opens the serial port, sends the SCPI command, waits for a response from the device, and then sends that response back to PHP. The latter then transmits the response to the web client [31].

This processing provides the link between the remote user interface and the laboratory equipment, allowing real-time control through the SCPI protocol. Precautions are taken to secure the command, including the use of escapes to avoid command injections, and error handling is provided to ensure reliable communication.

A proposed virtual oscilloscope design makes use of SCPI and interface technology. It can be used for data storage, parameter measurement, and waveform presentation. The VISA technology is used for programming oscilloscopes that include interfaces. Additionally, a virtual oscilloscope may acquire, display, and save data in real time without the need for an

additional PC data collection device. It was demonstrated that the design possesses the qualities of dependable operation, adaptable extensions, and usefulness in automatic testing [32].

II.13.1 SCPI:

All instruments must adhere to the standard's common syntax, command structure, and data formats. Any instrument might use the general instructions it introduced. Subsystems comprise these commands. Additionally, SCPI specifies several instrument classes that list the subsystems they implement along with any features unique to each instrument. The SCPI does not define the physical communications channel; Ethernet, USB, and other protocols can be used instead. ASCII text strings make up SCPI commands.one or more keywords, many of which require parameters, make up a command [32].

Integration with the oscilloscope is based on the use of the standard SCPI (Standard Commands for Programmable Instruments) language, which is widely adopted for remote control of measuring instruments. These text-based commands allow you to configure the oscilloscope, initiate acquisitions, change channel parameters (VOLT/DIV, TIME/DIV, etc.), and retrieve measured data.

II.13.2 Communication Protocol

The communication protocol used in this project is based on a serial link (RS-232 or USB-serial, LAN) between the server and the oscilloscope. This binding allows the exchange of SCPI commands and text responses. Each message follows a specific format defined by the device manufacturer, including command syntax, terminator characters (such as \n or \r\n), and response times. This protocol ensures reliable, simple, bidirectional communication, allowing the server to remotely control the oscilloscope and retrieve measurement data in real time.[29]

II.13.3 Input/output and measurement parameter management:

The server acts as an intermediary between the user interface (client) and the oscilloscope. It interprets requests from the interface (e.g., activate channel CH1, change the time base, or trigger an acquisition), converts them into SCPI commands, and then transmits them to the oscilloscope. In return, oscilloscope responses (measured values, channel status, etc.) are processed, formatted, and sent back to the customer for display. I/O management includes selecting active channels, adjusting display parameters (V/div, T/div), initiating

automatic measurements, and retrieving data (frames, waveforms, or encrypted measurements). The system is designed to be responsive and intuitive, ensuring smooth interaction with the oscilloscope while allowing full supervision via a remote interface [29].

II.14 Conclusion

This chapter has detailed the development of a fully functional web application designed for the remote control of a GDS oscilloscope. The implementation is grounded in a robust MVC architecture, which promotes a clear separation of concerns, enhances maintainability, and facilitates future scalability. The user interface, constructed using optimized XML structures and modern web technologies, delivers a responsive and user-friendly experience across various devices and screen sizes.

The next chapter is dedicated to testing and validating the system's functionality. It includes a detailed analysis of the implementation results and proposes improvement.

CHAPTER III: Tests and Results

III.1 Introduction

This chapter presents the practical implementation of the project developed. It begins with an overview of the system's architecture, followed by a detailed explanation of the interface components and their functions. It also discusses the core features that have been successfully implemented, along with their responsiveness and reliability. Additionally, some hands-on experiments are highlighted to demonstrate how the system performs under real-world conditions. At the end, I provide an honest evaluation of the work done and suggest areas that can be improved in future versions.

III.2 System Architecture

The project has been designed in a way that allows remote control of the GDS-2204A oscilloscope through a web browser, whether from a mobile phone or a computer. The front end was developed using HTML, CSS, and JavaScript, while PHP acts as the bridge between the interface and a Python script that communicates with the oscilloscope using SCPI commands. The communication flow can be represented as follows:(Figure III.1).

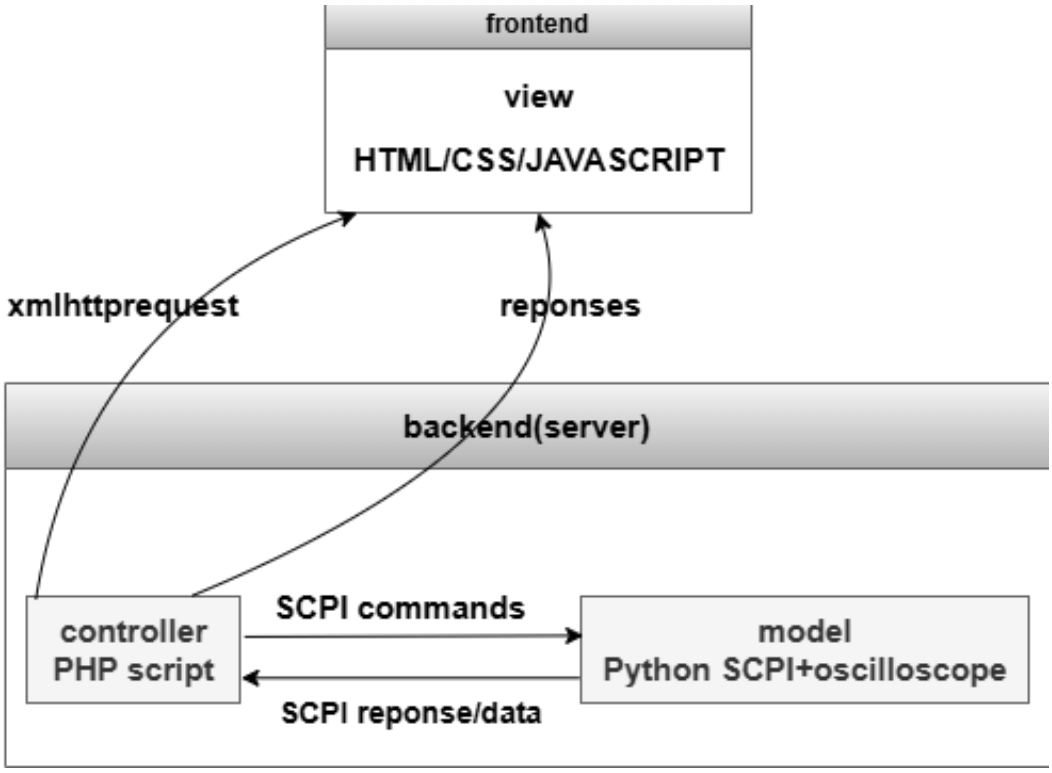

Figure III.1: The architecture general of system

III.2.1.Explanation of the General System Architecture

Figure III.1 illustrates the general architecture of the developed remote control system. It consists of two main layers: the frontend and the backend.

