الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Supérieure de Technologie

Département de Génie Industriel et Maintenance

Mémoire de fin d'études en vue de l'obtention du diplôme de

Ingénieur d'état

Spécialité

Mécatronique

Thème

Elaboration d'une plateforme des travaux pratiques de physique fondamentale dans le département de formation préparatoire

Réalisé par

SALAG Aicha &

HAMACHE Ghada

Les membres de Jury :

Président	MOHAMADI Said	Maitre Assistant – Classe A
Promoteur	KERROUCHI Slimane	Maitre de conférences – Classe A
Co-encadrant	CHERIFI Tarek	Maître de conférences – Classe B
Examinateur	GOUICEM Ali Mohamed Tahar	Maitre de conférences – Classe B

Année universitaire 2021 -2022

Acknowledgements

First of all, All Praise to ALLAH the almighty for his blessing, thus we were able to perform this dissertation project.

Extreme gratitude to our advisors *Mr. Kerrouchi* and *Mr. Cherifi* for giving us the opportunity to implement this platform, and for having spared no effort to guide us throughout this work with sincerity and patience.

We would like also to thank Mr. Chaterbache, the head of the preparatory formation department, and the physics laboratory engineer for giving us access to the physics laboratory and thus we were able to perform our experiments.

All thanks are addressed to the pedagogical staff of ENST, and particularly to our GIM department.

Finally, we would like to thank every single person who helped and supported us to accomplish this work.

Abstract

The goal of this work is the implementation of a platform for practical work in fundamental physics. The platform is made for educational purposes, it allows the user to highlight phenomena of fundamental physics through an innovative method allowing him to carry out his own experiment and to visualize the results thus obtained on his own smartphone or on a web page over a wireless connection. For that two solutions have been implemented, for local data acquisition in which Bluetooth communication is used, and for remote data acquisition in which WiFi communication is used.

Key words : Mechatronics for education, fundamental physics experiment, local data acquisition, Remote data acquisition, Bluetooth communication, WiFi connection

Résumé

L'objectif de ce travail est la mise en place d'une plateforme de travaux pratiques en physique fondamentale. La plateforme est faite à des fins pédagogiques, elle permet à l'utilisateur de mettre en évidence des phénomènes de physique fondamentale grâce à une méthode innovante lui permettant de réaliser sa propre expérience et de visualiser les résultats ainsi obtenus sur son propre smartphone ou sur une page web via une connexion sans fil . Pour cela, deux solutions ont été mises en œuvre, pour l'acquisition de données locale dans laquelle la communication Bluetooth est utilisée, et pour l'acquisition de données à distance dans laquelle la communication WiFi est utilisée.

Mots clés : Mécatronique pour l'éducation, expériences de physique fondamentale, acquisition de données locales, acquisition de données à distance, communication Bluetooth, connexion WiFi.

ملخص

الهدف من هذا العمل هو إنشاء منصة للعمل العملي في الفيزياء الأساسية. المنصة مصممة للأغراض التعليمية ، فهي تتيح للمستخدم تسليط الضوء على ظواهر الفيزياء الأساسية بفضل طريقة مبتكرة تسمح له بتنفيذ تجربته الخاصة وتصور النتائج التي يتم الحصول عليها بهذه الطريقة على هاتفه الذكي أو على صفحة ويب عبر اتصال لاسلكي . لهذا ، تم تنفيذ حلين ، للحصول على البيانات المحلية التي يتم فيها استخدام اتصال Bluetooth ، ولحصول البيانات عن بُعد التي يتم فيها استخدام اتصال WiFi.

الكلمات المفتاحية

الميكاترونكس للتعليم ، تجارب الفيزياء الأساسية ، الحصول على البيانات المحلية ، الحصول على البيانات عن بعد ، اتصال WiFi، اتصال Bluetooth.

Contents

1 General introduction **1**

2 Mechatronics for didactic solutions - An overview **3**

2.1 Introduction 3

2.2 Low-cost Physics experimental Kit 4

2.3 The effect of heat on resistance 5

2.4 Educational platforms 6

2.5 Smartphone and Arduino for simple and basic experiments 9

2.5.1 Cart movement study using Smartphone light sensor 9

2.5.2 Cell current-voltage characteristic measurement 10

2.5.3 Pendulum experiment with light-based timer 11

2.6 Description of the physics platform 13

2.7 Conclusion 14

3 Design of the fundamental physics platform **15**

3.1 Introduction 15

3.2 General overview of the system 15

3.3 Hardware Description 16

3.3.1 ESP32 MCU module 17

3.3.2 Micro-SD card and Micro-SD card module 18

3.3.3 Arduino bluno 18

3.3.4 Ultrasonic sensor for proximity measuring . . . 19
3.3.5 Liquid Crystal Display "LCD" . . . 21
3.4 Software description . . . 21
3.4.1 MIT App Inventor platform . . . 22
3.4.2 Bluetooth protocol . . . 22
3.4.3 Python Integrated Development Environment (IDL) . . . 22
3.4.4 HTML . . . 23
3.4.5 CSS . . . 24
3.4.6 JavaScript . . . 26
3.4.7 HTTP protocol . . . 27
3.5 Conclusion . . . 31

4 Tests and results **32**

4.1 Introduction . . . 32
4.2 Pendulum experiment for the study of oscillations . . . 32
4.2.1 Local data acquisition . . . 34
4.2.2 Test and results of local data acquisition . . . 36
4.2.3 Remote data acquisition . . . 40
4.2.4 Test and results of remote data acquisition . . . 44
4.3 Inclined Plane for the study of an accelerated movement Experiment . . . 47
4.3.1 Test and results of local data acquisition . . . 48
4.3.2 Test and results for Remote Data Acquisition . . . 52
4.4 Charge/Discharge of capacitor . . . 56
4.4.1 Schematic of RC circuit with Arduino board . . . 56
4.4.2 Local data acquisition . . . 58
4.4.3 Test and results of local data acquisition . . . 59
4.5 Conclusion . . . 63

5 General conclusion **64**

A Appendix **66**

A.1 Arduino code 66

A.1.1 Arduino code for the pendulum experiment for remote data acquisition 70

A.1.2 Arduino code for the inclined plane experiment for local data acquisition 76

A.1.3 Arduino code for the inclined plane for the remote data acquisition . 80

A.2 HTML code 88

A.2.1 HTML code for the home page 88

A.2.2 HTML code for pendulum experiment page 89

A.2.3 HTML code for inclined plane experiment 89

A.3 CSS code 90

A.4 JavaScript code 93

A.4.1 JavaScript code of the home page 93

A.4.2 JavaScript code for the pendulum oscillation chart 93

A.4.3 JavaScript code for drawing the chart 95

Bibliography **98**

Chapter 1

General introduction

We are all aware of the role of mechatronics in modern industries, and how it contributes to enhancing the industrial facility by providing new systems that are optimized in terms of energy consumption, and size and capable of ensuring their functions efficiently. However, the domain application of mechatronics is wider than the industrial facilities or modern systems.

Unexpectedly, education could be one of the mechatronics application domains. The main idea serves to provide the education institutes with a new generation of material, especially experimental setups, in which modern technologies are used. Indeed, mechatronics for education is not that recent thematic.For instance, in [1], they implemented a platform based on mechatronics to enhance the interest of students in high school in engineering. Moreover, a new learning approach has emerged recently such as the case in [2] and [3] serves to let students work on or with mechatronics systems even if they are not mechatronic students, this approach allows students to learn multiple subjects at once.

Nevertheless, the contribution of mechatronics is not only through complicated systems. Mechatronics can provide a simple and a reliable alternative to the conventional experimental setups for fundamental sciences such as physics.

This approach of integrating new technologies in physics experiments has become very in-

teresting for students, educators and researchers [4]. It has received much praise, especially from students because it allows them to interact with the modern technologies that are obsessed with while maintaining the main objective and achieving better results. In addition, more to the notions of physics that they acquire, they will be introduced to more information and knowledge.

The excellent results of using modern technologies are not the single advantage. Conventional physics experiment setups are usually expensive while implementing such new systems can be accomplished using less pricey material with even more accurate measurement results.

In the same context, we are presenting this work in order to contribute to enhancing the experience of students in fundamental physics experiments. This work serves to implement a platform for physics experience that allows students to receive the measurement data on their own smartphone by using Bluetooth communication. This platform can allow students to eliminate human errors that are usually frequent in any experience.

This work is organized as follows; the first chapter is an overview of the role of mechatronics for didactic solutions, as well as a state of the art on the solution for physics experiments. The second chapter is dedicated for the used hardware description, and the software used while implementing this platform. The third chapter discuss the implementation steps as well as the test results. Finally, the the results are summarized in the general conclusion.

Chapter 2

Mechatronics for didactic solutions - An overview

2.1 Introduction

The implementation of physics experiments using micro-controllers, sensors, and phones is not only used to attract student attention but is more likely to illustrate the relationship between theoretical subjects acquired and their application in real-life and engineering fields. Actually using electronic instruments in physics experiments is not a new approach, for example; photo-gates have been used since a long time ago for the free fall, rolling objects, and pendulums. However, students are not apprised of the phenomena behind sensing and are not directly aware of the way that data is received on the main photo-gate hub and control. Therefore, the adoption of new programs and the implementation of new physics platforms have become a severe requirement in order to guarantee effective academic achievements. In this chapter, a set of mechatronic platforms for physics experiments that have been carried out recently are presented.

Talking about mechatronics, we will first talk about the role of mechatronics in educational platforms and enriching the educational results of students.

In light of the development of mechatronics and the evaluation of modern technology day after day that dominates all fields.The possibility of completing models on employing mechatronics and relying on it in the educational field appears to facilitate the process of teaching and provide expertise. Therefore, we find many research and developments in this context, such as virtual and remote laboratories, which are open-source platforms with the first classification in terms of use due to their effectiveness and importance in light of the emergence of the term "remote study". And also the idea of "relying on sensors for smartphones and microcontrollers in carrying out simple, useful and investigative experiments [5].

2.2 Low-cost Physics experimental Kit

Generally, modern physics experimental kits that use computers and more sophisticated measuring tools cost expensive thus, and as an alternative, a low-cost physics experimental kit was implemented and proposed in [6].

Five experiments had been performed in total, simple tools were used for this purpose; a notebook or PC, a solar cell, a laser pointer, a pendulum, and a cardboard comb with wide teeth cut out. The free audio editor (FAE) freeware software was used in order to enhance the weak signal and makes it more visible (and more easily measurable) to the students.

Two experiments were for mechanics; velocity and friction. The first experiment serves to verify the temporal dependence of the velocity of a falling cardboard comb by carrying out a *Free fall* experiment. while the second experiment aims to determine the coefficient of kinetic friction from the acceleration on an inclined plane.

The three other experiments were for acoustics; a demonstration of the sound beats using a Visual analyzer, for this experiment a pair of external speakers, and an external microphone, a determination of the sound beats using glasses, and an experiment was carried out in order to determine the sound speed through the Doppler effect and sound beats.

The performed experiment that used a solar cell and a laser pointer allows a comparable

measurement with professional experimental kits such as Vernier, IP Coach, or Pasco. The implemented experimental kits offer an affordable cost which may represent an alternative to the expensive professional kit for primary and secondary school.

2.3 The effect of heat on resistance

Physics programs usually study factors affecting electrical resistance, such as cross-section, length, type of conductor, etc. Contrary to the factors of length, type, and section of the conductor which are examined and related experiments to these effects are widely practiced in schools, and thus enable students to understand the effect and even become able to apply these factors, heat factor studies are usually not sufficiently conducted, which makes students unable to understand the temperature effects. This last is due to the fact that such studies and experiment require a complex structure of the test equipment such as heating element, current module, and resistance measurement module.
Given all these complications, the researchers considered to implement an economical Arduino-based physics experiment which examines the variation of the resistance of a copper wire with temperature [7].

The experiment starts with preparing an Arduino ohmmeter which is a very simple resistance meter built using a voltage divider, two resistors (one of which is unknown) and an Arduino board. Then the temperature sensor and the copper wire were placed in a beaker of pure water,which in turn was placed on a stove for heating as shown in the figure 2.1 below .During the experiment it was noticed that as the temperature sensor measures the temperature there is a change in resistance.

A harmonious match can be observed between the resistance temperature data obtained in the experiment and the theoretical model as shown in the figure. 2.2.

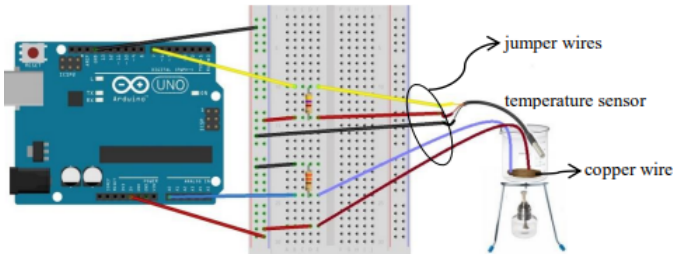


Figure 2.1: Experimental set-up [7].

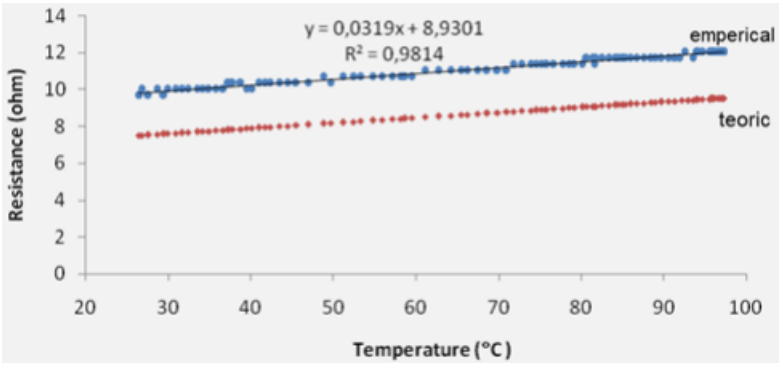


Figure 2.2: Graphs of experimental data and theoretical calculations[7].

2.4 Educational platforms

Equipment lacking and their inefficiency is one of the most important factors affecting the educational attainment of the practical laboratory work of the educational curricula, so many educational strategies can be applied to overcome this issue such as virtual laboratories, video laboratories and laboratory group activities sent to students by regular postal service.

The number of online resources and platforms for teaching vital disciplines is extensive. However, the educational tools available for practical activities in physiology are not common for activities in anatomy and anthropology, virtual atlases and interactive videos for teaching anatomy are effective in promoting participation and learning. Likewise, the virtual microscope slide set and tissue set are an effective tool for distance education and blended learning, whereby students can achieve similar performance to their peers in a face-to-face format. Interestingly, students seem to prefer working online and educational platforms

rather than doing traditional lab experiments[8].

To inspire educators to develop assignments and apply Physics activities to education, here we present the rationale for Mobile Learning Labs (MobLeLabs), a smartphone-assisted learning framework for experiments.

Below there will be a short review of the functions and importance of MobLeLabs.

Mobile learning labs or MobLeLabs are developed for portable devices,such as smartphones and tablets,that work as mini-laboratories for hands-on activities.These learning technologies are smartphone applications where they must be operated by the user to collect physiological data through smartphone sensors. To be classified as MobLeLab, the smartphone's operating system must perform all data acquisition, and therefore must be responsible for creating a number, value, graph or nominal classification according to the collected variable. The idea of using smartphones as laboratory devices is embedded in mobile labs and pocket labs and can be found in brand names such as PocketLab and PhoneLab.

One example of MobLeLab is Google's Science Journal app, which can record data from the physical world, such as light, sound, pressure and motion, and was developed for educational purposes. Likewise, the Phyphox app is a dedicated MobLeLab developed for physics education that includes experimental protocols by default[8].

Variety of sensors that can be used in smartphone-assisted experimentation. Types of built-in sensors that collect data through mobilelearning laboratories (MobLeLabs) are shown in the figure 2.3 below. To continue talking about educational platforms, teaching and separate control systems can be a challenging task. It involves multiple disciplines, such as electronics, physics, programming, and theoretical concepts that use many branches of mathematics. So thanks to 3D printers and Arduino micro-controllers, creating a custom learning control platform is fast, and available. For starters, applying theoretical concepts to a real system is not a trivial task and often requires some references. A good starting point is to search the literature where it is easy to find information on PID controllers. This is by far the easiest and most effective solution to many real-world control problems. A PID controller can be

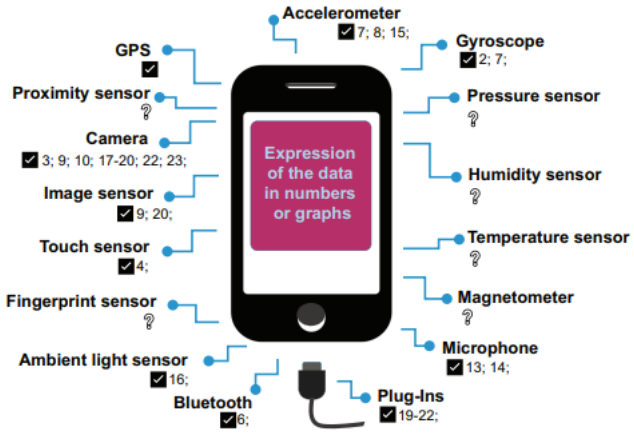


Figure 2.3: Variety of sensors that can be used in smartphone-assisted experimentation[8]

tuned using the famous Ziegler and Nichols method, the magnitude optimum criterion and other techniques summarized but most of them are too advanced for beginners.

From a pedagogical point of view, many articles show how to solve a particular control task, but most of them ignore small details that make it difficult for beginners to reproduce, such as wiring, coding, step-by-step calculations, or sampling methods among others. The proposed low-cost educational platform contains a 3D-printed plastic frame that contains a 12-volt motor, a micro-controller and an electric motor. With this basic setup shown in (fig. 2.4), students have to control the motor's speed and position and evaluate its performance against different scenarios [9].

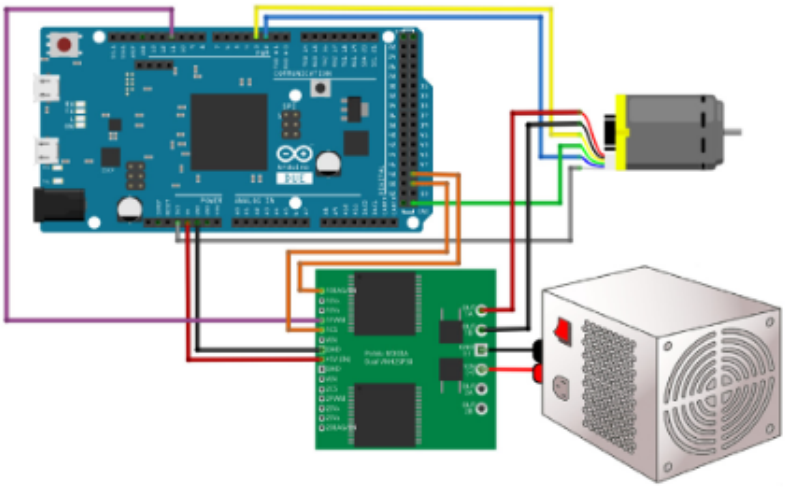


Figure 2.4: The electronic schematic overview [9].

2.5 Smartphone and Arduino for simple and basic experiments

Let's cite some platforms based on the use of a smartphone and an Arduino board to do simple and basic experiments in physics.

2.5.1 Cart movement study using Smartphone light sensor

One of the most important uses of smartphones, in recent times, is taking and recording measurements in various laboratory experiments. This is due to the fact that smartphones contain an accelerometer, a gyroscope, a magnetometer, sensors of pressure, and a microphone. However, the problem posed, in the applications that help to exploit these technologies and go through those experiences, is that they only have the advantage of recording data in the form of a table that displays values. Hence, an application called **phyphox** was created by mechatronics developers and mechatronics principles, that is because it's being a synergistic blend of multiple engineering disciplines and mechatronics makes use of microprocessors, data acquisition sensors, and actuators, all within a Phyphox App, which allows displaying data in the form of curves. The application also provides the feature of sharing and displaying the smartphone screen with the computer in the classroom [10].

To study the cart movement we have a cart that is placed on an inclined table 2m along, a small cart (to be attached to a smartphone), and a pulley of different weights. In addition to an "optical stopwatch" from the **phyphox** app. This watch works if the light sensor in the smartphone senses the intensity of light by a value greater than the value set by the user and stops in the reverse case. The smartphone is placed on the side of the cart so that the screen and the light sensor are facing the lights. The "optical stopwatch" can visualize in real-time for up to 5 consecutive periods.

The lights are separated by a distance of 20 cm between two successive lights to ensure that definite conclusions are obtained on the type of movement along a sufficient path.Initially

the trolley is stationary on the inclined plane due to the equilibrium component of the weight acting parallel to the slope with the tension of the thread, i.e. the total force on the cart is zero.Then push the cart upward which results in uniform linear motion with different speeds as shown in the figure 2.5.

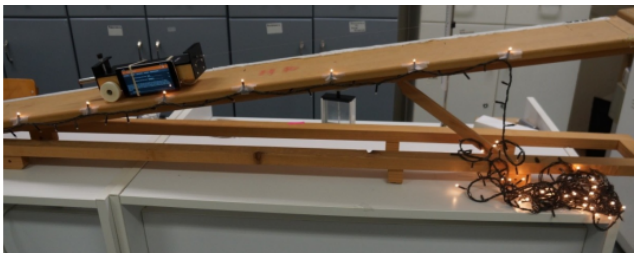


Figure 2.5: The setup of the experiment [10].

2.5.2 Cell current-voltage characteristic measurement

A low-cast physics experiment has been developed and presented in [4], the experiment was implemented using an Arduino-based platform with the aim to facilitate the study of photoresistor cell current-voltage characteristics.

The study is devoted to determining the current-voltage and current-light radiation characteristics through a photoresistor cell based on cadmium sulphide CdS. The experiment setup consists mainly of a light source *6 V/30 W*, a diaphragm, two polarizers, a converging lens *(f = +150 mm*, a photoresistor cell (LDR 05 STE), adjustable generator DC *0-20*, and two multimeters (Fig. 2.6).

The traditional method intends to obtain the current-voltage characteristics for each angle α. Nevertheless, the new method allows data collecting using an Arduino-based platform (Fig. 2.7). For this goal, the electrical elements were replaced by an INA219 DC Current sensor, a DC-DC step converter (2–24 V, 2 A)[1], and an Arduino-Nano board. The DC current sensor INA219 is connected to the terminals of the photoresistor cell and to the branches of an Arduino Nano card,, the circuit is supplied by a DC-DC step converter *(2–24*

[1]DC DC Step Up Boost Converter

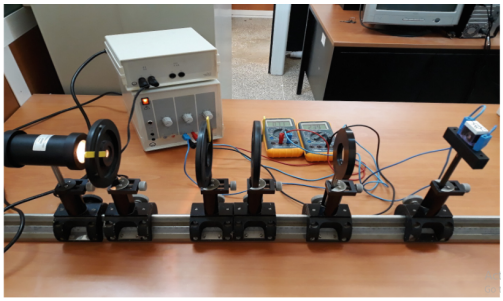


Figure 2.6: Image of the experimental setup without Arduino [4].

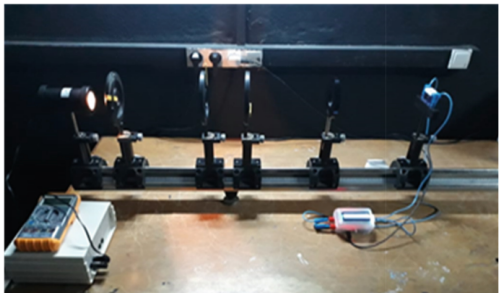


Figure 2.7: Image of the experimental setup with Arduino [4].

V, 2A). An Arduino code was used to retrieve data and an LCD display was used to show data.

The measurement values obtained by the platform presented an excellent agreement with the measurements data obtained by the classical experiment which allowed the verification of Malus law in an innovative way, and with an accessible cost.

2.5.3 Pendulum experiment with light-based timer

A pendulum experimental tool with a light-based timer has been presented in [11], the experiment has been developed to measure the period of a simple pendulum and thus determine gravitational acceleration.

The experiment setup serves to determine the swing period using Excel as a real-time data acquisition tool. The setup consists of an ATMega328 micro-controller, 3DU5C phototransistor, and laser as the source of monochromatic light. To make the system work more efficiently, light is set perpendicularly to the 3DU5C phototransistor. The experimental

setup is shown in figure 2.8. As the pendulum swings, light intensity changes periodically

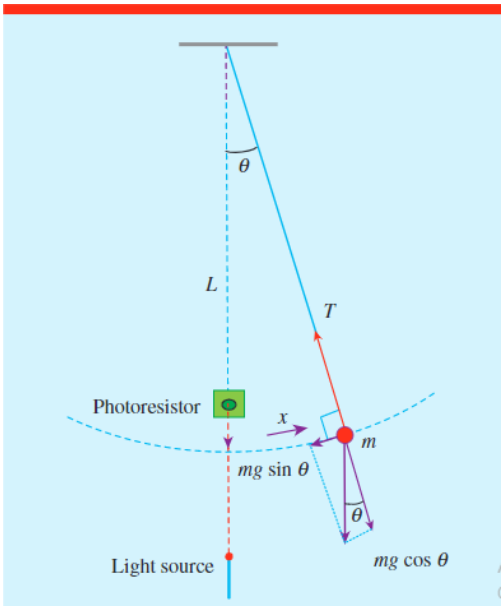


Figure 2.8: Pendulum Experimental setup [11].

when the bob blocks the light. The potential difference value, which is formed from current flows as analog data, will be converted into digital data using an ADC on the microcontroller, figure 2.9 presents the electrical schema of the system. The use of phototransistor allows

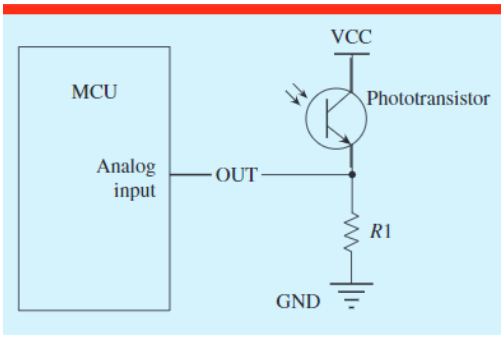


Figure 2.9: Electrical scheme [11].

time measurement when the bob across the light by using brightness function. The analog data is received and stored in the microcontroller, After the data is processed in Arduino, it is recorded in an Excel spreadsheet by using some VBA codes that make it accept raw data from Arduino. A data acquisition toolbox is used, it's used by linking Arduino and Excel by matching the port and then setting the baud rate to "9600 baud".

The system has been tested to time pendulum oscillations with small amplitudes, in relation to varying pendulum lengths. The proposed system works perfectly in real-time acquisition data using Excel. When it is applied to measure the local acceleration of gravity, the obtained values are $9.79\ m.s^{-}2$, $9.79\ m.s^{-}2$, and $9.80\ m.s^{-}2$ for pendulums with 0.9 m, 1.2 m, and 1.5 m in length.

After taking a look at many of the modern methods and platforms adopted in the curricula in physics and looking at their effectiveness in educational settings. We had the opportunity to digitize some physics experiments in the form of an educational platform.

Before describing our physic platform,let us begin by answering some preliminary questions,such as: what is this physic platform,What is the motive for its creation and achievement, what is its importance, and what kind of experiences it contains.

2.6 Description of the physics platform

It is a combination of Fundamental physics experiments, which are represented in: the study of a simple pendulum, the study of rectilinear motion, and the study of the charging and discharging of a capacitor. The data acquired will display into two interfaces, the first on the smartphone screen, while the second form is on a web page using WiFi depending on the web server.

The main objective of this platform is to verify the laws of physics and to visualize certain simple physical phenomena, which may require large, complex, and expensive equipment. It can be ordered to conduct the experiment once for a group of students, and immediate results are obtained, and therefore must be memorized and recorded manually for later study and analysis, development, and processing, which makes it difficult for students to understand, observe and scrutinize the phenomenon while observing and limiting the results. So the incentive is as follows

- Carrying out simple experiments with simpler equipment and electronic materials that

are available.

- Simplify the observation of physical phenomena.
- Obtaining and processing the results of an experiment and keep them for later use.

The platform contains two Fundamental experiments in the principles of physics and is the basis for other experiments ,the experiments are:

- A simple pendulum experiment using a potentiometer.
- Inclined plane experiment using an ultrasonic sensor.
- capacitor charging and discharging experiment (RC circuit)

2.7 Conclusion

In this chapter, we tried to demonstrate the crucial role of mechatornics in fundamental physics experiments. The main role of mechatronics lies in developing a new experiment setup with the integration of modern technology. As we have seen, these kind of setup don't only ensure efficient results but also can be implemented by a simple and accessible tools. In the second part of this chapter, we briefly presented the platform that we wish to implement, as well as the main objectives and the experiment to be implemented.

Chapter 3

Design of the fundamental physics platform

3.1 Introduction

The platform aim mainly to allow users to receive the data of a fundamental physics experiment on their own smartphone. The data transferring is granted mainly through Bluetooth connection and as an extended solution; through WiFi connection. In this chapter, we are going to present the different materials and software used in order to accomplish the goal

3.2 General overview of the system

The platform will base mainly on the generic WiFi+BT+BLE MCU module "ESP32". The first solution of the platform will employ Android devices (smartphone or tablet) in order to receive and send data to and from the ESP32 board through a Bluetooth connection. As for the second solution, the platform will employ a web page hosted on the ESP32 board to send and receive the measuring data. Figure (fig. 3.1) shows a general overview of the system. As depicted in (fig. 3.1), connecting to the board could be established by using two ways (BL or WiFi) all depending on the location of the user. Bluetooth connection presents

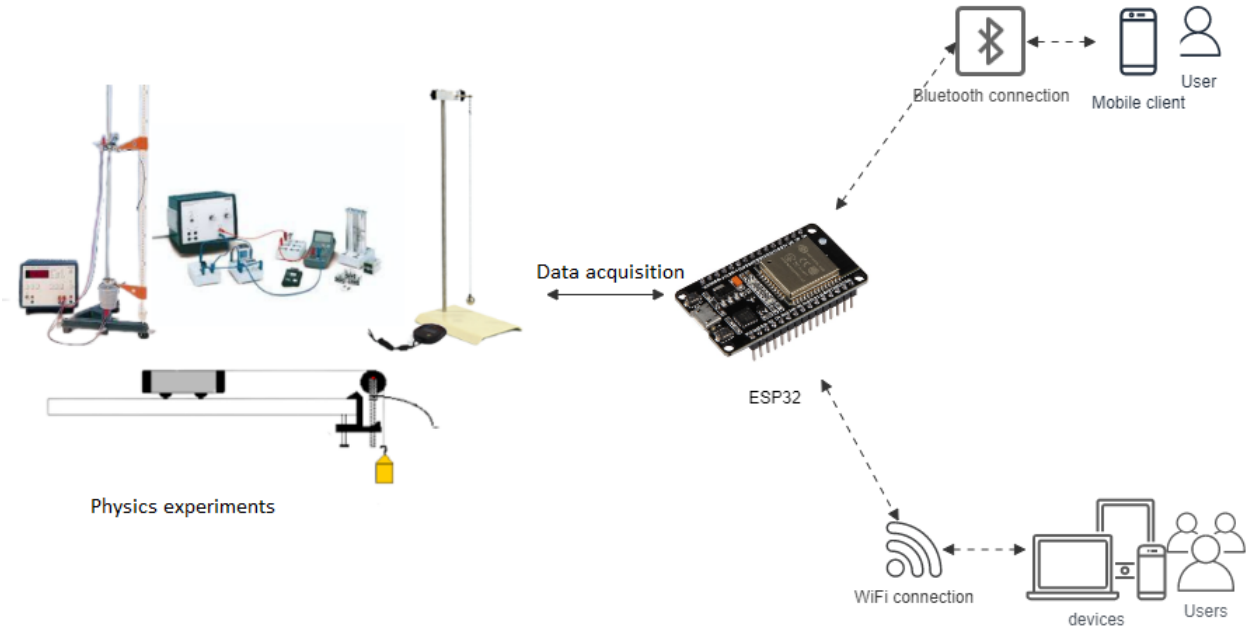


Figure 3.1: General overview of the system

the advantage of low energy consumption with respect to WiFi communication, therefore we had chosen to use the Bluetooth connection for data acquisition in case the users are in the same location of the experiments. However, using a web page offer the possibility to access to experiment data regardless of the location of the student.

The platform could be divided in two main parts; software and hardware. The software part will provide students with an interactive interface while the hardware part presents the physics experiments and the ESP32 board.

3.3 Hardware Description

The hardware of our platform will be the ESP32 board and the experiments components. As an initiative,To illustrate our achievement, we performed three fundamental physics experiments; the pendulum for the study of oscillations, the study of an accelerated movement and the study of an RC circuit highlighting the charging and discharging of capacitors:

3.3.1 ESP32 MCU module

The main component of our platform is the ESP32 micro-controller. The ESP32 MCU integrated the Classic Bluetooth *BT*, the Bluetooth Low Energy *BLE*, figure (fig. 3.2) depicts an image of the ESP32 CPU module. The module embeds two "ESP32-D0WDQ6"



Figure 3.2: The ESP32 MCU module

CPU cores chip[1] that can be controlled individually with an adjustable clock speed between 80 MHz to 240 MHz. The chip provides as well a co-processor, that can work instead of the main CPU in sleeping mode to save power while performing tasks that do not require much computing power. Besides, the module integrates a variety of peripherals such as touch sensors, Hall sensors, Ethernet, high-speed SPI, UART, I2S, and I2C. Table 3.1 presents the important specification of the module

Table 3.1: ESP32 WROOM 32 Specifications

ESP32 WROOM 32 Specifications	
CPU	Two low-power Xtensa 32-bit LX6 microprocessors
Internal memory	448KB of ROM, 520KB of on-chip SRAM, 8KB RTC FAST Memory, 8KB RTC SLOW Memory, 1Kbit of eFuse.
External flash memory	Up to 11 MB + 248 KB can be mapped at a time.
External SRAM	Up to 4 MB can be mapped at a time.
WiFi	2412 2484 MHz as center frequency range of operating channel.
Bluetooth	Bluetooth v4.2 BR/EDR and Bluetooth LE.