The frontend represents the user interface, which was built using HTML, CSS, and JavaScript. It allows users to interact with the system through buttons that send SCPI commands.

These commands are transmitted to the backend using XHR (AJAX), without reloading the page.

The backend, which runs on the local server (XAMPP), receives the request through a PHP script (controller). This script acts as a bridge between the frontend and the Python script, which is responsible for communicating directly with the GDS-2204A oscilloscope using the SCPI protocol over a TCP/IP connection.

Once the command is executed, the response is passed back from Python to PHP, and then returned to the frontend, where the result is displayed to the user.

III.2.1.1 Frontend Layer Description

The frontend is designed as a simple and user-friendly web page that contains multiple control buttons.

Each button is associated with a specific SCPI command. When a user clicks on a button, a JavaScript function is triggered, which sends the selected command using AJAX to the PHP script.

This design allows real-time interaction without page refresh, enhancing user experience.

III.2.1.2 Backend Layer Description

The backend handles all logic and communication with the oscilloscope.

The PHP script (controller) receives the SCPI command from the frontend and uses shell_exec() to call a Python script, passing the command as an argument.

The Python script establishes a socket connection with the oscilloscope using its IP address and port, sends the command, and captures the response.

This response is then sent back to the frontend and displayed.

III.3 Main control zones of the GDS-2204A oscilloscope

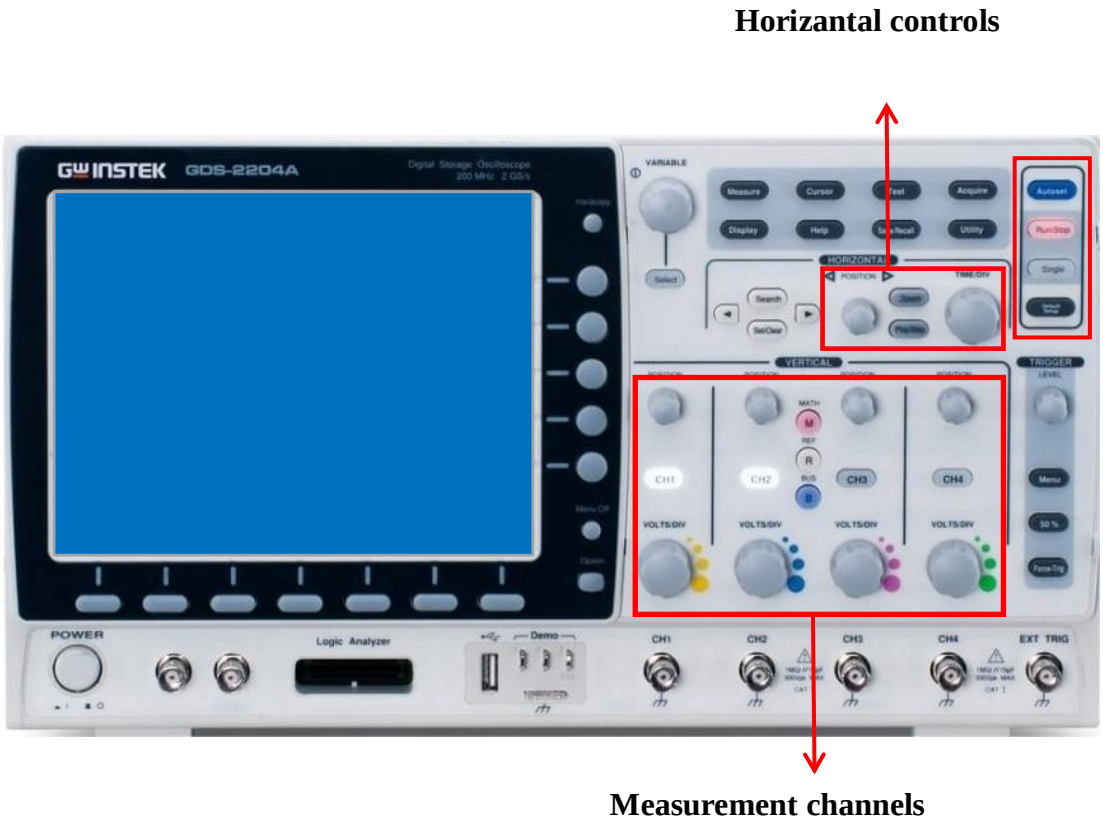

Figure III.2: Main control zones of the GDS-2204A oscilloscope

As shown in **Figure III.2**, the GDS-2204A oscilloscope consists of several functional zones that are essential for measurement and configuration:

The Horizontal controls section allows users to adjust the time scale (TIME/DIV) and the horizontal position of the waveform.

The Measurement channels (CH1–CH4) enable multiple input signals to be analyzed simultaneously. Each channel has its own VOLTS/DIV and POSITION knob.

The Main control buttons on the right side include basic functions like Autoset, Run/Stop, Single, and Utility, which are frequently used for signal acquisition and display settings.

In our web interface, the goal was to replicate the functionality of these buttons and knobs digitally. For example, the Autoset button was mapped to the SCPI command :AUTOscale, and channel controls were implemented using appropriate SCPI commands to activate, configure, or hide channels

III.4 Selected SCPI Commands Used in the Project

UI Element	SCPI Command	Description
AutoSet Button	:AUTOSet	Automatically adjusts display
Run /Stop Toggle	:RUN, :STOP	Starts or stops waveform acquisition
Default Button	*RST	Resets the oscilloscope
Timebase Knob	:TIMebase:SCALe 10e-3	Sets time scale to 10 ms/div
Horizontal Position	:TIMebase:POSition0.005	Shifts waveform horizontally
Channel 1 On/Off	:CHAN1:DISP ON	Displays or hides Channel 1

Table III.1: Selected SCPI Commands Used in the Project

The SCPI commands shown in **Table III.1** represent a selection of the most relevant instructions used to interact with the GDS-2204A oscilloscope through the developed web interface.

Each command was carefully tested to ensure correct transmission and response between the frontend (browser) and the backend (Python script).

These commands were mapped to specific buttons or controls in the HTML interface, allowing real-time interaction with the physical device.