[1]released by the fabless semiconductor company "Espressif Systems"

3.3.2 Micro-SD card and Micro-SD card module

In order to overcome the problem related to RAM memory saturation, we decided to use an 8GB Micro-SD card. Interfacing the Micro-SD card with the ESP32 requires a Micro-SD card module (fig.3.3), which employs the SPI[2] communication protocol.

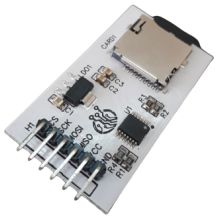


Figure 3.3: Micro-SD card and Micro-SD card module

3.3.3 Arduino bluno

As a member of the Bluno Family, Bluno (fig. 3.4) is the first of its kind in integrating Bluetooth 4.0 (BLE) module into "Arduino Uno", making it an ideal prototyping platform for both software and hardware developers to go BLE.



Figure 3.4: Bluno Board

Bluno integrates a TI CC2540 BT 4.0 chip with the "Arduino UNO" development board.

[2]Serial Peripheral interface

It allows wireless programming via BLE, supports Bluetooth HID, AT command to config BLE and we can upgrade BLE firmware easily. Bluno is also compatible with all "Arduino Uno" pins which means any project made with Uno can directly go wireless!

3.3.4 Ultrasonic sensor for proximity measuring

Ultrasonic sensors based mainly on the transmission and the reception of Ultrasonic Waves (USW) energy. These kind of sensors are mainly used as proximity detectors and velocity meters. These USW are mechanical acoustic waves covering frequency range well beyond the capabilities of human ears (over to 20kHz).

When the USWs propagate, it incidents an object, a part of the energy is absorbed while the rest is reflected. The waves are reflected in diffusion manner. That is, regardless of the direction where the waves come from, they are reflected almost uniformly within a wide solid angle, which may approach 180° (fig. 3.5).

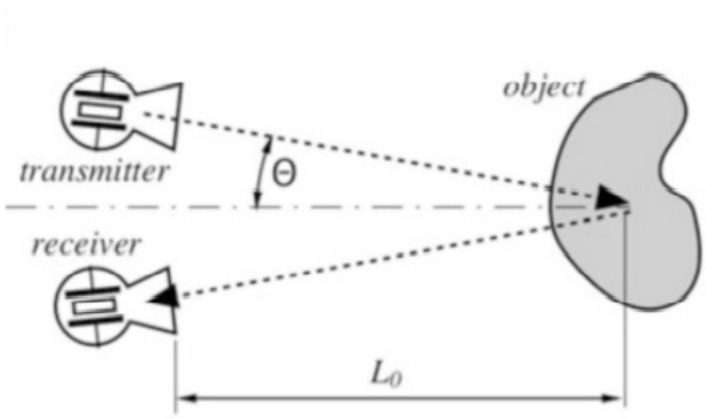


Figure 3.5: Ultrasonic proximity measurement[12]

If the object moves, frequency of the reflected wavelength will differ from the transmitted waves in which is called the Doppler effect [12]. The distance L_0 could be calculated through the speed of the ultrasonic waves$v$, and the angle θ.

$$L_0 = \frac{v \times t \times cos(\theta)}{2}$$

where t is the time for the ultrasonic waves to travel to the object and back to the receiver. Besides, if the transmitter and the receiver are positioned close to each other as compared with the distance to the object, then $\theta = 0$, and thus $cos(\theta) = 1$.

To generate mechanical waves, movement of the surface is required. This movement creates compression and rarefication of medium that can be gas, liquids, or solids. There are two types of the ultrasonic transducers for operation in air: piezoelectric and electrostatic. Although, the electrostatic provides high sensitivity and bandwidth, they required a high voltage for polarization and operation. Therefore, piezoelectrics are more common, they convert electrical energy into mechanical energy and require less low voltage excitation signals (10-20 Vrms), making the excitation steps simpler and less expensive [12]. For our experiment, we are using the HC-SR04 sensor (fig. 3.6). The sensor operates when the Trig



Figure 3.6: The HC-SR04 sensor

is excited by a pulse of 5V for at least 10µs (see fig. 3.7).This will initiate the sensor to transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic wave reflection from receiver, it will set the Echo pin to high 5V and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (T_{on}) of Echo pin [13].

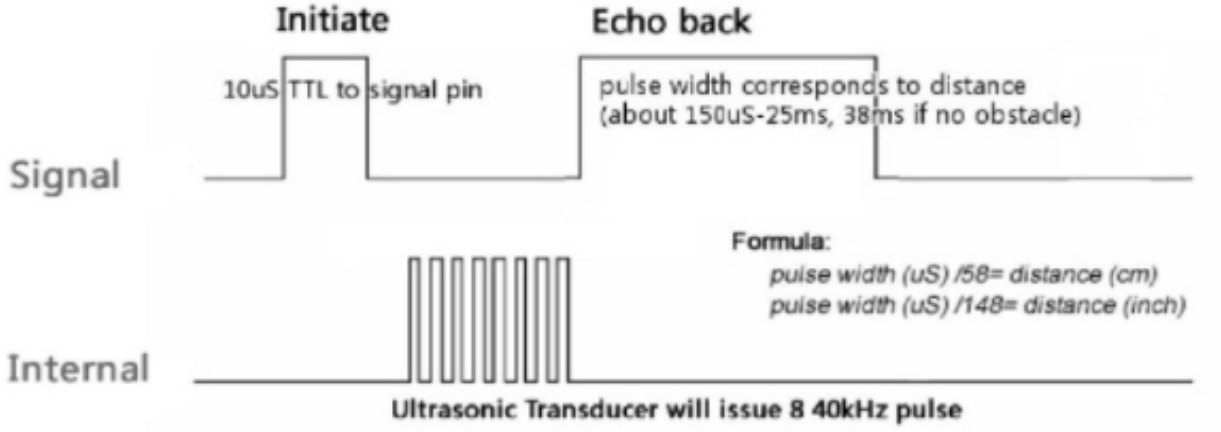


Figure 3.7: The timing diagram of HC-SR04 sensor [13]

3.3.5 Liquid Crystal Display "LCD"

The LCD display (16×2) allows us to display a text of 16 characters for the two rows. The LCD (fig. 3.8)is used in order to display the different states of the measuring process and make the experiment setup as integrated as possible.

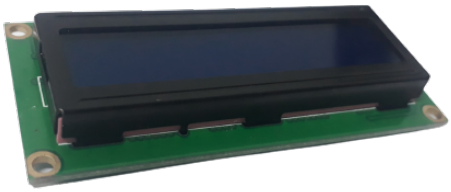


Figure 3.8: LCD display

3.4 Software description

Our platform includes two wireless solutions; for a local experience in which the connectivity is ensured via Bluetooth using an Android application, and for remote experience in which the connectivity is ensured via WiFi through a web page. In this section, we present the different technologies used to develop both the solutions (we had gone a little bit in-depth

with regard to web technologies since it's not in our formation)

3.4.1 MIT App Inventor platform

The mobile phone application will be implemented using "MIT App Inventor" platform (fig. 3.9). "MIT App Inventor" allows us to create applications for smartphone that are compatible with all the operating system.Through a visual programming environment (by using Blocks) MIT App Inventor allows implementing phone applications with a variety of options.



Figure 3.9: MIT App Inventor.

We used MIT App Inventor mainly to build an application that connects with the ESP32 via Bluetooth and thus receives the measuring data.

3.4.2 Bluetooth protocol

The communication between the ESP32 board and the smartphone application is established through a Bluetooth connection. What distinguishes Bluetooth is its special attention to short-distance communication, usually less than 30 ft. Both hardware and software are affected by this special attention [14].

3.4.3 Python Integrated Development Environment (IDL)

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for

use as a scripting or glue language to connect existing components together.

The Python to IDL bridge provides a mechanism for calling IDL code from Python. The IDL module contains a set of methods that allow you to initialize and interact with the IDL interpreter. In addition, each object passed back from IDL is wrapped in an instance of the IDL class; you can then call methods on the underlying IDL object, or set and retrieve attributes on the object.

3.4.4 HTML

HTML stands for "HyperText Markup Language", it marks-up text with instructions to be displayed through a web page. Programming languages, on the other hand, use logic control statements to direct the flow of program execution. Programs take input and produce output, while HTML and related markup languages format existing content [15]. Therefore, HTML is responsible for defining the basic structure of a web page by formatting and organizing the content of a website.

"Hypertext" in HTML refers to text that includes "hyperlinks" or more often called "links". These links are instructions that tell a web browser to move to a new page. "Markup" is text with HTML tags applied to it, such as "bold" or "italicize" [15].

HTML files are simply text files named with a specific extension; usually, .html, though there are other extensions, .html is the most used. When a file is saved with this extension and then loaded into a web browser, the page will display as formatted according to the instructions in the markup. In the extracted code (Listing 3.1) from *index.html* file, the markup elements are those written within brackets"< >" such as "body". The entire HTML code lives within the tags: **<html>** and **</html>**. The HTML page structure is divided mainly into two main divisions: firstly, the head **<head>...</head>** which contains the title of the page, metadata, libraries, icons...etc. The second division is the body of the page which lives within the two tags **<body>...</body>**, the body contains all the elements that are visible on the page.

Listing 3.1: Extracted THML code

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Pendulum experiment </title>
    <link rel="icon" type="image/png" href="pendulumIcon.png">
    <link href="style.css" type = "text/css" rel = "stylesheet">
</head>
<body>
     <h1 id = "title" class="title" > Pendulum experiment WEBPAGE </h1>
    <script src = "app.js"> </script>
</body>
</html>
```

In our page web, we attempt to use the HTML5, the latest version of HTML language. Alike, the previous version of HTML, in which the HTML, CSS, and JavaScript tends to overlap, HTML 4 and 5 imposes the the standardization and demarcation of different roles within web development. Therefore, the web page building codes are going to be in different files (.html files, .css files, and .js files). In order to ensure that, the web page code will not be written directly in the Arduino code, but rather in separate files stored in the micro SD card.

3.4.5 CSS

CSS Stands for "Cascading Style Sheets", it's a language that handles the look and feel of the internet are guided by an additional set of instructions handed to the browser. It's a language different from HTML but dependent. CSS provides the ability to change the appearance and the shape of HTML structure in aim of providing a better visual experience. CSS allows

us to set general rules for our various styles and to set exceptions to those rules when needed [15].

While the HTML provides the building blocks for the web page, CSS allows to stylize the way these blocks look, work, and interact with the blocks around them. The term ”Cascading” in CSS means that the defined styling rules will cascade throughout the site. the code (listing. 3.2) shows an example of CSS code.

Listing 3.2: Extracted CSS code

```
html{
    background: #7fc7bd3f;
    color: #025259;
    padding: 10px;
}
.title{
    color: #2A3E59;
    padding: 10px;
    text-align: center;
}
```

The code above (Listing 3.2) illustrates the use the CSS to style the web page by setting the background color to ”#7fc7bd3f”, and the text color to ”#025259”, with a padding of ”10px”. while the class ”title” will set the a centered text with a color to ”#2A3E59” and ”10px” padding. This style rule will be applied to all elements belonging to the ”title” class. In CSS the attribute such as background, color, padding ...etc are called *properties*, while the corresponding values are *values*. The sets of properties and values inside the braces ”” are called *declaration*. ”html” and ”title” are called *selectors*.

The code (Listing 3.2) define the style rules in separate code from the HTML code (Listing 3.1), this kind of CSS code is called *External CSS*, it allows the styles to be added to multiple pages at once. External CSS uses CSS code in a separate, unique file called a style sheet

that is used as a reference by all the HTML files. The CSS file is called, within the **<head>** tags of the HTML file as shown in (listings. 3.3) .

Listing 3.3: Calling an external CSS code from HTML page

```
<link href="style.css" type = "text/css" rel = "stylesheet">
```

Exist there other styles of CSS uses which are *Inline CSS*, in this case the CSS styles are added inside individual HTML elements, and *Internal CSS*, in this case, the CSS is added to a selection of HTML elements within a single individual HTML page. Usually this is done by placing a set of **<style></style>** tags in the **<head></head>** elements. This last, is useful when there is just one web page. In our case, we attempt to use *External CSS* since there will be couple of pages to stylize.

3.4.6 JavaScript

JavaScript is a high-level, dynamic, interpreted programming language that is well-suited to object-oriented and functional programming styles used principally for web development[16]. JavaScript allows the web user to interact directly without doing a page reload for every action[17].

In short term, JavaScript allows to create highly responsive interfaces that improve the user experience and provide dynamic functionality, without waiting for the server to load up a new page [18]. As same as CSS, Js scripts could be embedded into the HTML page similarly as *Internal CSS* styles, in this case it's called *Embedded scripts* and it need only to be added into the **<body></body>** element using the two tags **<script></script>**. Additionally, Js script could be written in separate file in which called *External scripts* and HTML page point to the Js file by using "scr" attribute as shown in the codes (listings. 3.4)

Listing 3.4: Calling external JavaScript code for HTML page

```
<script src = "app.js"> </script>
```

The script could be placed anywhere in the HTML page, but the most common places for scripts are in the **head** of the document and at the very end of the **body**. However, the the end of the document, just before the **</body>** tag, is the preferred placement because the browser will be done parsing the document and its DOM[3] structure [18]. Therefore, the information will be ready and available by the time it gets to the scripts and they can execute faster.

In our case, Js will be used mainly to make XML HTTP requests to the server which is the ESP32 board. These requests will allow us to get the data stored on the server and exploit it later.

3.4.7 HTTP protocol

HTTP stands for "Hypertext Transfer Protocol", it is the protocol programs use to communicate over the World Wide Web [19]. The web servers communicate through HTTP protocol (fig. 3.10), thus they are often called HTTP servers. These servers store the internet data and provide it to the HTTP client once it is requested.

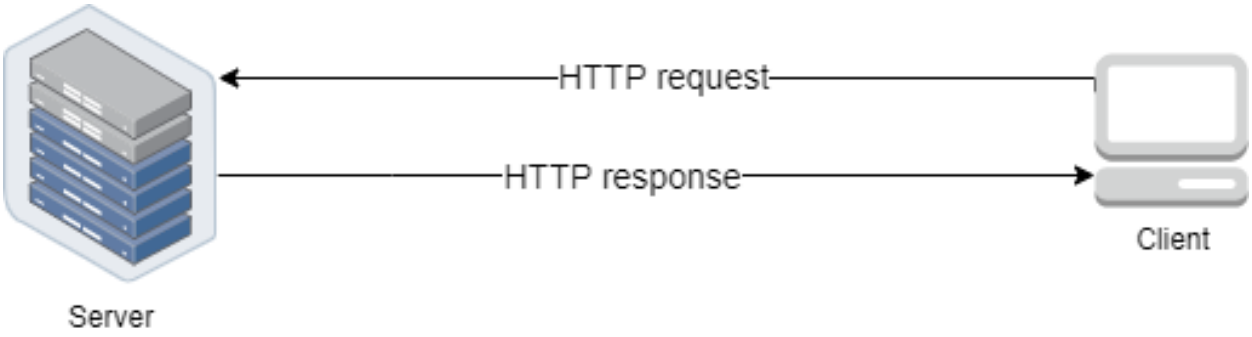


Figure 3.10: Server-Client communication

HTTP messages are divided into two types; *request message* and *response message*. Request messages request an action from a web server. Response messages carry results of a request back to a client. Both request and response messages have the same basic message structure. The below formats (listings. 3.5, 3.6) shows the two formats for request and response

[3]Document Object Model

message .

Listing 3.5: The format for request message [19]

```
<method> <request-URL> <version>
<headers>
<entity-body>
```

Listing 3.6: The format for response message [19]

```
<version> <status> <reason-phrase>
<headers>
<entity-body>
```

The action that the client wants the server to perform on the resource. It is a single word, like “GET,” “HEAD,” or “POST” ...etc. The ”GET” operation allows the client to retrieve an object from the server. the ”GET” operation is used in our JS code to get the measurement data via sending a request **xhr**, an object form **XMLHttpRequest** class as the code below shows (listing. 3.7).

Listing 3.7: Retrieve data from the ESP32

```
var xhr = new XMLHttpRequest()
xhr.open("GET", "/potentiometer", true);
xhr.send();
```

the argument ”/potentiometer” is a Uniform Resource Identifier (URL) which is the name of one of the resources of the server. Using the resource name will allow us as clients to point out the desired resource. The third argument which is a Boolean concern whether we want our requests to be asynchronous, and in our case, it’s ”true”. From the side of the server, it should make sure that there are no issues in the request (listing. 3.8), and then it will provide the corresponding response (here it is *chartData.c_str()*).

Listing 3.8: Server side code to treat the requests related "/potentiometer" resource

```
server.on("/potentiometer", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", chartData.c_str());
    });
```

Each request should have a response whether it's successfully treated, denied, or even failed (listing 3.9).

Listing 3.9: Response of XMLHttp request

```
XMLHttpRequest{readyState: 4, timeout: 0, withCredentials: false, upload:
   XMLHttpRequestUpload, onreadystatechange: f…,}
```

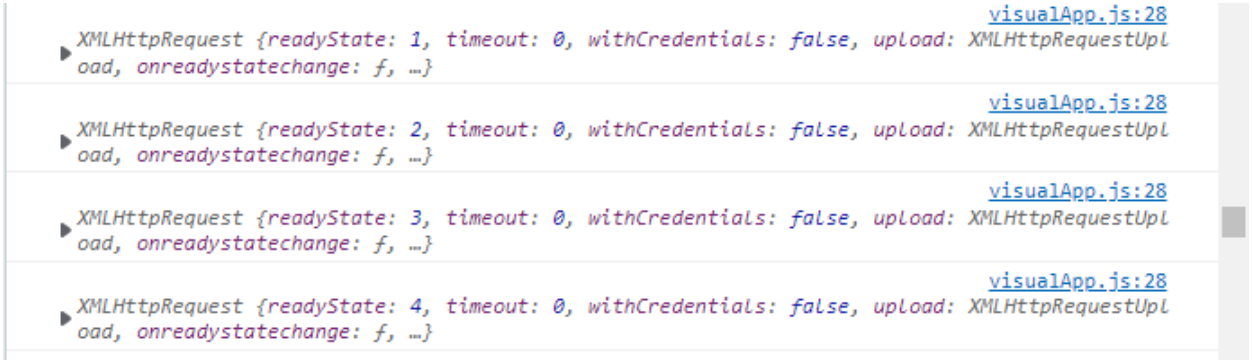


Figure 3.11: XML HTTP request response

As the shown in figure (fig. 3.11), each HTTP request is passing through four states **readyState** to indicate the state of the request, **ReadyState** method could return 5 different values in which:

- "readyState = 0" indicate that the request has not been initialized.
- "readyState = 1" indicate that the request is loading the **send()** method has been invoked, request in progress.
- "readyState = 2" indicate that the request has been Loaded, and the **send()** method

has completed, entire response received

- ”readyState = 3” indicate that the request is being parsed
- ”readyState = 4” indicate that the request has been completely parsed, and it is ready for harvesting.



Figure 3.12: XML HTTP Request methods

Another important method is the **status** method (fig. 3.12), this method tells the client what happened, as instance *status : 200* indicate that the request was successful. Stutus code classes and their signification are presented through the table 3.2.

Table 3.2: Status code class [19]

Overall range	Defined range	Category
100-199	100-101	Informational
200-299	200-206	Successful
300-399	300-399	Redirection
400-499	400-415	Client error
500-599	500-505	Server error

3.5 Conclusion

The First part of this chapter was dedicated to presenting the essential hardware used in our implantation, we had chosen ESP32 as a development board due to its integrated Bluetooth and WiFi connectivity, thus we won't need an extension module.

The second part of this chapter was for the software description, we have presented the MIT App Inventor platform used for the application development. Bluetooth is briefly described because we had not to go through a lot of programming to control it, MIT App Inventor takes charge of its specification.

However, the part relative to the web page development was presented with some details since it was a new subject for us. Hence, some research was conducted to allow us to exploit such technology.

Chapter 4

Tests and results

4.1 Introduction

In this chapter we are going to presents firstly the different steps in the process of implementing the platform, and then the results of the tests. In this chapter we will present the realization of three experiments highlighting physical phenomena as well as the results obtained. These are the study of a simple pendulum, the study of rectilinear motion and the study of the charging and discharging of a capacitor.

4.2 Pendulum experiment for the study of oscillations

The pendulum experiment based on using a mass attached via a solid shaft to a potentiometer as showing in the figure (fig. 4.1). The upper end of the shaft is attached to the potentiometer in such a way when the pendulum is dragged, the potentiometer control shaft is turned at the same angle as the swinging mass.

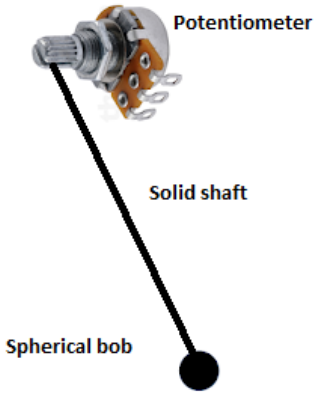


Figure 4.1: The basic apparatus of the pendulum experiment setup.

Obviously, this is a real pendulum in which the damping forces are evident and must be considered during data analysis. Moreover, if it's for damping study, friction is not a problem, however, it could be minimized relative to change in amplitude per swing by making the mass more massive, or the shaft longer [20]. We used an LCD display provides the user with an interface to display the state of the micro-controller (measurement, measurement done, sending data, sending done,..etc.) and thus keep the user informed of the progress of the experience. Figure (fig. 4.2) depicts the implemented setup.



Figure 4.2: The implemented pendulum setup

To illustrate the wiring circuit of the potentiometer and the LCD, a wiring circuit is depicted in the figure (Fig. 4.3) by using an open-source to develop CAD software for the design of electronics hardware *Fritzing*. The potentiometer wiring circuit is shown in the figure.

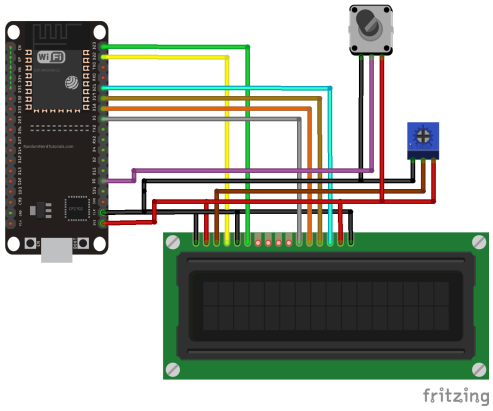


Figure 4.3: Potentiometer wiring circuit

4.2.1 Local data acquisition

Local data acquisition, (fig. 4.4), is ensured through the Bluetooth communication between the ESP32 and the smartphone application. In this case, the micro-controller starts the data acquisition by triggering a dedicated function ”pendulumMeasure()” as soon as the pendulum is dragged from its rest position.

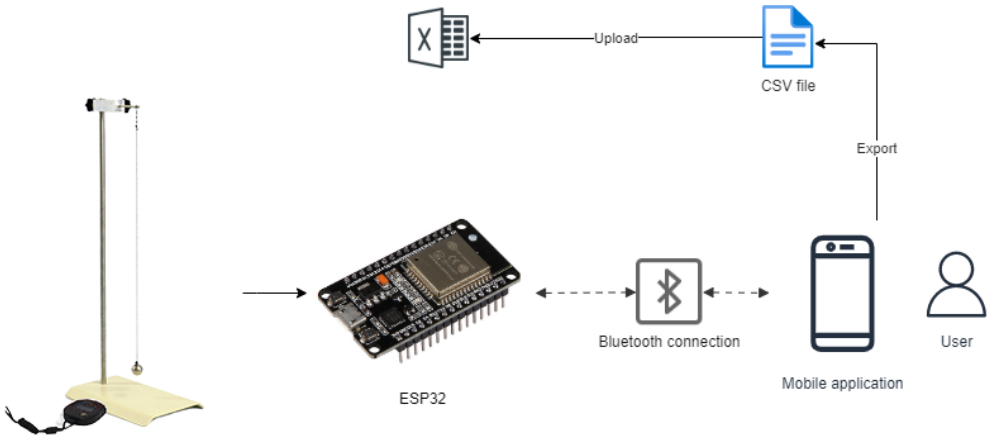


Figure 4.4: Overview of the system with the Bluetooth solution

The micro controller measures the voltage of the potentiometer, as well as the corresponding measurement instance in milliseconds using the "millis()" function in the Arduino code. In this case the data is stored throughout the measurement process in predefined arrays as a global variable in the Arduino code to be sent later as showing in the code (listing 4.1).

Listing 4.1: Measuring the voltage of the potentiometer

```
int Pot_Ana_Val = analogRead(potentiometerPin); // read the analog value the
    potentiometer (of pin 4)
// Convert the analog value to digital value
float Voltage = Pot_Ana_Val * (3.3 / 4095.0);
timeArr[i] = currentMillis; //times array;
startMillis = currentMillis;
valueArr[i] = Voltage; // Value array
```

Data transferring via Bluetooth is a point-to-point connection, thus, just a single user can connect to the board at once. This kind of connection could be managed in the ESP32 by defining an object of "BluetoothSerial" class. This class of the library "BluetoothSerial.h" offers a set of methods to transfer data from and to the ESP32 efficiently. The data is sent, to the smartphone application, in such format that allows us to be able to export this data as a *.CSV* file as showing in (listing 4.2).

Listing 4.2: Sending data via bluetooth in format CSV

```
ESP_BT.print("Time");
ESP_BT.print(";");
ESP_BT.print("Voltage");
ESP_BT.print("\n");
for(int i = 0; i<400 ; i++){
  ESP_BT.print(timeArr[i]);
```

```
ESP_BT.print(";");
ESP_BT.print(valueArr[i]);
ESP_BT.print("\n");
```

On the other hand, the phone application would receive the data and allows the user to download it in *.CSV* format, and thus it will be easier to be used as a data file. We were able to ensure such functionality by using the following blocks (figs. 4.5, 4.6). Although, we tried several ways to achieve the goal to save the received data as *.CSV* file, the only way that we were able to come up with was to define two lists but work with them as strings. This method allows us the achieve the goal and get a *.CSV* in its proper format that could be exported and then uploaded directly to excel to draw the chart.

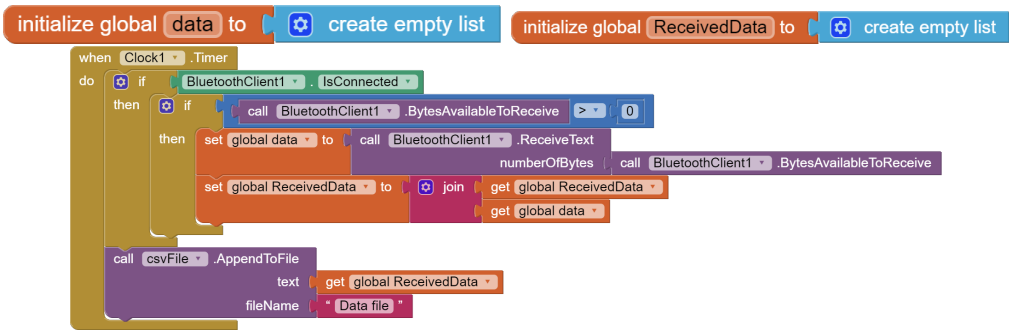


Figure 4.5: Receive the data from the ESP32 and create a .CSV file

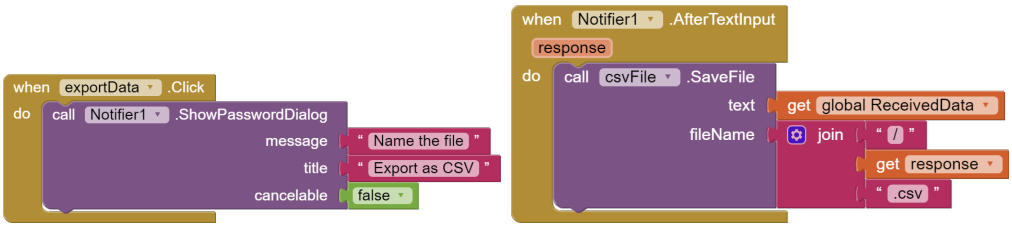


Figure 4.6: The blocs responsible to store the received data as a CSV file

4.2.2 Test and results of local data acquisition

The application allows the user to connect to the ESP32 board to fetch the already measured data.

First of all, the user should be able to connect to the board efficiently, to do that, the user should paired his devices to the board firstly. Activating the Bluetooth connection from the application will display all the paired devices (fig. 4.7).

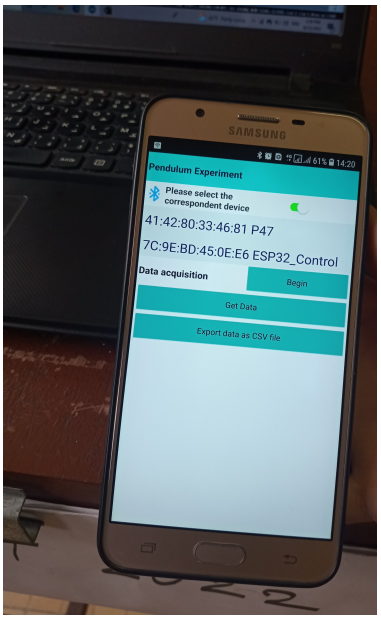


Figure 4.7: Connect to the ESP32 from a smartphone

The user can then select the ESP32 board and the connection is established automatically.

In the beginning, we planned to implement an application in which a real-time oscillation graph could be displayed on the user screen, however, this was not possible since the MIT App Inventor platform doesn't offer such an option. As an ulterior solution, we allowed the user to retrieve the data by pressing the "Get Data" button and the data will be transferred. At this point, the data are not significant since they are just voltages with the corresponding measuring instances. Although MIT App Inventor could display graphs and tables, they are all based on small amount of data, this is why displaying our data as a graph, or at least as a table was not possible as well.

Finally, we was able to figure out in a way to export the data as "CSV" file, (fig .4.8). The file could be uploaded later to the "Excel" to view the oscillations and exploit it. In order to open a CSV file in "Excel", the user may click on "**From Text/CSV**" from **Data** in

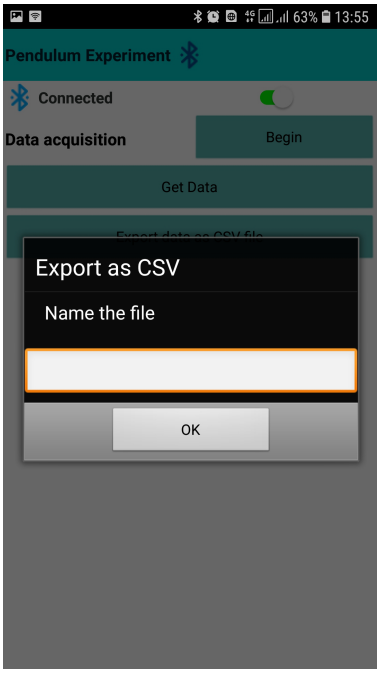


Figure 4.8: A screenshot of exporting data as CSV file

the tool bar to select the corresponding file (fig. 4.9) (in our case the path of the file was "This PCJ7 Prime.ai_aichasalag.Physics_Lab"). Excel will upload the file and open it in an "Excel sheet" (figs. 4.10, 4.11). At this point, an appropriate graph could be displayed by selecting the two columns then from **"Insert"** in the tool bar and choose **Scatter** from **charts** option (for our test, we chose **Scatter with Smooth Lines**). The voltage of the potentiometer changes analogously to the oscillation angle of the mass. The obtained result is shown in figure (fig. 4.12).

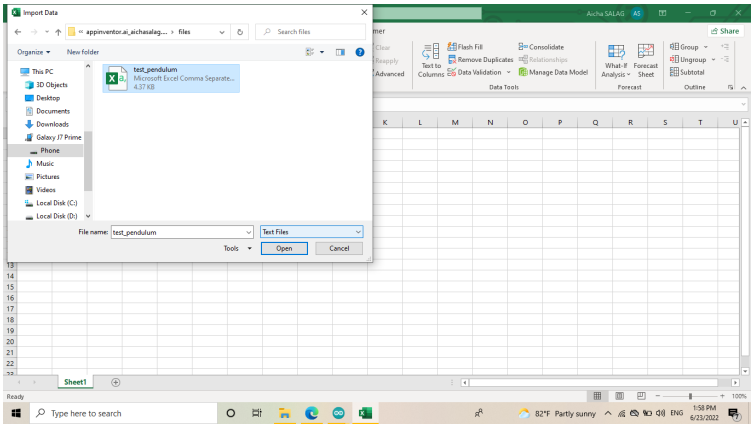


Figure 4.9: Selection of the CSV file

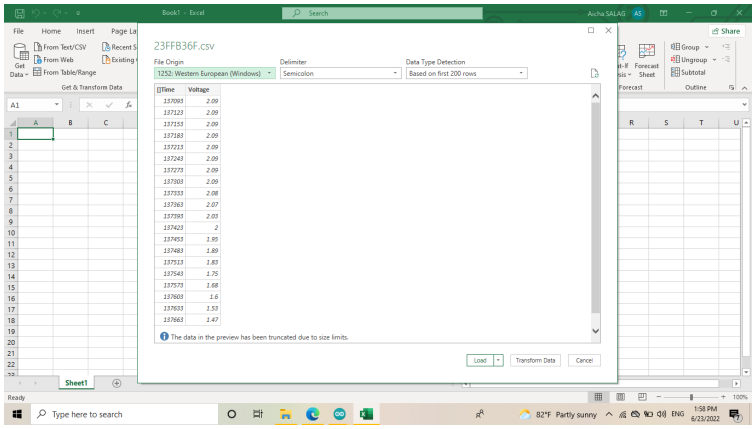


Figure 4.10: Upload the CSV file

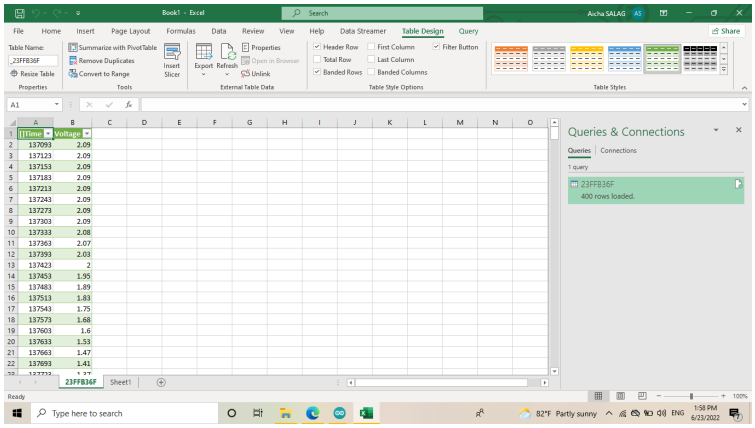


Figure 4.11: Open the data in Excel sheet

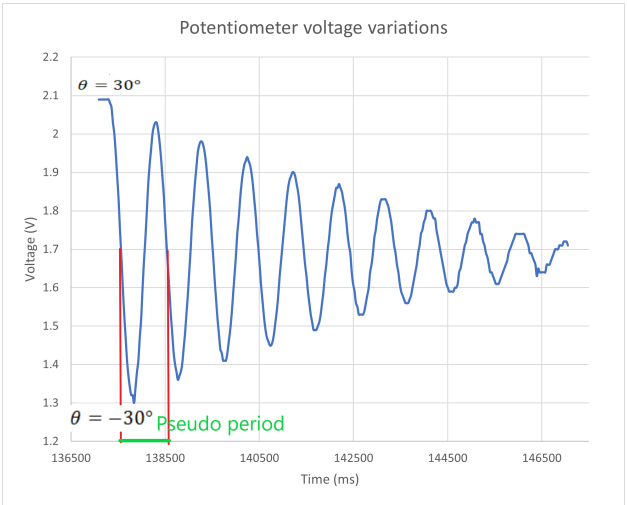


Figure 4.12: The obtained results from the test

4.2.3 Remote data acquisition

In order to make the experiment results available remotely, we have extended our solution to implementing a web page. Measurement results could be transferred via WiFi connection from the ESP32 to a web page. In this case, the ESP32 will be our server on which the web page is hosted and the users are its clients. Unlike the Bluetooth connection, in this case, the communication is a server-client communication in which several clients could have access to the result simultaneously. Figure (fig. 4.13). The first step to achieve the goal is to create a

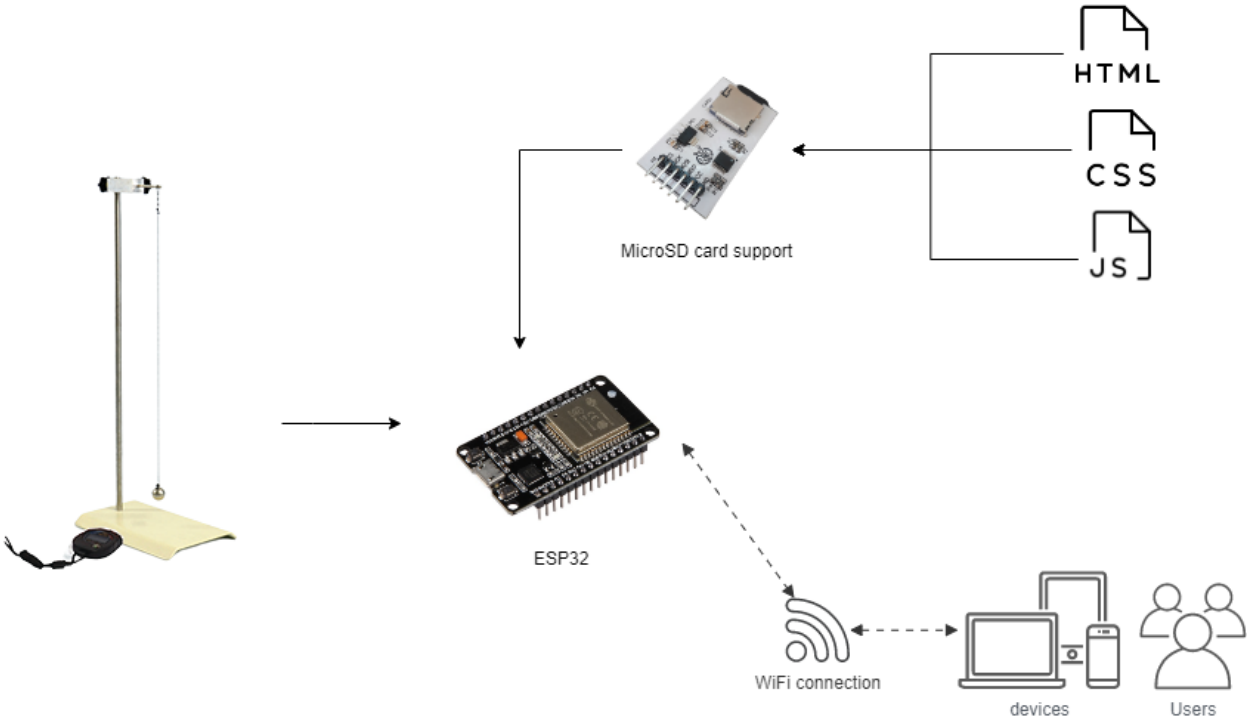


Figure 4.13: General overview of the system for remote data acquisition

web page that could send requests and receive responses form the server which is our ESP32 board. The web page is implemented using HTML5 for the structure, CSS for the style, and JavaScript to make it dynamic and to allow us make asynchronous HTTP requests. All the files of the web page are stored in an micro-SD card. Storing the web page codes outside the ESP32 board aims to avoid all kinds of problems such as RAM saturation, flash memory full...etc. Therefore, using a micro-SD card allows us to create several pages instead of a single page without warring too much about memory. In order to connect the micro-SD

card to the ESP32, we need a micro-SD card module. The wiring circuit of the micro-SD card and the potentiometer is shown in the figure (fig. 4.14). Our web page consists of three

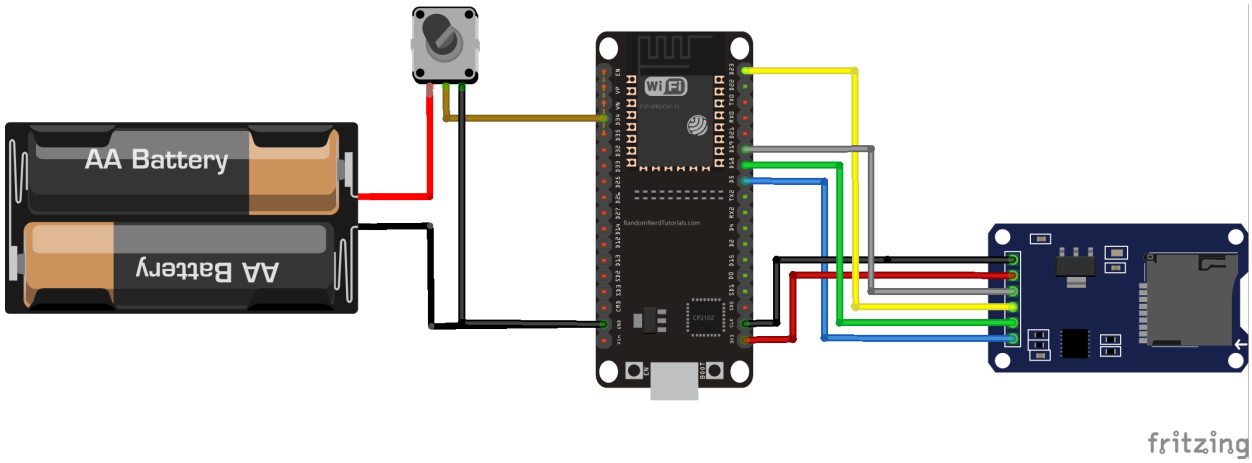


Figure 4.14: Wiring circuit of the ESP32 and the micro-SD card

HTML files; *index.html*, *pendulumExperiment.html*, and *trolleyExperiment.html*, one CSS file: *style.css*, three JS files; *app.js*, *pendulumExperimentApp.js*, and *trolleyExperiment.html* and icons. From the principle page *index.html* the user can command the start of measuring the voltage of the potentiometer by pressing a button, (listing. 4.3);

Listing 4.3: Extracted THML code from index.html page