III.5 Code Structure and Command Flow

In this section, we present the overall structure of the source code used in the system, along with the detailed flow of SCPI command execution, from the user interface to the oscilloscope. Each layer of the system is explained with key examples

III.5.1Python Script for Sending SCPI Commands

scpi_client.py > ...

```python
# scpi_client.py (debug version)
import socket
import sys

def send_scpi_command(ip, port, command):
    print(f"Sending to {ip}:{port} => {command}")
    try:
        with socket.create_connection((ip, port), timeout=2) as sock:
            sock.sendall((command + '\n').encode())
            sock.settimeout(1.5)
            try:
                response = sock.recv(1024).decode().strip()
                return response if response else "Command sent (no content)."
            except socket.timeout:
                return "No response (timeout)."
    except Exception as e:
        return f"Connection failed: {str(e)}"

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Missing command.")
        sys.exit(1)

    ip = "192.168.1.9"
    port = 5025
    command = sys.argv[1]
    print(send_scpi_command(ip, port, command))

```

Figure III.3: Python Script for Sending SCPI Commands to the Oscilloscope

The Python script above is responsible for establishing a TCP/IP connection with the GDS-2204A oscilloscope and sending SCPI commands to it. It uses the built-in socket module to create a connection and the sys module to retrieve the command passed from the PHP script via command-line argument.

Here's a breakdown of the key parts of the script:

import socket: Imports the module required to create and manage TCP/IP socket connections.

import sys: Allows the script to access command-line arguments, such as the SCPI command sent from the web interface.

def send_scpi_command(ip, port, command): Defines a function that sends a SCPI command to the oscilloscope at the specified IP and port.

socket.create_connection((ip, port), timeout=2): Opens a TCP connection to the oscilloscope with a 2-second timeout for connection establishment.

sock.sendall((command + '\n').encode()): Sends the SCPI command, encoded in bytes, and appends a newline as required by the instrument.

sock.settimeout(1.5): Sets a timeout for receiving a response to avoid indefinite waiting.

sock.recv(1024).decode().strip(): Attempts to read the response from the oscilloscope, decodes it from bytes to string, and removes whitespace.

return response if response else "Command sent (no content).": Returns the response if available; otherwise, it indicates the command was sent but no content was returned.

except socket.timeout: Handles the case where no response is received within the set timeout.

if _name_ == "_main_": Ensures that this block runs only when the script is executed directly (not when imported as a module).

command = sys.argv[1]: Retrieves the SCPI command from the command-line arguments passed by the PHP script.

Print (send_scpi_command(...)): Executes the function and prints the result so it can be read by the calling PHP script.

This script plays a critical role in the backend of the project, as it acts as the communication bridge between the web interface and the physical oscilloscope.

III.5.2 PHP Backend Bridge

send_command.php

```php
<?php
if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST["command"])) {
    $command = escapeshellarg(arg: $_POST["command"]);
    $output = shell_exec(command: "python scpi_client.py $command 2>&1");
    echo "<pre>$output</pre>"; // pour afficher toute la sortie
    exit;
}
?>
```

Figure III.4: PHP Code Handling SCPI Commands and Invoking Python Script

This PHP script is responsible for receiving SCPI commands sent via a POST request from the frontend (JavaScript).

It first checks that the HTTP method is POST and that the command field exists.

The command is then safely escaped using escapeshellarg() to prevent shell injection.

The script then calls the Python file scpi_client.py, passing the command as an argument.

The full output of the Python script is captured (including any errors, via 2>&1) and displayed inside a <pre> tag to preserve formatting.

Finally, the script exits after displaying the result.

III.5.3 JavaScript Function for Sending SCPI Commands

```
function sendCommand(command) {
  const xhr = new XMLHttpRequest();
  xhr.open("POST", "send_command.php", true);
  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  xhr.onload = () => {
    const resp = xhr.responseText.trim();
    document.getElementById("status").innerText = resp !== "Response:" ? resp : "";
  };
  xhr.onerror = () => {
    document.getElementById("status").innerText = "Error sending command.";
  };
  xhr.send("command=" + encodeURIComponent(command));
  console.log("SCPI Sent:", command);
}
```

FigureIII.5: JavaScript Function to Send SCPI Commands via AJAX(XHR)

This function sends a SCPI command from the frontend to the backend using an XHR.

It opens a POST request to send_command.php, sets the appropriate content type, and sends the encoded command as a parameter.

When a response is received, it trims the result and displays it inside an HTML element with the ID status.

If the response is exactly "Response:", nothing is shown.

If an error occurs during the request, an error message is displayed.

The function also logs the command to the browser console, which helps with debugging during development.

III.5.4 HTML Interface Trigger

```
<div class="control-buttons">
  <button class="default" onclick="sendCommand(':AUTOSet')">AutoSet</button>
```

Figure III.6: HTML Button Interface Triggering the AutoSet Command

This HTML snippet displays a button labelled "AutoSet" inside a container with the class control-buttons.

The button has a CSS class called default for styling purposes.

When the user clicks the button, it calls the JavaScript function sendCommand() and passes the SCPI command :AUTOSet as an argument.

This triggers the entire backend process, eventually sending the command to the GDS-2204A oscilloscope and returning the result to be displayed on the interface.

III.5.5 CSS Styling for the AutoSet Button

```
.control-buttons button:nth-child(1) {
  background-color: #0066cc;
  color: white;
  border-radius: 30px;
}
```

Figure III.7: CSS Styling for the AutoSet Button in the Control Panel

This CSS rule targets the first button inside any container with the class control-buttons.

In this case, it is used to style the "AutoSet" button.

The background color is set to a deep blue (#0066cc) for visibility, and the text color is white for contrast.

The border-radius is set to 30px to give the button a pill-like, rounded appearance, enhancing the visual design and making it more user-friendly.

III.6 Complete Execution Flow of the: AUTOSet Command

To illustrate the complete operational flow of the system, we provide here a complete explanation of how the :AUTOSet SCPI command is executed when the user interacts with the interface.

1. User interaction: The user clicks on the "AutoSet" button on the web interface. This button is written in HTML and triggers a JavaScript function when clicked.

2. JavaScript request: The sendCommand(':AUTOSet') function is called. It creates an XHR and sends the command to the backend via a POST request to send_command.php.

3. PHP bridge: The PHP script receives the command using the $_POST["command"] variable. It then securely passes the command to a Python script using shell_exec().

4. Python execution: The Python script receives the command as an argument, connects to the oscilloscope using TCP/IP via sockets, sends the SCPI command, and waits for a response.

5. Oscilloscope response: The oscilloscope processes the :AUTOSet command, adjusts the settings automatically, and sends a response (if any).

6. Displaying the result: The Python script returns the result to the PHP script, which then echoes it. The JavaScript function captures the result and displays it in the browser interface.