```
<button id = "pendulumElement" class="button" placeholder = %STARTpendulum% >
    Pendulum experiment</button>
```

Pressing ”Start measuring” button will trigger an event to get the resource with the URL */start* of the web server (listing. 4.4);

Listing 4.4: Js code to send *Start* request

```
document.getElementById("pendulumElement").addEventListener('click', function(e){
  window.open("/start_pendulum");
});
```

The server in its turn will provide a response that opens a new web page */visualization.html*

and triggers the measuring function by assigning the placeholder *START* to the *state* variable, (listing. 4.8);

Listing 4.5: Arduino code to provide HTTP response

```
server.on("/start_pendulum", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(LED_BUILTIN, HIGH);
    state = "START";
    request->send(SD, "/pendulumExperiment.html", String(), false, processor);
    });
```

The *pendulumExperiment.html* page will send asynchronous HTTP requests *XMLHttpRequest* to get the resource associated to the URL */pendulumVoltage*, (listing. 4.6);

Listing 4.6: Js code to send an XMLHttpRequests

```
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/potentiometer", true);
    xhr.send();
```

Once the server gets the request, it will provide the client with a string value that contains the measured voltage and its correspondent instance of measuring, (listing.4.7, 4.8);

Listing 4.7: The Concatenation of time and measured value

```
    chartData = String(currentMillis)+ "," + String(Voltage);
```

Listing 4.8: Arduino code to provide HTTP response

```
    server.on("/pendulumVoltage", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send_P(200, "text/plain", chartData.c_str());
        });
```

For each change in the state of the request, we will call a function that will split the response

in order to get the X-axis value with is the time, and the Y-axis value which is the voltage. Then the function will add a new point (x, y) to the chart that we have already defined its characteristics. This operation will be repeated each 30 milliseconds due to the *setInterval* function as presented in (listing. A.1);

Listing 4.9: Treatement of the request

```
var i = 0;
refreshIntervalId =
setInterval(function ( ) {
  i+=1;
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
          console.log("i" + i);
      var chartInput = this.responseText;
          console.log(typeof chartInput);
          console.log(chartInput);
      var xyArray = chartInput.split(",");
          x = 0.001*parseFloat(xyArray[0]);
          y = parseFloat(xyArray[1]);
          console.log("length"+ chartT.series[0].data.length);
       if(chartT.series[0].data.length > 500) {
        chartT.series[0].addPoint([x, y], true, true, true);
      } else {
        chartT.series[0].addPoint([x, y], true, false, true);
       }
    }
    else{
      console.log("Error")
```