This sequence demonstrates how all components of the system—from frontend to backend—communicate in real time to control a real instrument remotely.

III.7 Test Experiment

To test how well the system works, in this chapter, we will present the different states that the application we developed can have during execution.

The photo below (Figure III.8) shows the oscilloscope that we are controlling remotely connected to a circuit (boost convert).

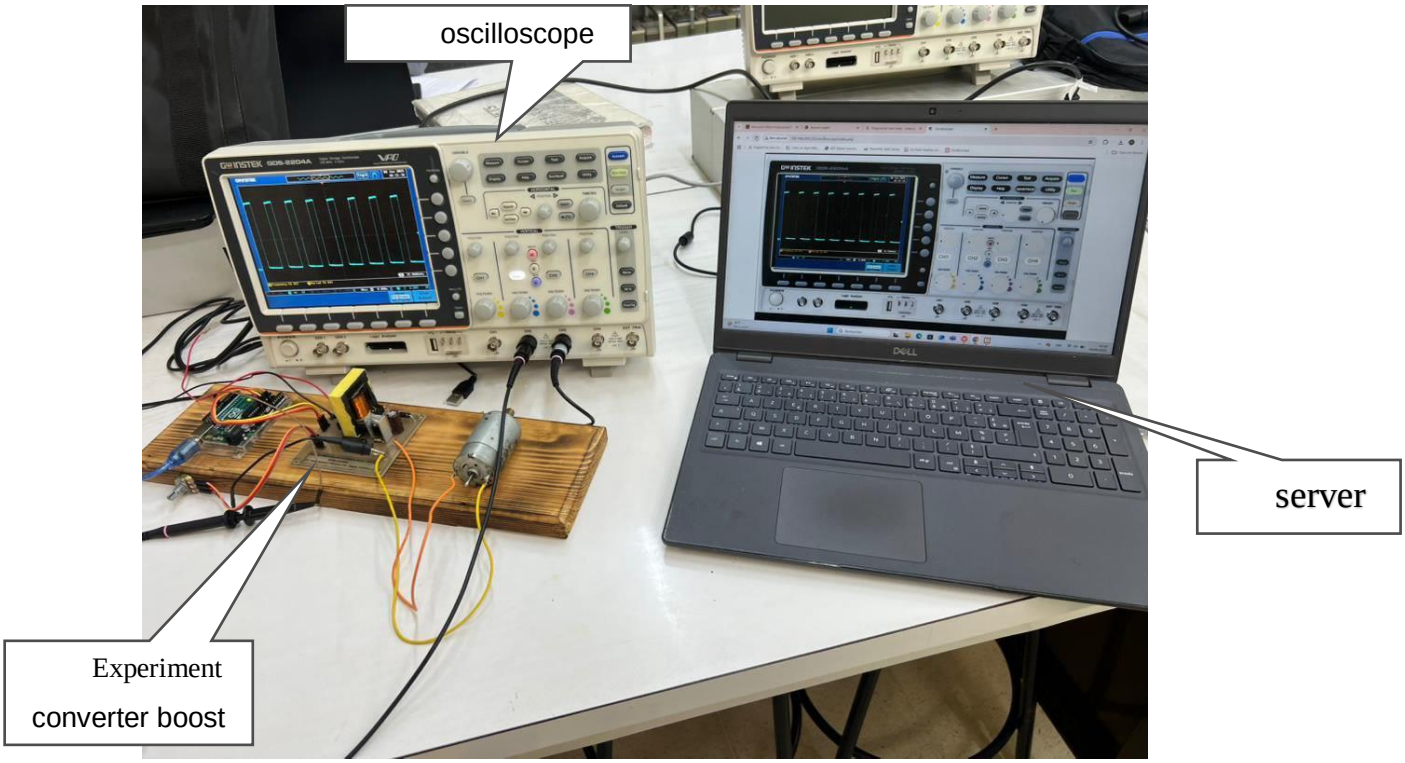

Figure III.8: laboratory experiment

This confirmed that the connection between the interface and the oscilloscope is responsive and works exactly as expected. It also demonstrated that the system can be used to remotely monitor and control real signals just like being physically present in the lab.

Another test for the web site:

- Page login: The entrance to the façade is oscilloscope (Figure III.9).

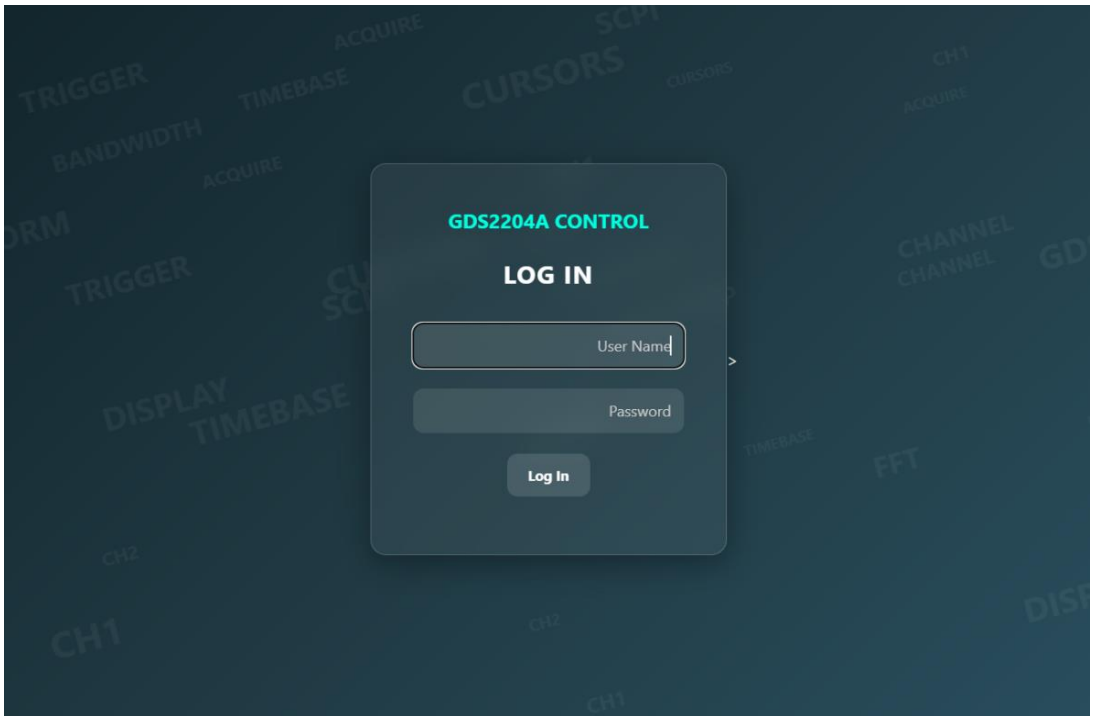

Figure III.9: page login

➢ Stat 1: The initial state of the application made after its launch (Figure III.10).

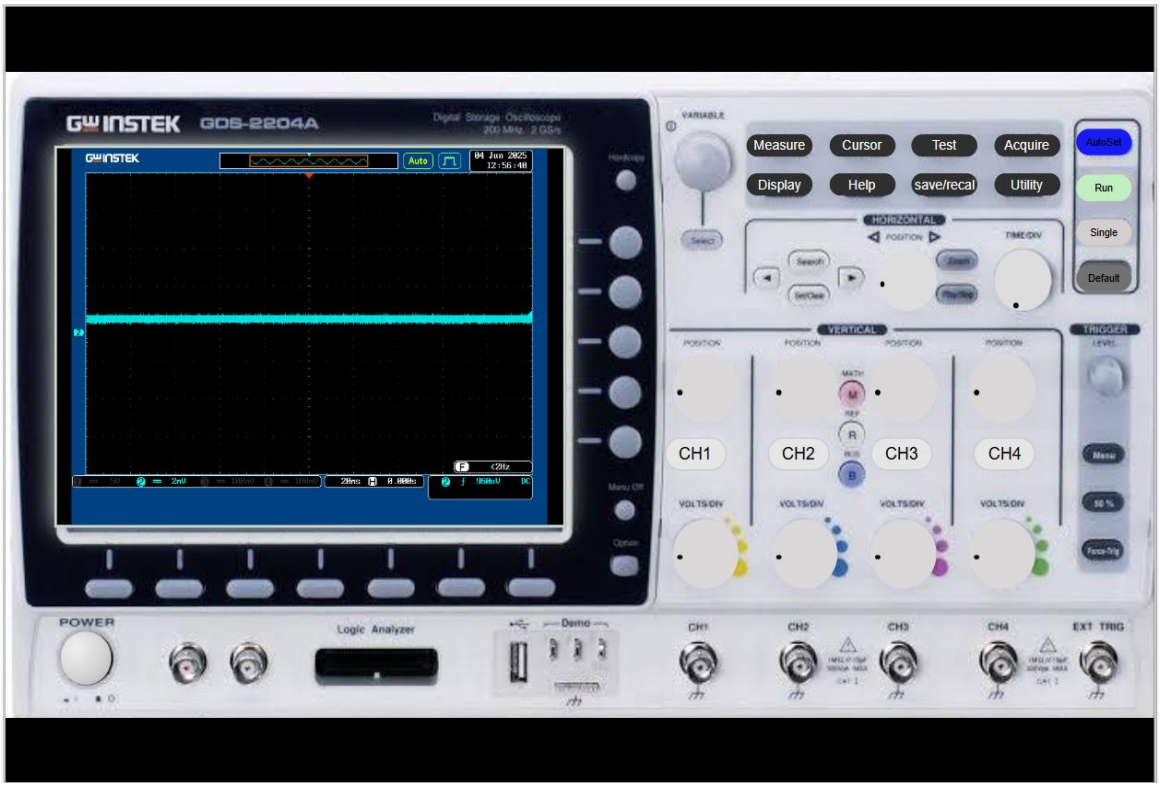

Figure III.10: initial state

➢ **Run/Stop button:** to start or stop the signal display (FigureIII.11).

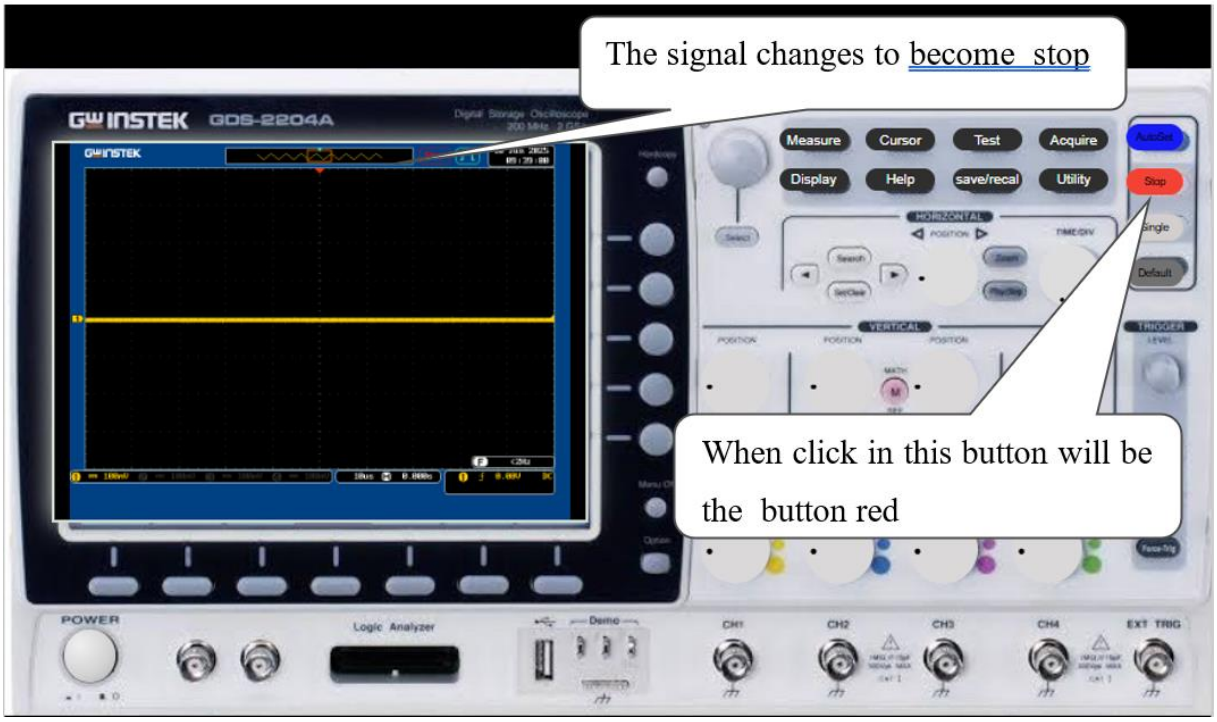

Figure III.11: state button stop

➢ **Channel 1-2- buttons:** to activate or deactivate the channels. (Figure III.12).