```
    }
    if(i == 400){
      clearInterval(refreshIntervalId);
  }
  };
  xhr.open("GET", "/pendulumVoltage", true);
  xhr.send();
}, 30) ;
```

To prevent the *setInterval* function to be executed forever, as long as the page is opened, we used the *clearInterval* method that allows us to stop the function execution at a specific condition, in this case, we chose after 400 successfully treated requests the function will stop running.

4.2.4 Test and results of remote data acquisition

The main goal of the web page is to display the voltage chart of a potentiometer attached to a pendulum, the same as for the "local data acquisition" solution. However, in this solution, the chart is displayed in real-time. We were able to do that due to *The Highcharts library* of JavaScript, which allows us to create charts in a proper format. The user can have access to the web page by typing the IP address provided by the ESP32. The user can connect using a lab-top, smartphone, or tablet. The figure below (fig. 4.15) illustrates the home page for the pendulum experiment, the user can press *Start measuring* once he is ready to get the measurements. Pressing this button will both trigger the measuring function of Arduino code and open a new page in which a real-time chart will be displayed (fig. 4.16), while figure (fig. **??**) shows the chart with the initial release angle. At this step, we need to make sure that the displayed chart is correspond actually to the real values. For that, we used *console.log()* function, and thus we will be aware of the quality of our chart. Figure (fig. 4.17) depicts a sample of the successfully treated data from the HTTP request.

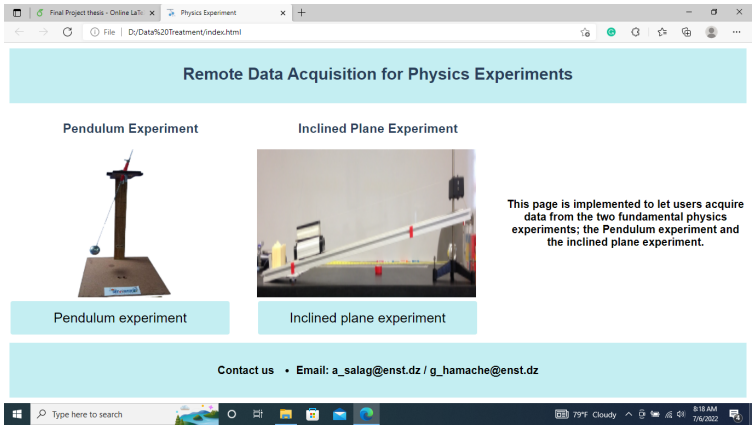


Figure 4.15: The user interface for the pendulum experience web page

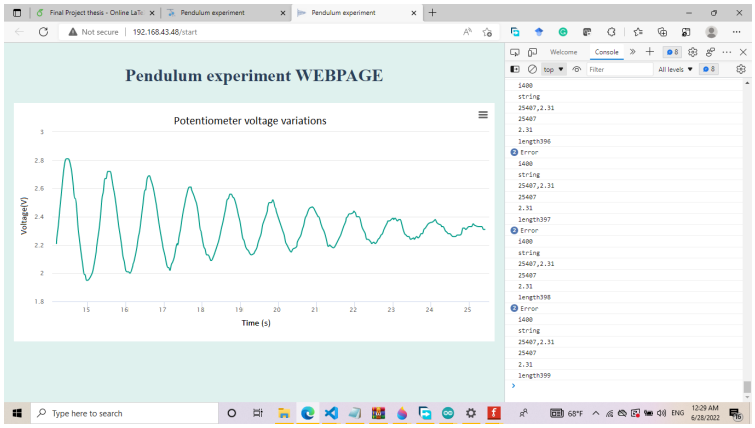


Figure 4.16: A screenshot of *pendulumExperiment.html* data display the oscillation of the pendulum (Voltage in (V) against time in (ms))

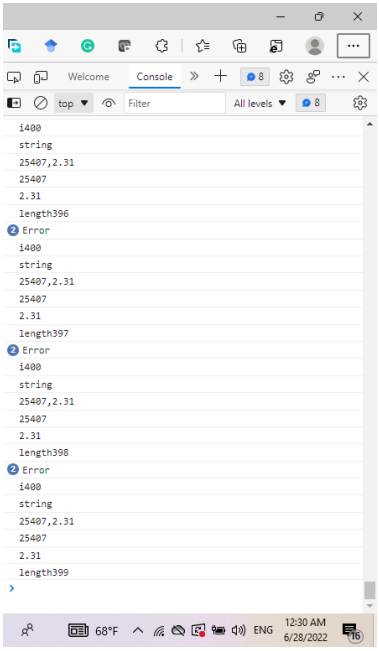


Figure 4.17: Investigating whether the results are consistent through the console

Passing the cursor over the chart will display the voltage measure and its corresponding instance of measuring. Furthermore, the chart could be displayed in full screen or downloaded using the *chart context menu* from the top right corner of the chart (fig. 4.18). Figure (fig. 4.19) shows a result of downloading the chart as *.png* image format.

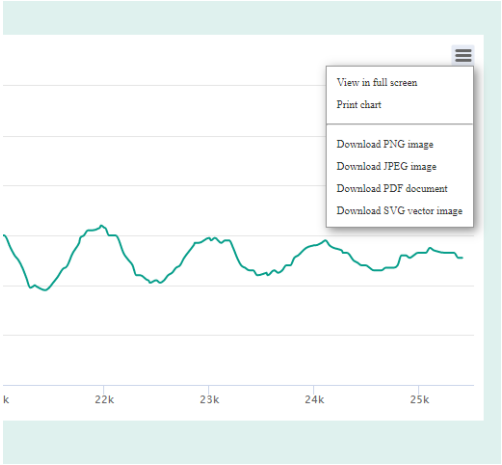


Figure 4.18: The chart context menu options

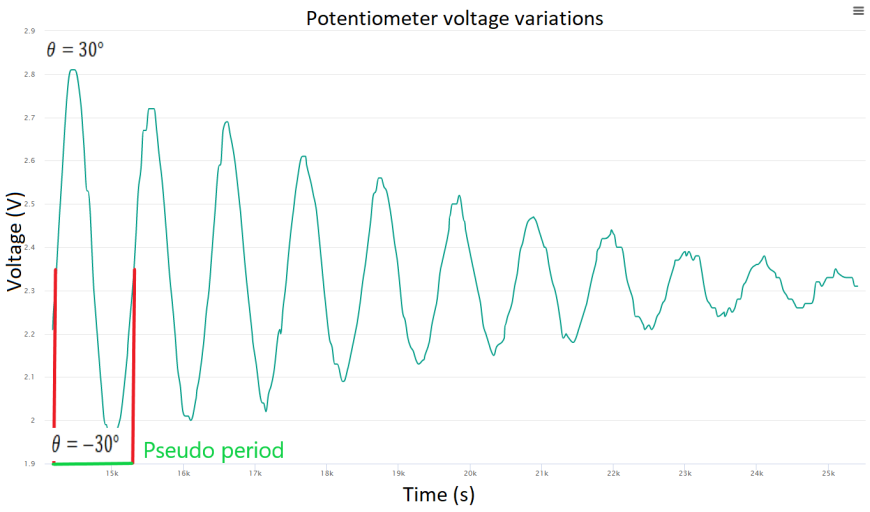


Figure 4.19: Potentiometer voltage variation chart

4.3 Inclined Plane for the study of an accelerated movement Experiment

The inclined plane experiment is one of the widely used fundamental physics experiments due to the variety of studies that can be done through it. the experiment serves to verify newton's first and second laws. It could be used to determine gravity **"g"** or coefficient of friction ***"f"***. The traditional setup for this experiment (fig. 4.20) mainly consists of an inclined plane, a cart, and photo-gates to obtain different cart positions and velocities, while in our setup we're using the ultrasonic sensor HC-sr04 to acquire the positions of the cart for about 5 seconds. The sensor is mounted on the plane exactly against the cart in such a way that allows it to measure the distance with the minimum error. Wiring the ultrasonic sensor

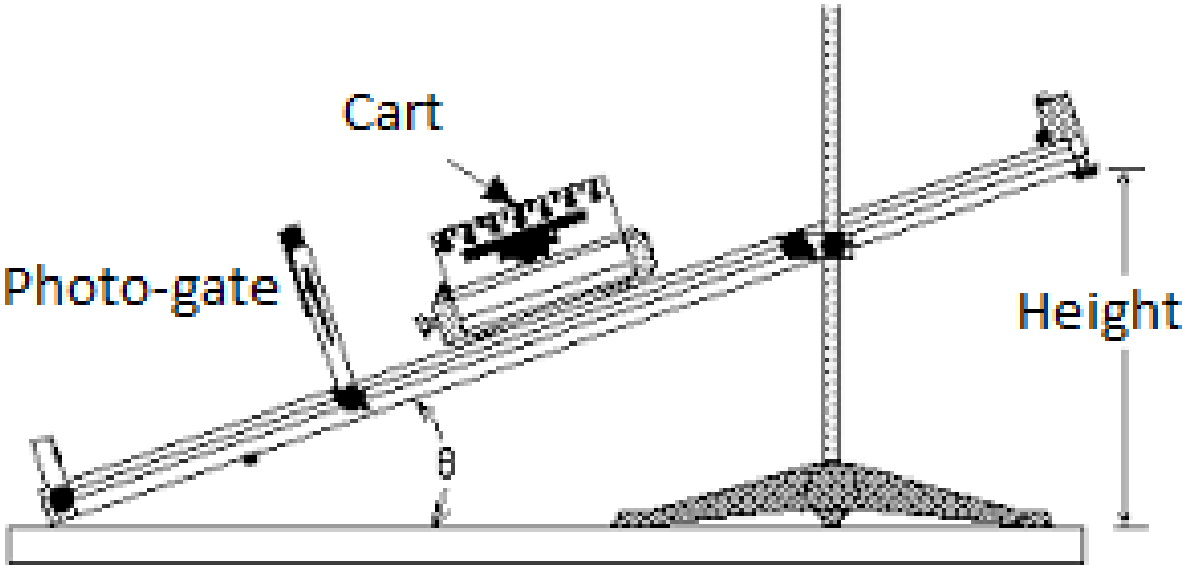


Figure 4.20: Traditional inclined plane setup

to the ESP32 board need four pins ($V_{cc}, Gnd, Trig, Echo$). The power supply V_{cc} need to be $5V$ which could not fulfilled by the ESP32, so we had used a $5V$ power supply. The trig pin is used to generate a signal with a minimum duration of 10 s of at least 2 V. This signal will trigger a burst of eight acoustic pulses at 40 kHz by the ESP32. In the meantime, an

electrical line is raised from 0 to 5 V and stays in this state until the microphone detects an echo of the signal.

Listing 4.10: Trigger the HC-sr04

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

`pulseIn(echoPin, HIGH);` is used later to get the sound wave travail time once the Echo pin is turned HIGH, this duration will allow use us to calculate the position of the cart `distanceCm = duration * SOUND_SPEED/2;`.
For this experiment two solutions are proposed; a local solution and a remote solution. Both the experiment based on the previous steps used to set the HC-sr04 sensor, however, the part corresponding to data acquisition differs

4.3.1 Test and results of local data acquisition

Similarly to the pendulum experiment, local data acquisition is done through Bluetooth connection (fig. 4.21). The user can select the experiment from the application home screen and use it to begin measuring distance (fig. 4.22). On the other hand, the ultrasonic sensor is connection to the ESP32 as showing in figure (fig. 4.23), while the entire setup is showing in (fig. 4.24).

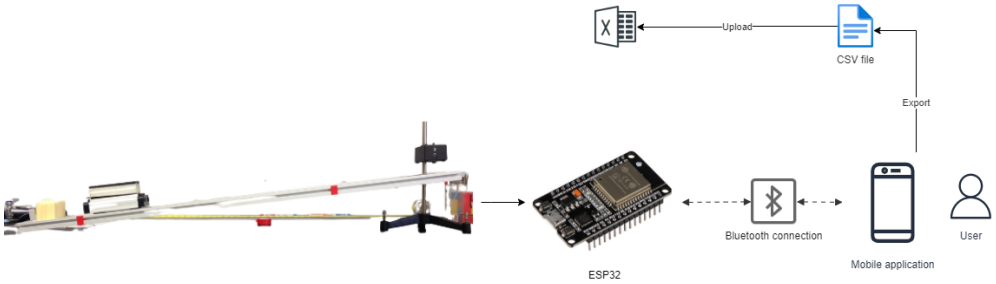


Figure 4.21: Overview of the inclined experiment setup for local data acquisition

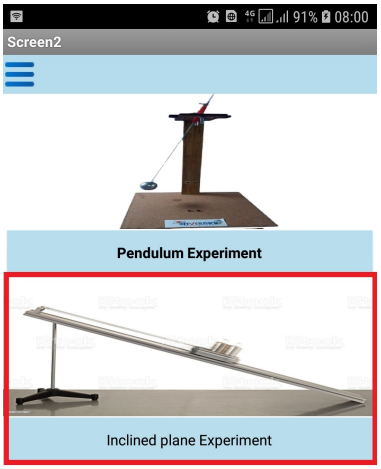


Figure 4.22: Home screen of the smartphone application

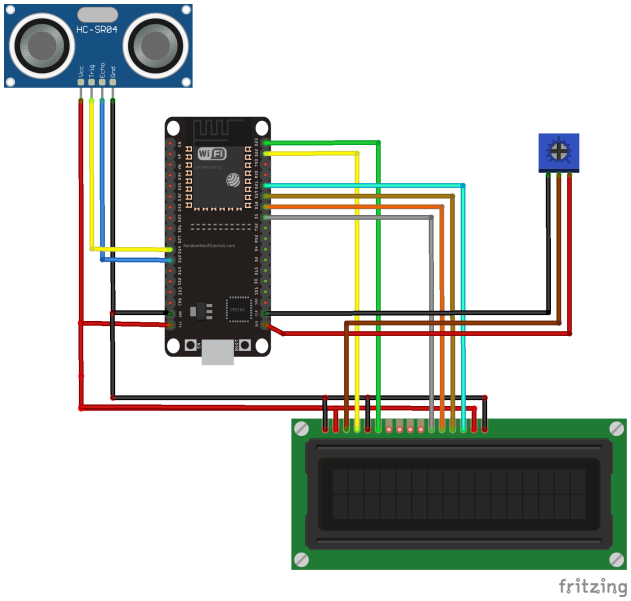


Figure 4.23: Wiring diagram of the ESP32, the HC-sr04, and the LCD



Figure 4.24: Image of the inclined plane connected to the ESP32 for local data acquisition

The measuring function is triggered when a starting command is received through Bluetooth. Hence, the user should press begin button before realizing the cart. In our program, we had chosen a measuring duration of 5 seconds which is quite sufficient for the user to trigger the measuring function and realize the cart which spends less than 2 seconds to arrive at the last point. Moreover, in order to get consistent data we set the time between two measurements to 30ms. This will all us to get more accurate data. As soon as, the measuring process is done, the user could retrieve the data by pressing the button *Get Data*. At this point, the user could export the data as *.csv* file under a chosen name in order to uploaded to EXCEL (fig. 4.25). The result of the data acquired from the test is shown in (fig. 4.26) and as it appears; it strongly corresponds to the equation of an accelerated movement *"position-time"*.

$$x = x_0 + v_0 t + \frac{1}{2} a t^2$$

Where:

- x_0 is the initial position of the cart (m).

- v_0 is the initial velocity of the cart (m/s).

- a is the acceleration of the cart (m/s^2).

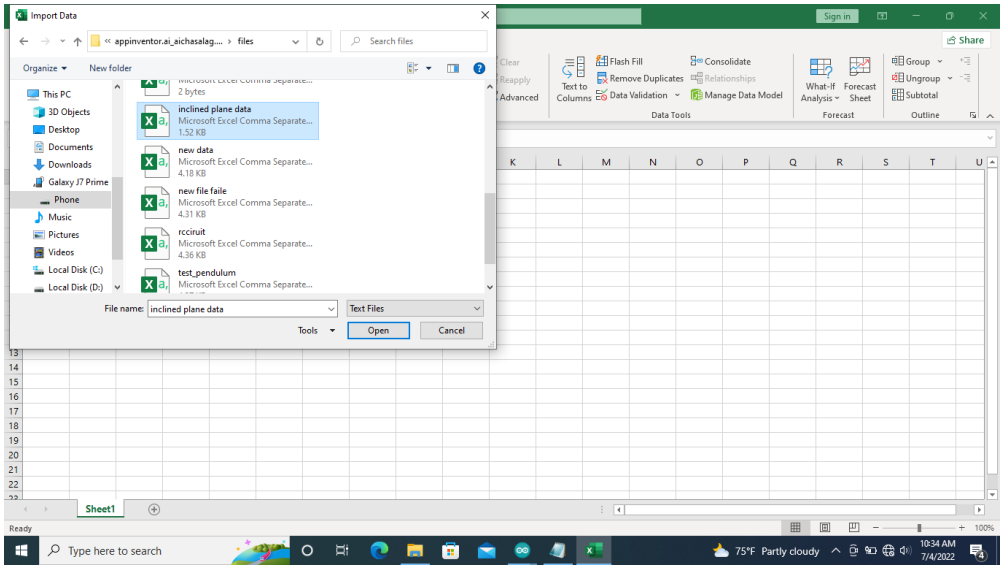


Figure 4.25: Uploading the CSV file to EXCEL

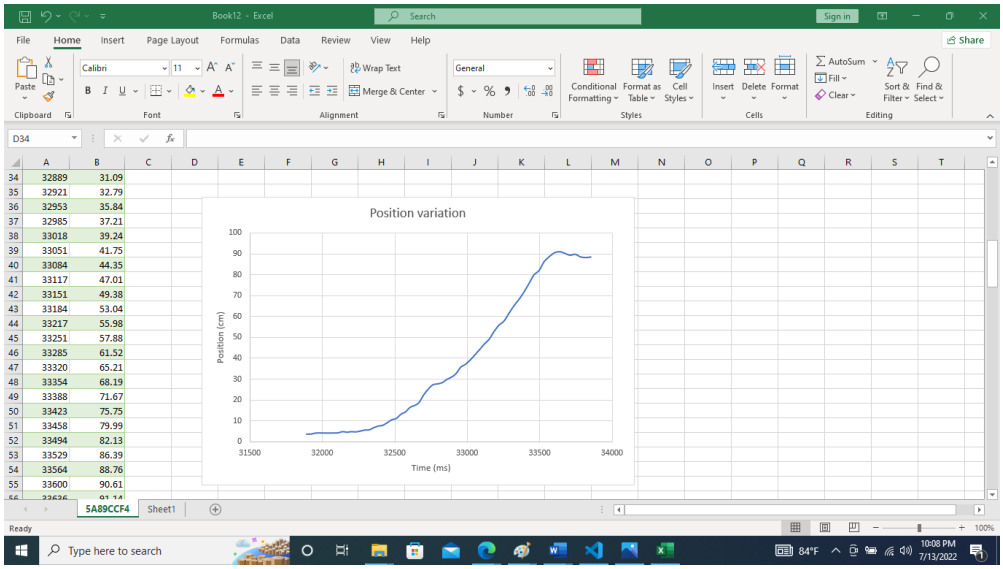


Figure 4.26: The obtained result using the local data acquisition solution

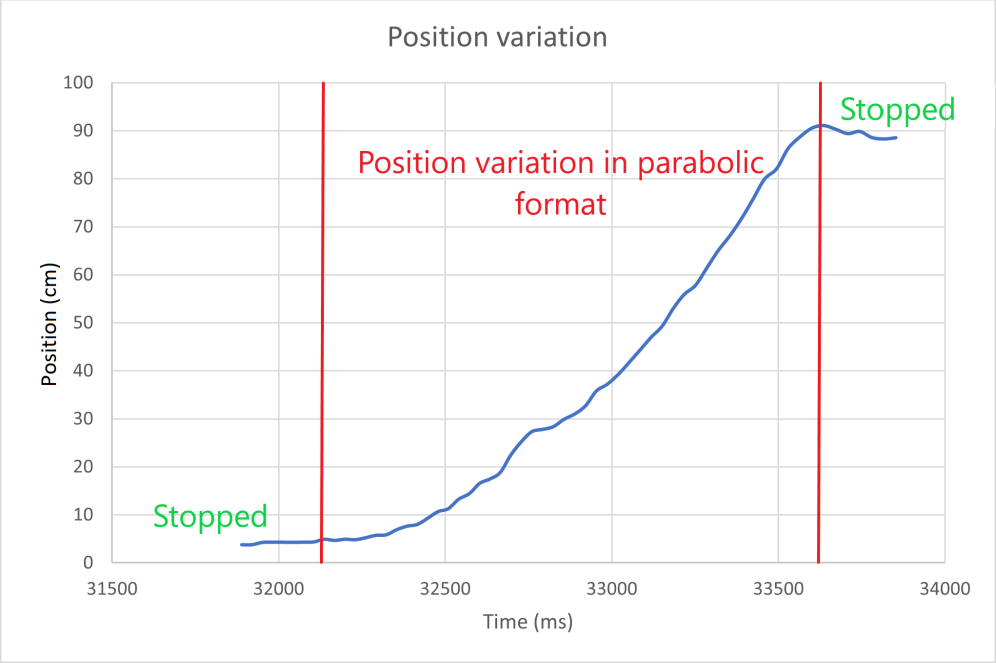


Figure 4.27: Chart of the position variation of the cart

4.3.2 Test and results for Remote Data Acquisition

For remote data acquisition, the same previous web page is stored in the micro SD card (fig. 4.28) and used the inclined plane experiment. The experiment setup is depicted in (fig. 4.29). From the home screen, the user can select the *Inclined Plane Experience*. Although the code is quite similar, the number of points and the interval are different, and so we have to handle.

The position chart can also be displayed on the user's webpage interface. In this case, getting a real-time data chart is possible, however, due to the fact that the rolling cart takes about 2 seconds which is largely insufficient for our web page to receive a good amount of measurements in such a short time, the results will not consistent and several points will get missing.

Therefore, we used an array *dataArr* (listing. 4.11) to store the measurement all along with their corresponding measuring instance.

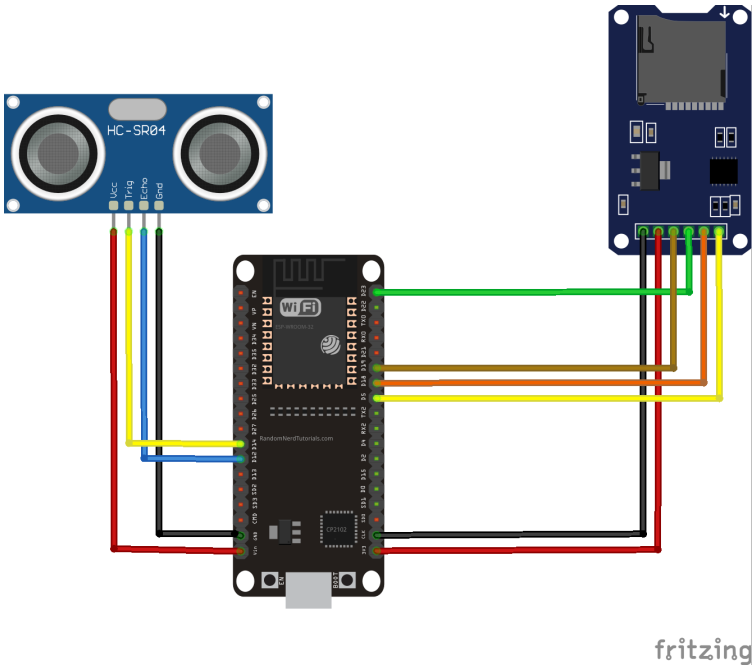


Figure 4.28: Wiring diagram for the ESP32, the HC-sr04, and the micro SD card module

Listing 4.11: Storing the distances in an array