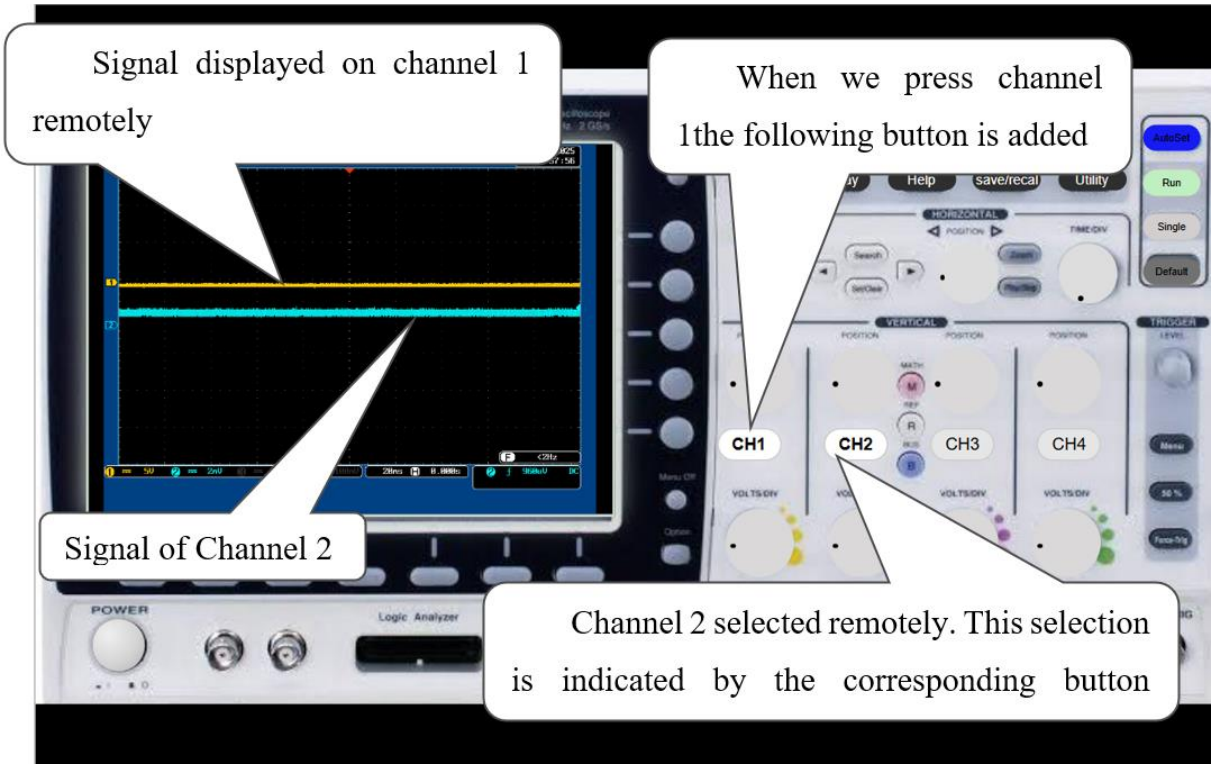

Figure III.12: state button channel

➢ **Horizontal Position knob:** to adjust the signal's position on the screen (Figure III.13).

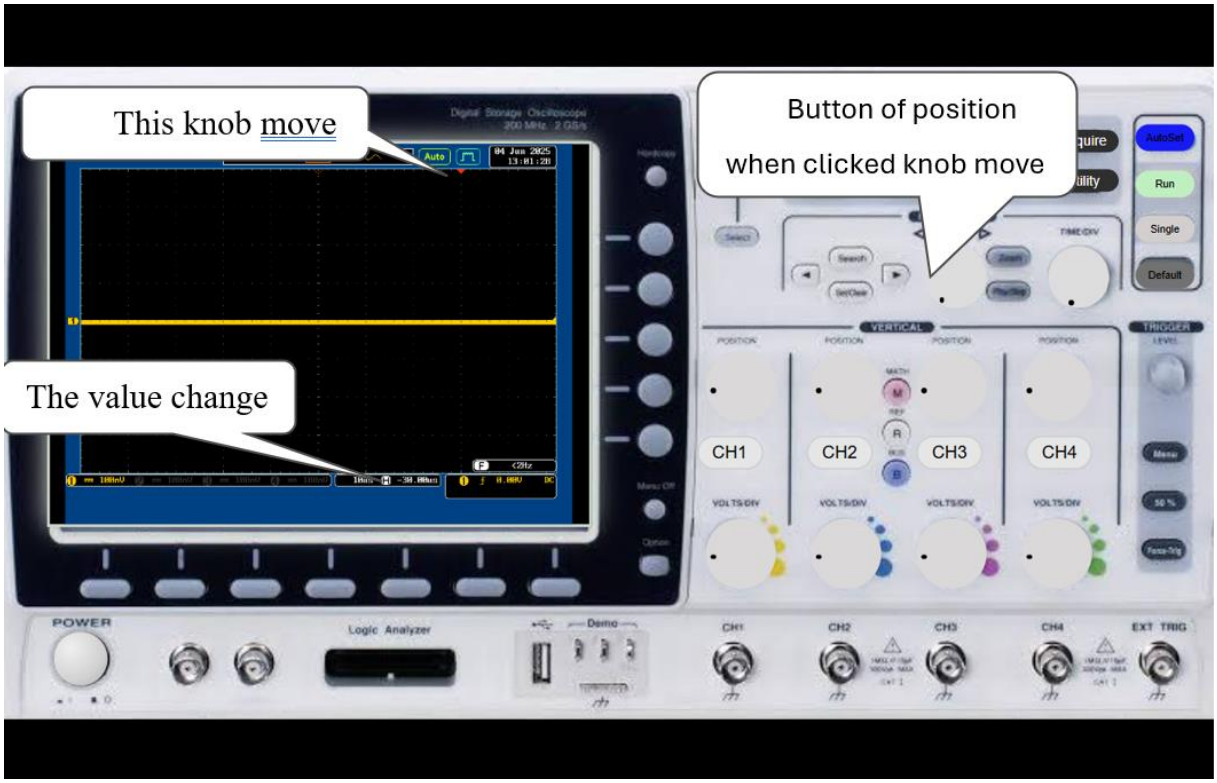

Figure III.13: state button position

III.8 Command Logging Feature

One of the important features we added to the system is session logging. Each session generates a daily log file stored in a dedicated logs folder. This file records all SCPI commands sent, and all responses received from the oscilloscope, along with timestamps.

This logging system is extremely useful for analysing user actions, troubleshooting issues, or reviewing the usage history – especially in educational or multi-user environments.