```
chartData = String(currentMillis)+ "," + String(distanceCm);
dataArr[i] = chartData;
```

Hence, the data could be used as soon as the measuring process is done (listing. 4.12). Besides, in this way, we make sure that there are fewer missing points in our chart and the results are more accurate.

Listing 4.12: Sending the measuring data to the web page

```
while( i <sizeArr){
   datatSend = dataArr[i];
     server.on("/trolleyPosition", HTTP_GET, [](AsyncWebServerRequest *request){
            request->send_P(200, "text/plain", datatSend.c_str());
            }); ...
  }
```

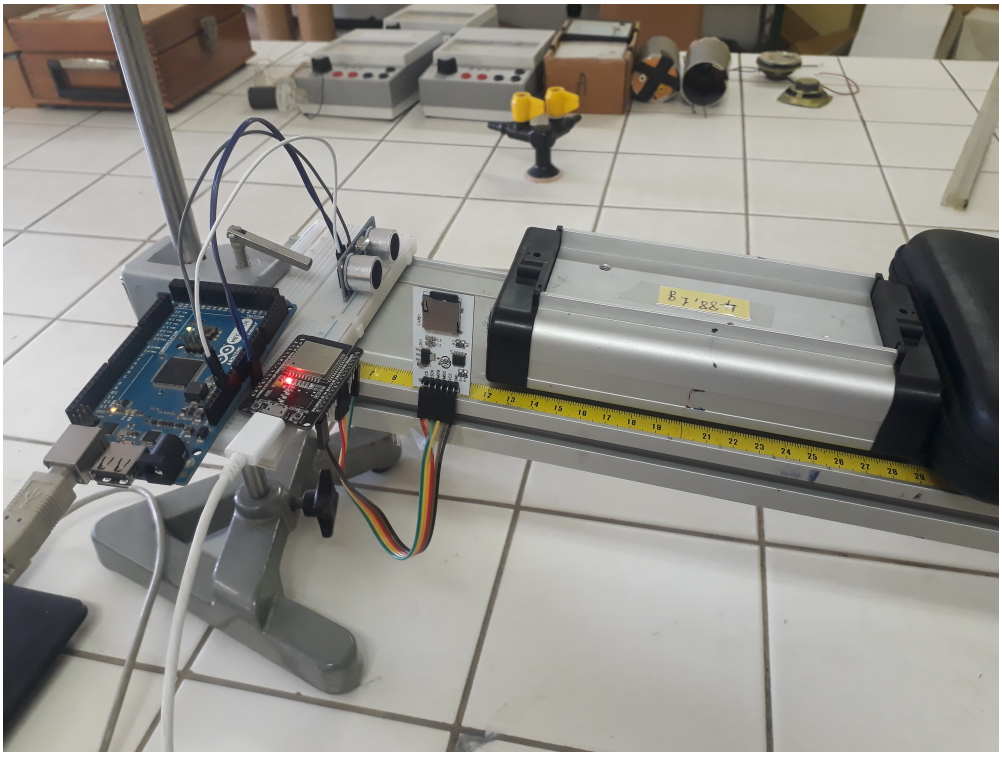


Figure 4.29: Image of the inclined plane connected to the ESP32 for the remote data acquisition

The *onreadystatechange* from the JS code of the *trolleyExperimentApp.js* code will be executed each 30ms which is as same as the delay duration we set in the *sendData()* function from the Arduino code. This allowed us to get fower requests errors. Moreover, the among of points that we need in this situation is fewer than in the previous experiment due to the shot time of the measuring process (100 points) as shown in (listing. 4.13).

Listing 4.13: Js code to send an XMLHttpRequests

```
if(chartT.series[0].data.length > 110) {
          chartT.series[0].addPoint([x, y], true, true, true);
       } else {
          chartT.series[0].addPoint([x, y], true, false, true);
        }
     }else{
       console.log("Error")
```

```
    }if(i == 100){
      clearInterval(refreshIntervalId);
  }
  };
  xhr.open("GET", "/trolleyPosition", true);
  xhr.send();
}, 30) ;
```

The resulting chart are shown in (figs. 4.30, 4.31).

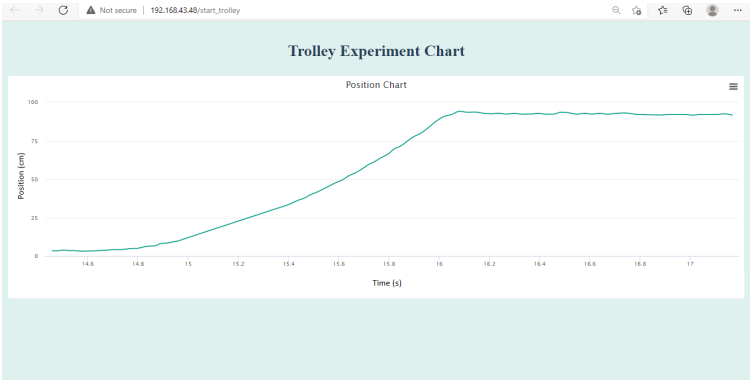


Figure 4.30: Position chart from the web page interface

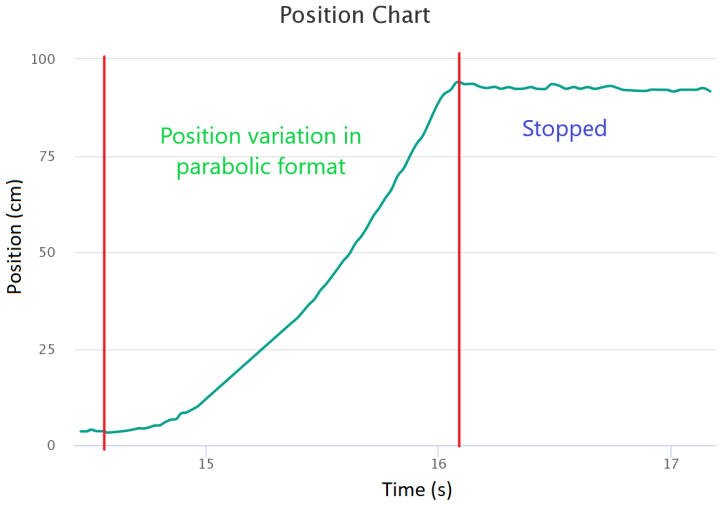


Figure 4.31: Image of the Position-chart of the cart

4.4 Charge/Discharge of capacitor

The capacitor should be empowered for a sufficient time to be fully charged and then discharged through a resistor. The charge and discharge duration depend on the value of the capacitor and the resistor.

The necessary time of a capacitor to get charged is equal to

$$5\tau = 5R_{ch} \times C \tag{4.1}$$

Where:

- R:Resistance (Ω).
- C:Capacity (F).
- τ:characteristic discharge or charge time.

For our experiment we had chose a capacitor with a grand value to allows us to visualize the phenomena in real-time,the capacitor has a value of $220uF$, while the resistor is of $10K\Omega$.From these two value, we can can calculate the necessary duration of charging the capacitor

$$T_{charge} = 5\tau = 5 \times R_{ch} \times C = 5 \times 10 \times 10^3 \times 220 \times 10^{-6} = 5 \times 0.97 = 11s \tag{4.2}$$

So this capacitor needs $11seconds$ to get fully charged.

The capacitor should be empowered using a continuous supply for charging then, the supply should be turned off, so the capacitor could begin discharging.

4.4.1 Schematic of RC circuit with Arduino board

Here,the resistor(10kΩ)and the capacitor(220 μF) are arranged in series.The microcontroller performs successive charges and discharges of the capacitor using Pin 2.

The negative pole of the capacitor is connected to ground GND, and the positive pole is connected to the resistor that are connected the pin A0 , and the other end of the resistor is connected to the pin 2, without forgetting to connect GND to the negative pole its showing in the figure below.

During charging,a voltage of 5V is imposed across the terminal of the resistor and the capacitor.when discharging,the power is cut off and the capacitor discharges through the resistor.The values are read using pin A0 returns a value encoded on 10 bits(0 equals 0V and 1023 equals 5V) ,as show in (fig. 4.32).

The real RC circuit with the Bluno board is showing in figure 4.33

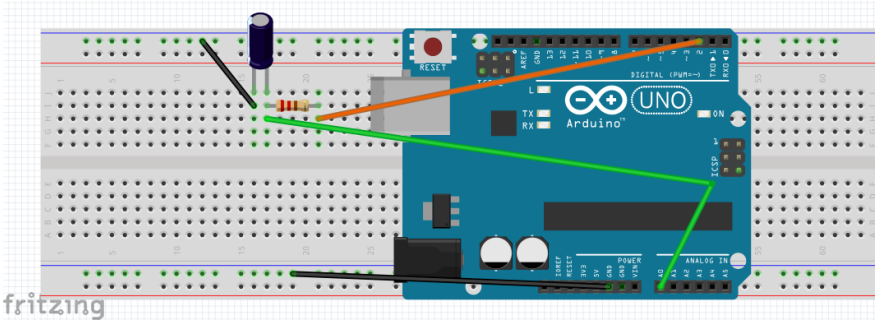


Figure 4.32: RC schematic

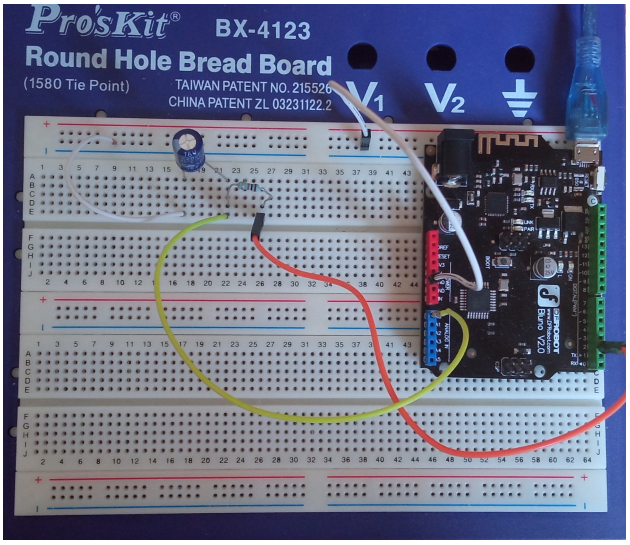


Figure 4.33: The RC circuit.

4.4.2 Local data acquisition

The data of charging and discharging capacitor experiments ,will be displayed on the Python interface on PC using serial communication of Arduino board.

With the impetus given by the new high school programs, the use of microcontrollers in Physics and Chemistry allows an opening on new ways of collecting data from the world around us, and this, by acquiring basic notions of algorithms and programming.

Accessible to the general public, Arduino cards allow you to read data (from a sensor, for example) in real time. In order to access this data, one can for example use:

Arduino IDE serial monitor: the statement is then made point by point in the monitor then the use of a spreadsheet allows to obtain a graphic representation of the data

Arduino IDE serial plotter which allows you to directly obtain the graphic representation.

The objective of this section is to propose a solution that is both complete and practical in order to graphically represent, store and process data from an Arduino card in real time.

Talking about the serial communication protocol of Arduino,when an Arduino board is connected to a computer via a USB port, the protocol used is serial communication. It is this protocol that is used by the Arduino IDE software to transmit or receive data (fig. 4.34).

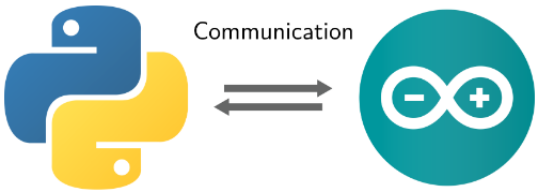


Figure 4.34: Python and serial communication

The information to be transmitted is then divided into blocks of fixed size (10 bits). These blocks are then transmitted asynchronously one after the other at a chosen frequency called the baud-rate (number of characters per second). This protocol will prove to be amply sufficient in most simple applications in Physics-Chemistry but may prove to be limited for the transmission of heavier data which will impose a longer transmission time and thus the

loss of the “real time” character of the experiment.

Here we will try to recover the data by having the possibility of adding a data processing part to it.

It is this communication protocol that we will try to exploit in order to plot the data in real time using Python.

Note:Python has the Pyserial library, allowing the use of serial communication.

4.4.3 Test and results of local data acquisition

The First part concern *Arduino code*, first of we need to include Arduino library by the request *include <Arduino.h>*. The void **setup()** is as follows:

```
  int readvalue = A0; // analog pin (0-1023)
  int power = 2;     // power pin (0-5V)
  int state = 1;    // Charge of capacitor
void setup() {
    Serial.begin(9600);
    pinMode(readvalue, INPUT);
    pinMode(power, OUTPUT);
    digitalWrite(power,HIGH);    // activation of power pin
}
```

The body of this code is presented in the *void loop ()*, we can summarize it as follows: Using the request *analogRead()* for collect the values of voltage on the analog pin A0. Then conversion of the rang of values red from [0to 1023] to[0 to 5v]. Then we put the condition of charge and discharge (listing .4.4.3).

```
  //The void loop function in Arduino code
  void loop() {
 float analog = analogRead(readvalue);
 float uC = (analog/1023)*5 ; // Conversion of voltage
```