III.9 Project Evaluation and Suggested Improvements

III.9.1 Overall Assessment

Here are the main achievements of the project:

- Full remote control via browser.
- Direct execution of SCPI commands.
- Automatic logging of all activity.

- Fast and responsive communication with the oscilloscope

III.9.2Connection Setup

Instead of using the traditional RS232 port, the system now uses a LAN (Ethernet) connection. This change made the setup easier and significantly faster when sending commands to the oscilloscope.

III.10 Future Improvements:

III.10.1 Real-Time Signal Display Using Web Socket

Currently, the signal display is updated by sending commands one by one. In future versions, I plan to implement Web Socket technology to provide live, real-time signal streaming without delays. This would make the experience much more fluid and closer to how a real oscilloscope works in real time.

III.10.2 Expanding the System to Support Multiple Lab Devices

One of my goals for the future is to expand the system into a complete lab platform that supports various instruments – not just the oscilloscope. For example, the user could choose from a list of devices like a signal generator, mustimeter, or power supply. Once selected, the interface would load the specific controls and commands related to that device. This would turn the system into a centralized and flexible lab management tool for both learning and practical use.

III.11 Conclusion

This chapter presented the practical implementation of the remote oscilloscope control system. The system uses HTML/CSS/JS for the front end, PHP as middleware, and Python for SCPI communication. It enables full control of the GDS-2204A oscilloscope over a LAN connection. Experiments confirmed the system's reliability and responsiveness. Key features include channel control, signal positioning, and command logging. Future improvements include real-time display via WebSocket and multi-device support.

GENARAL CONCLUSION

General conclusion

At the end of the Master's project in Electrical Engineering (Electrical Control), we successfully developed and implemented a complete remote-control solution for the GDS-2204A digital oscilloscope. The project met the needs for efficient, flexible and user-friendly measurement instrument control in modern industrial and scientific environments.

The solution incorporates modern web technologies such as HTML, CSS, and JavaScript for the front-end user interface, combined with PHP and Python for back-end processing and communication management. The overall system architecture follows a client-server model based on (MVC) design pattern. This architecture ensures separation of responsibilities and improves system maintenance, scalability, and ease of development.

A key outcome of the project was the creation of a powerful, ergonomic, and intuitive web interface that allows users to remotely control the oscilloscope through any standard web browser. This eliminates the need for dedicated client software and improves accessibility. The interface supports basic oscilloscope controls such as channel activation, voltage/div setting, time base adjustment, and waveform acquisition. These commands are translated into SCPI commands. Messages and transmitted over TCP/IP networks, ensuring compatibility with industrial communication standards.

The backend component, primarily implemented in PHP and Python, acts as a bridge between the frontend interface and the oscilloscope hardware. PHP scripts handle incoming user commands and relay them to the Python SCPI client, which manages low-level communication with the oscilloscope via network sockets. This design leverages Python’s flexibility and powerful libraries for instrument control, while PHP provides a reliable web communication interface.

From a learning point of view, this venture has considerably upgraded our specialized competencies over numerous spaces. Selected up down to earth involvement in meddle with mechanical hardware utilizing standard communication conventions, particularly SCPI over TCP/IP. Moreover, we created capability in object-oriented programming, web improvement advances, and applying program building standards through UML and the MVC design. This comprehensive introduction bridges hypothetical information with hands-on application.

This work combines the concepts of centralized electrical building control with advanced computer programming methods and represents a breakthrough in our scientific development.

General conclusion

In this framework as a springboard for further developments in inaccessible observation and control. Possible future directions include extending this approach to other mechanical systems, integrating advanced data analytics, and applying it to smart monitoring systems based on the Internet of Things.

Finally, this project brings together all the basic ideas and concepts needed to quickly start a project to control systems remotely over the Internet. It lays a solid foundation and serves as a forerunner for engineering and research projects in the same direction.

In summary, this work not only achieved its basic goals, but also provided insightful information and practical tools that can promote further developments in industrial remote control and instrumentation.

Annexes:

List of command [31]

Command	description
*** IDN?**	Returns the manufacturer, model, serial number and version number of the unit.
*** RST**	Resets the GDS-2204a (recalls the default panel settings).
: AUTOSet	Runs the Autoset function to automatically configure the horizontal scale, vertical scale, and trigger according to the input signal.
: CHANnel<x>: COUPling	Selects or returns the coupling mode
: CHANnel<x>: DISPlay	Turns a channel on/off or returns its status
: CHANnel<>: IMPedancee	Sets the impedance of the oscilloscope.
: CHANnel<x>: Expand	Sets Expand from ground or from center for a channel. Queries the Expand status of a channel: Sets the impedance of the oscilloscope.
: CHANnel<x>: INVert	Inverts a channel or returns its status
: CHANnel<x>: POSition	Sets or returns the position level for a channel. Note, the vertical position will only be set to closest allowed value. The position level range depends on the vertical scale.
: CHANnel<x>: SCALe	Sets or returns the vertical scale. The scale depends on the probe attenuation factor. Note the probe attenuation factor should be set before the scale.
: CURSor: MODe	Sets cursor mode to horizontal (H) or horizontal and vertical (HV).
: CURSor: SOURce	Sets or queries the cursor source
: CHANnel<x>:	Sets or returns the probe type (voltage/current).

PROBe: TYPe	
: CHANnel<x>: PROBe: RATio	Sets or returns the probe attenuation factor. Same as: hannelkey → variable knob
: CHANnel<x>: DESKew	Sets the deskew time in seconds
: Run	The run command allows the oscilloscope to continuously make acquisitions (equivalent to pressing the Run key on the front panel).
: STOP	The stop command stops the oscilloscope making further acquisitions (equivalent to pressing the Stop key on the front panel).
: SINGle	The single command allows the oscilloscope to capture a single acquisition when trigger conditions have been fulfilled. (Equivalent to pressing the Single key on the front
: FORCe	The Force command forces an acquisition. (equivalent to pressing the Force key on the front panel).
: WINDow: SOURce	Sets or queries which window is the active window in split screen mode
: TIMebase: POSition	Sets or queries the horizontal position
: TIMebase: SCALe	Sets or queries the horizontal scale
: TIMebase: MODe	Sets or queries the time base mode. The time base mode determines the display view window on the scope
: TIMebase: WINDow: POSition	Sets or queries the zoom horizontal position.

References:

[1] A. Abdu et al., "Remote Laboratory Implementation for Oscilloscope Using Web Interface and SCPI Commands," 2020 International Conference on Smart Technology & Applications (ICoSTA), 2020, pp. 1-6. doi: 10.1109/ICoSTA48221.2020.1570610417.

[2] PBS, Master of Lightning: Remote Control - Tesla. [Online]. available: https://www.pbs.org/tesla/ins/lab_remotec.html. (04/2025)

[3] K. Goldberg, The Telegarden Website. [Online]. Available: https://goldberg.berkeley.edu/garden/Ars/ . (03/2025)

[4] "How do IIoT solutions facilitate remote monitoring and control of industrial machinery?" [Online]. Available: https://www.indmallautomation.com/faq/how-do-iiot-solutions-facilitate-remote-monitoring-and-control-of-industrial-machinery/ (04/2025)

[5] E. Sousa, B. Cunha, M. Piedade, and E. Morgado, "On a web-based oscilloscope interface app for e-learning: Software architecture, practical applications, and user experience," Informatics, vol. 7, no. 1, p. 19, 2022.

[6] Larbi, K. (2014). Développement d'une application client-serveur de gestion de paie (Mémoire de licence, Université Abou Bekr Belkaid – Tlemcen, Algérie).

[7] T. ZEMOUL and K. Aid, (2014) "Conception et Réalisation d'une Application Client/Server 3-tiers. Cas: Gestion du Personnel des IMPÔTS de Tizi-Ouzou," Université Mouloud Mammeri de Tizi-Ouzou.

[8] Sayah, T. (2018). Conception of a client/server system with encrypted data exchanging using sockets (master's thesis, Mohamed Khider University of Biskra, Algeria).

[9] Brinis, T. (2013). Control via Internet (TCP/IP) (master's thesis, Mohamed Khider University of Biskra, Algeria).

[10] Cisco. Understanding IPAddressing. [Online]. Available at: https://www.cisco.com/(04/2025)

[11] "The ultimate guide to TCP/IP," [Online]. Available: https://www.comparitech.com/net-admin/ultimate-guide-tcp-ip/ (05/2025)

[12] Benmahbous, A., & Rouabah, E. (2020). Réalisation d'une application web de gestion de cabinet d'avocat (Mémoire de Master, Université Mohammed El Bachir El Ibrahimi – BBA, Algérie).

References

[13] "La sécurité des communications par satellite," [Online]. Available: https://www.areion24.news/2023/10/23/la-securite-des-communications-par-satellite-une-infrastructure-critique-pour-la-transmission-de-donnees/ (05/2025)

[14] Birem, M., & Benazzouz, K. (2020). [La réalisation d'une plateforme d'évaluation Etude de cas: Une application de suivi pour les patients autistes] (Mémoire de Master, Université de M'Sila, Algérie).

[15] Ait Hammou, A. A., & Belkheir, H. E. (2018). Développement d'une interface Web pour la manipulation à distance d'une plate-forme d'instruments de mesure (Mémoire de Master, Université Ibn Khaldoun de Tiaret, Algérie).

[16] Amrani, I., & Doumi, A. (Année). Développement d'un ERP pour la gestion des échanges de médicaments entre pharmaciens (Mémoire de Master, Université Abou Bekr Belkaïd – Tlemcen, Algérie).

[17] Matoug, B., & Kafnemer, Y. (2022, June 27). Développement d'une application mobile pour le service de tourisme, cas d'étude « Wilaya de Tlemcen » [Mémoire de Master, Université Aboubakr Belkaïd

[18] Farah, S., Benachenhou, A., Neveux, G., & Barataud, D. (n.d.). Description of a flexible software architecture for remote control of instruments with RS232, USB and GPIB interfaces. LEOG, University of Mostaganem, Algeria; XLIM, University of Limoges, France.

[19] Cloudflare. (n.d.). Client-Server vs. Peer-to-Peer: What's the difference? Cloudflare. Retrieved June 10, 2025, from https://www.cloudflare.com/learning/ddos/glossary/client-server-vs-peer-to-peer

[20] Sung, J.-Y., Jeong, J.-Y., & Yoon, K.-S. (2006, February 20–22). DRM enabled P2P architecture. Presented at the UST–ETRI Digital Contents Conference, Daejeon, South Korea.

[21] L. Zi-Wei, L. Shao-Bin, L. Yan, and L. Hui-Yong, "Remote ATS Simulation System Based on WebSocket Communication Protocol," 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA).

[22] J. Namjoshi and A. Gupte, "Service Oriented Architecture for Cloud Based Travel Reservation Software as a Service," 2009 IEEE International Conference on Cloud Computing, Bangalore.

[23] Université Mouloud Mammeri de Tizi Ouzou. (2020, October 8). Conception et réalisation d'un oscilloscope numérique avec le microcontrôleur STM32 [master's thesis, Université Mouloud Mammeri de Tizi Ouzou, Faculté de génie électrique et informatique, Département informatique].

References

[24] Bensattalah, A. (2021). Comparison and performance evaluation of ATM, FR, and MPLS wide area networks (master's thesis, University of Ibn Khaldoun – Tiaret, Algeria).

[25] "Comprendre l'interface utilisateur graphique (GUI)," [Online]. Available: https://commentouvrir.com/technologie/comprendre-linterface-utilisateur-graphique-gui/ (05/2025)

[26] Larouci, B., & Meddahi, I. (2017). Conception et développement d'une application médicale distribuée à l'aide d'un composant Enterprise JavaBeans (EJB) [Mémoire de Master, Université Abou Bakr Belkaid – Tlemce).

[27] "The effects of the AJAX technology on web application usability," [Online]. Available:https://www.researchgate.net/publication/251179041_The_effects_of_the_ajax_technology_on_web_application_usability. (05/2025)

[28] "AJAX or the new standard for faster web interfaces," [Online]. Available: https://www.researchgate.net/publication/259339601_AJAX_or_the_new_standard_for_faster_web_interfaces (05/2025)

[29] Cours et tutoriels," [Online]. Available: https://apcpedagogie.com/cours-et-tutoriels/ (05/2025)

[30] "Standard Commands for Programmable Instruments (SCPI)," [Online]. Available: http://www.ivifoundation.org/scpi/ (05/2025)

[31] "PySerial documentation," [Online]. Available: https://pyserial.readthedocs.io/ (05/2025)

[32] L. Guili and K. Quancun, "Design of virtual oscilloscope based on GPIB interface and SCPI," 2013 IEEE 11th International Conference on Electronic Measurement & Instruments, 2013.

[33] Digital Storage Oscilloscope GDS-2000 series. Programming manual.www.gwinstek.com

[34] https://www.php.net/manual/en/book.dio.php (02/2025)

[35] **https://www.apachefriends.org/docs/ (02/2025)**