```
 if (state == 1) { // Charging
 digitalWrite(power,HIGH);
    if (analog < 1020){ // chech if the capacitor charged completly
      Serial.println(uC);}
 else{
     state = 2;
     digitalWrite(power,LOW);
    }}
 if (state == 2) {
    if (analog > 3){
     Serial.println(uC);
    }else {
      state = 1;
   }
 }
delay(500);
}
```

The second part is the **python code**; firstly, we should import all necessary libraries. For that, we need to define the title of the graph, as well as labeling the X-axis and the Y-axis

After that we will create a main function which contains two sub Functions.The first sub Function is used for management the interruption and creating list of data received by Arduino by serial communication. For the second sub Function used to plot the graph.

After connecting the Arduino board to the computer,We run the Python script a Matplotlib window opens and here is the result as shown in the figure 4.35:

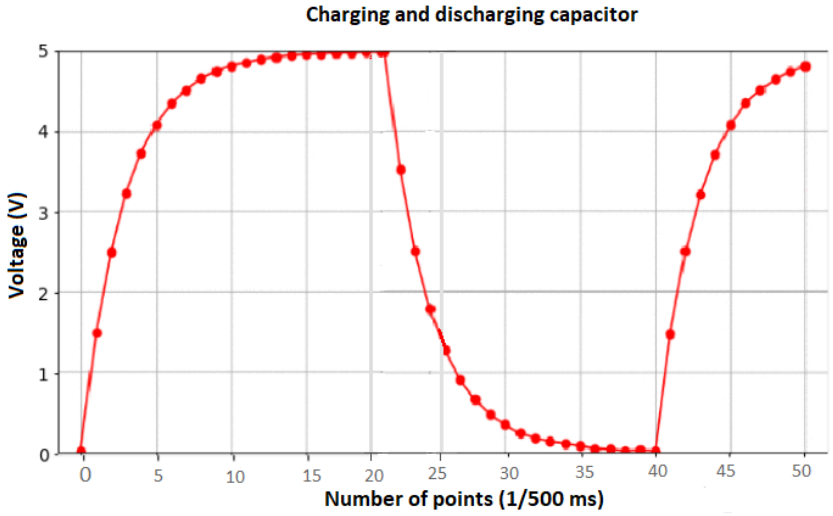


Figure 4.35: Matplotlib window shows the charge and discharge of the capacitor.

After connecting the Arduino board to the computer,We run the Arduino code. As a results, a serial plotter window opens (fig. 4.36):

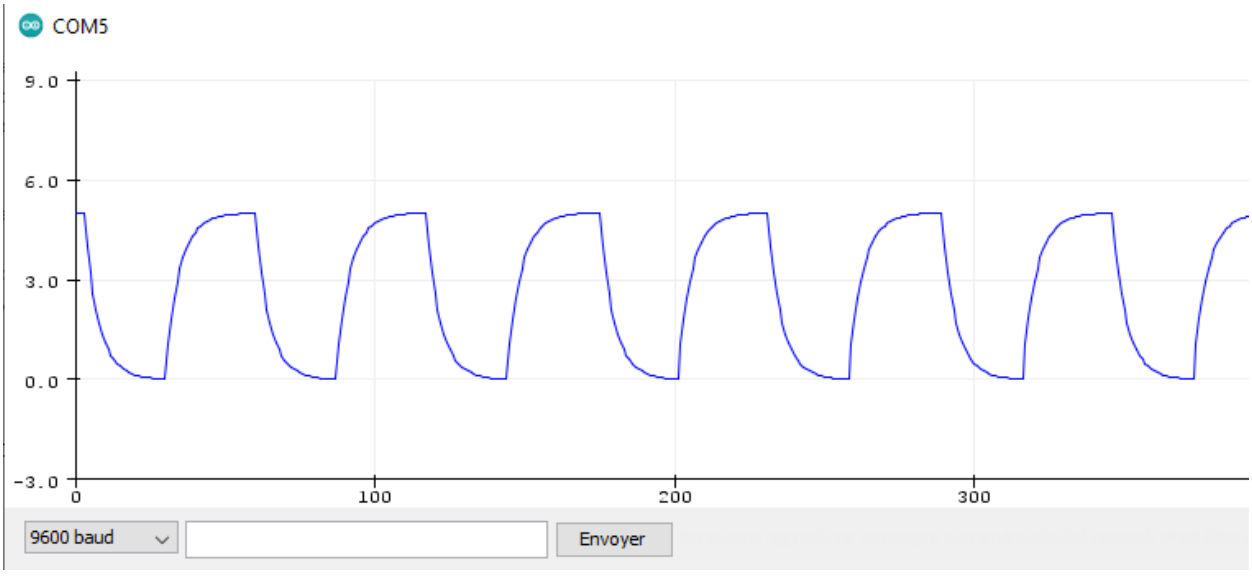


Figure 4.36: chart of charge and discharge of the capacitor in Serial plotter

Here it's possible to graphically estimate the characteristic discharge time by plotting the tangent to the origin of the discharge as the figure 4.37 shows.

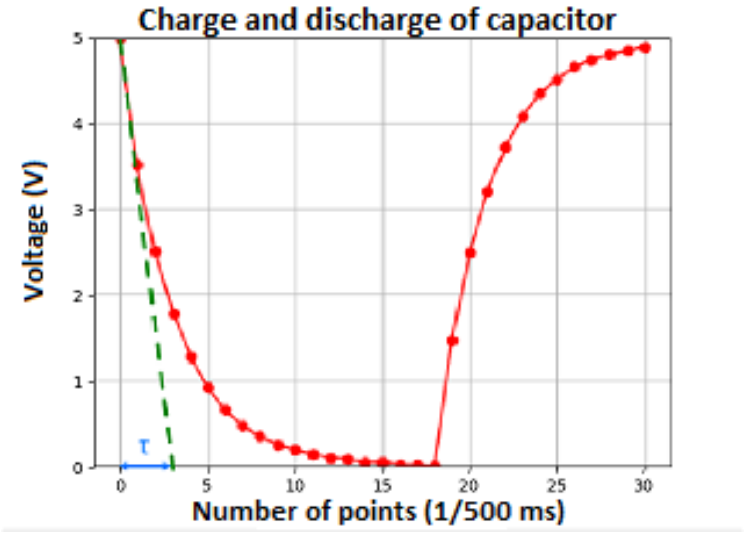


Figure 4.37: Estimate the characteristic discharge time graphically.

We then obtain τ=2,2s taking into account the graphic reading. By comparing with the theoretical valueτ=RC=2,2s,we can conclude that the experimental value provided with its uncertainly seems coherent with the theoretical value.

Coupled with Python, Arduino is therefore useful for performing live acquisitions of simple data. We can add quantities calculated from the data received, for example, for the RC circuit, the voltage across the resistor as in the figure below 4.38:

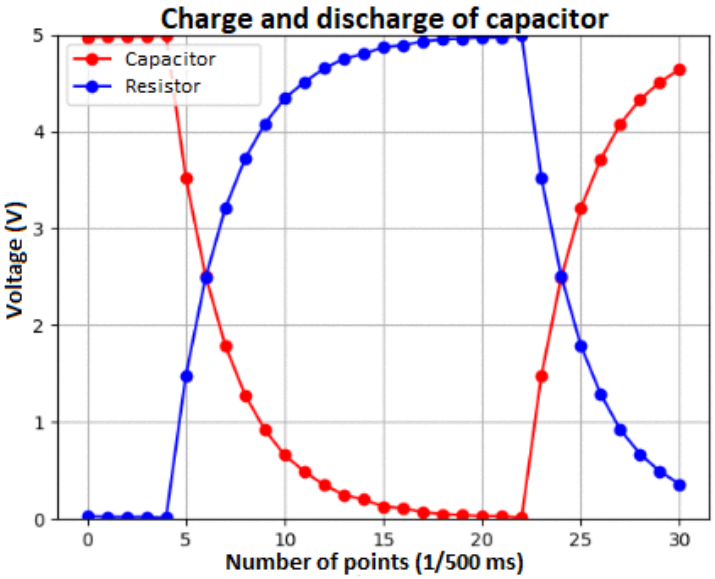


Figure 4.38: The voltage across the resistor and the capacitor in a same graph.

4.5 Conclusion

In this chapter, we presented the different steps in implementing this platform that include two solutions for data acquisition; local and remote. Firstly, for the pendulum experiment, there was an overall presentation of the system. Then a detailed description of local and remote data acquisition. For each solution, we had affected a series of tests. Using the smartphone application allowed us to retrieve the measurement data and then draw the correspondent oscillation chart by using EXCEL. The web page has been tested as well, it allowed us to visualize the oscillation of the pendulum in real-time.
Besides, the famous inclined plane experiment was renewed as well by using an ultrasonic sensor, and the results were quite satisfying for both solutions.

However, while implementing the third experiment which is the charge and discharge of capacitor we encountered some problems when using the ESP32 board, thus we had turned to using a compatible Arduino board "Bluno". Finally, we were able to display the result of the experiment as a chart as a local solution. Using the combination of Arduino IDE and Python IDL (Integrated Development Environment).
To go further, we could imagine reading several values at the same time (with data transmission in the form of a table or in JSON format) or using the bidirectional nature of serial communication to control the Arduino board using Python.

Chapter 5

General conclusion

In this work, we tried to demonstrate that mechatronics could be an efficient tool to enhance the experience of students in practical work. This could be achieved by introducing modern technology to the traditional experiment setups. In this context, we had implemented a platform for physics experiments, the platform consists of three main experiments for the fundamental physics; the pendulum experiment, the inclined plane, and the charge and discharge of a capacitor.

From the pendulum experiment, we were able to achieve the goal which was the graphical presentation of the data, yet using the smartphone application for a chart displaying was not possible, therefore we provided an alternative solution which was using "CSV" files and "EXCEL". This last solution gives excellent results and a smooth chart that the students can use for their studies. Similarly, the results of the web page were satisfying as well, the oscillation chart could be displayed in the real-time at the user interface.

In the same context, the inclined plane experiment for the acceleration studies was implemented in a similar way and based on both results for the local solution and the remote solution were excellent and provided a clear format of the chart that should be drawn by the different positions of the trolley.

Finally, we concluded with the charge and discharge of a capacitor. Although we were not

able to use the same smartphone application and the web page for the data acquisition, we turned to using Python IDL as a solution for local data acquisition.

From the obtained results, we can assume that such a platform could present a reliable alternative to traditional experiment setups, data acquisition using the smartphone app or through the web page should eliminate problems related to human error during measurement. Moreover, these kinds of experiments attract students' attention more than the conventional ones and introduce them to modern technologies at an earlier stage of their education. Furthermore, the web solution turns the measuring data available outside the laboratory which could be the first step for further development in this thematic.

The extension of this platform is possible by adding more functionalities to the wireless solutions, especially for the web page. For instance, we can extend the remote solution to allow the user to control the start of the experience remotely by adding actuators to our experiment setups. This will even duplicate the benefits that students could acquire.

Appendix A

Appendix

A.1 Arduino code

Arduino code for the pendulum experiment for local data acquisition

```
// ------------------------------------------------
//
// Code for control of ESP32 through MIT inventor app (Bluetooth).
// device used for tests: ESP32-WROOM-32D
// Experiment: Fundamental physics Pendulum experiment using ESP32 AND MIT app
    inventor.
// Author: SALAG Aicha
//
// ------------------------------------------------
#include "BluetoothSerial.h"
#include "LiquidCrystal.h"
// init Class:
BluetoothSerial ESP\_BT;// Object of BluetoothSerial class
```

```
LiquidCrystal lcd(22,23,5,18,19,21);
// Input variables
char inputByte = 'd';
unsigned long StartMillis;
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long duration = 12000; // the duration of measurement process
const unsigned long period = 30; // the duration between two measures
const byte potentiometerPin = 4;
int timeArr[500];
float valueArr[500];
// measuring of voltage
void pendulumMeasure(){
  int i = 0;
     StartMillis = millis();
     startMillis = millis();
     while(millis()-StartMillis <=duration){
        currentMillis = millis();
        while (currentMillis - startMillis >= period){
        lcd.setCursor(0,1);
        lcd.print(".");
        int Pot_Ana_Val = analogRead(potentiometerPin); // read the analog
           value the potentiometer (of pin 4)
        // Convert the analog value to digital value
        float Voltage = Pot_Ana_Val * (3.3 / 4095.0);
        timeArr[i] = currentMillis; //currentMillis = millis();
        startMillis = currentMillis;
        valueArr[i] = Voltage;
          Serial.print("index : " );
```

```
            Serial.println(i);
            Serial.print("Voltage : ");
            Serial.println(Voltage);
            i += 1;
          }
    }
    lcd.setCursor(0,1);
    lcd.print("Measuring Done!");
}
// Send data
void sendData(){
      lcd.clear();
      lcd.print("Sending Data! ");
      ESP_BT.print("Time");
      ESP_BT.print(";");
      ESP_BT.print("Voltage");
      ESP_BT.print("\n");
     for(int i = 0; i<400 ; i++){
      ESP_BT.print(timeArr[i]);
      ESP_BT.print(";");
      ESP_BT.print(valueArr[i]);
      ESP_BT.print("\n");
      Serial.println(i);
      lcd.setCursor(0,1);
      lcd.print (".");
      delay(100);
      if (timeArr[i]==0){
        Serial.print("DONE" );
        lcd.setCursor(0,1);
```

```
        lcd.print ("Sending Done!");
        break;
        }
      }
      lcd.setCursor(0,1);
      lcd.print ("Sending Done!");
      Serial.print("DONE" );
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200); // Serial communication bandwidth 115200 bits per second
  ESP_BT.begin("ESP32_Board"); // Bluetooth interface name -> will show up on
      your phone
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("ESP32_Board");
}
void loop() {
     int Pot_Ana_Val = analogRead(potentiometerPin); // read the analog value the
         potentiometer (of pin 4)
          // Convert the analog value to digital value
     float Voltage = Pot_Ana_Val * (3.3 / 4095.0);
     lcd.clear();
     lcd.print("Voltage: ");
     lcd.setCursor(0,1);
     lcd.print (Voltage);
     delay(1000);
      if(Voltage < 1.4 || Voltage > 1.5){
```

```
        Serial.println(Voltage);
        delay(1000);
        pendulumMeasure();
        lcd.clear();
        lcd.print("Measuring... ");
        lcd.setCursor(0,1);
        lcd.print (Voltage);
        }
       if (ESP_BT.available() > 0){
        lcd.clear();
        lcd.print("Connected! ");
        lcd.setCursor(0,1);
        lcd.print (Voltage);
        inputByte = ESP_BT.read();
        Serial.println("Start data acquisition");
        if(inputByte == 'T'){
        sendData();
         }
      }
}
```

A.1.1 Arduino code for the pendulum experiment for remote data acquisition

```
// ------------------------------------------------
//
// device used for tests: ESP32-WROOM-32D
```

```
// Experiment: Fundamental physics Pendulum experiment that is connected to a
    web page.
// Author: SALAG Aicha
//
// ------------------------------------------------
// SD card Libraries
#include "FS.h"
#include "SD.h"
#include "SPI.h"
// Libraries to get time from NTP Server
#include "WiFi.h"
// Library for Asynchrone WebServer
#include "AsyncTCP.h"
#include "ESPAsyncWebServer.h"
// Network confidentials
const char* ssid = "AndroidAP2398";
const char* password = "cytc7060";
// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
// Variables
String state;
unsigned long StartMillis; //
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long duration = 12000; // the duration of measurement process
const unsigned long period = 3; // the duration between two measures
int potentiometerPin = 34;
int LED_BUILTIN = 2;
String dataMessage;
```

```
String chartData;
String dataArray[400];
// Initialize WiFi
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}
// Initialize SD card
void initSDCard(){
   if (!SD.begin()) {
    Serial.println("Card Mount Failed");
    return;
  }
  uint8_t cardType = SD.cardType();
  if(cardType == CARD_NONE){
    Serial.println("No SD card attached");
    return;
  }
  Serial.print("SD Card Type: ");
  if(cardType == CARD_MMC){
    Serial.println("MMC");
  } else if(cardType == CARD_SD){
    Serial.println("SDSC");
```

```
  } else if(cardType == CARD_SDHC){
    Serial.println("SDHC");
  } else {
    Serial.println("UNKNOWN");
  }
  uint64_t cardSize = SD.cardSize() / (1024 * 1024);
  Serial.printf("SD Card Size: %lluMB\n", cardSize);
}
// Write to the SD card
void writeFile(fs::FS &fs, const char * path, const char * message) {
  Serial.printf("Writing file: %s\n", path);
  File file = fs.open(path, FILE_WRITE);
  if(!file) {
    Serial.println("Failed to open file for writing");
    return;
  }
  if(file.print(message)) {
    Serial.println("File written");
  } else {
    Serial.println("Write failed");
  }
  file.close();
}
// Append data to the SD card
void appendFile(fs::FS &fs, const char * path, const char * message) {
  Serial.printf("Appending to file: %s\n", path);

  File file = fs.open(path, FILE_APPEND);
  if(!file) {
```

```
    Serial.println("Failed to open file for appending");
    return;
  }
  if(file.print(message)) {
    Serial.println("Message appended");
  } else {
    Serial.println("Append failed");
  }
  file.close();
}
String processor(const String& var){
  Serial.println(var);
  return String();
}
void startMeasuring(){
    int i = 0;
      StartMillis = millis();
      startMillis = millis();
      while(millis()-StartMillis <=duration){
          currentMillis = millis();
          while (currentMillis - startMillis >= period){
          int Pot_Ana_Val = analogRead(potentiometerPin); // read the analog
              value the potentiometer (of pin 4)
          // Convert the analog value to digital value
          float Voltage = Pot_Ana_Val * (3.3 / 4095.0);
          startMillis = currentMillis;
          dataMessage = String(currentMillis) + ";" + String(Voltage)+ "\r\n";
          //Append the data to file
          appendFile(SD, "/data.txt", dataMessage.c_str());
```

```
        Serial.println(Voltage);
        Serial.println(i);
        i++;
        chartData = String(currentMillis)+ "," + String(Voltage);
        server.on("/pendulumVoltage", HTTP_GET, [](AsyncWebServerRequest
            *request){
        request->send_P(200, "text/plain", chartData.c_str());
        });
    }
   }
   state = "DONE";
}
void setup() {
   Serial.begin(115200);
   initWiFi();
   initSDCard();
    // If the data.txt file doesn't exist
    // Create a file on the SD card and write the data labels
    File file = SD.open("/data.txt");
    if(!file) {
      Serial.println("File doesn't exist");
      Serial.println("Creating file...");
      writeFile(SD, "/data.txt", "Measuring instance, Voltage \r\n");
    }else {
      Serial.println("File already exists");
    }
    file.close();
    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SD, "/index.html", "text/html");
```

```
    });
    server.serveStatic("/", SD, "/");
    server.on("/start", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(LED_BUILTIN, HIGH);
    state = "START";
    request->send(SD, "/pendulumExperiment.html", String(), false, processor);
    });
    server.begin();
}
void loop() {
  if(state == "START"){
    startMeasuring();
    }
```

A.1.2 Arduino code for the inclined plane experiment for local data acquisition

```
// ------------------------------------------------
// Code for control of ESP32 through MIT inventor app (Bluetooth).
// device used for tests: ESP32-WROOM-32D
// Experiment: Fundamental physics acceleration measurement experiment using
    ESP32 AND MIT app inventor.
// Author: SALAG Aicha
//
// ------------------------------------------------
#include "BluetoothSerial.h"
#include "LiquidCrystal.h"
//------------------------------------------------
```

```
// Define sound speed in cm/us
#define SOUND_SPEED 0.034 // sound Speed
//-------------------------------------------------
// init Class:
BluetoothSerial ESP_BT;// Object of BluetoothSerial class
//LCD pin setting
LiquidCrystal lcd(22,23,5,18,19,21);
//-------------------------------------------------
// variables
int sizeArr;
unsigned long StartMillis;
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long measuringDuration = 5000; // the duration of measurement
    process
const unsigned long period = 50; // the duration between two measures, 40 ou 50;
const int trigPin = 14; // Trigger pin
const int echoPin = 12; //Echo pin
long duration; // Elapsed time variable
float distanceCm;
boolean bt = false;
int timeArr[500];
float valueArr[500];

void distanceMeasuring(){
   int i = 0;
   StartMillis = millis();
   startMillis = millis();
   lcd.setCursor(0,1);
```

```
    lcd.print(".");
    while(millis()-StartMillis <= 5000){
                digitalWrite(trigPin, LOW);
                delayMicroseconds(2);
                digitalWrite(trigPin, HIGH);
                delayMicroseconds(10);
                digitalWrite(trigPin, LOW);
                duration = pulseIn(echoPin, HIGH);
                timeArr[i] = millis();
                distanceCm = duration * SOUND_SPEED/2;
                valueArr[i] = distanceCm;
                Serial.print("index : " );
                Serial.println(i);
                Serial.print("Distance : ");
                Serial.println(distanceCm);
                i += 1;
               delay(30);
    }
          lcd.setCursor(0,1);
          lcd.print("Measuring Done!");

}
void sendData(){
      lcd.clear();
      lcd.print("Sending Data! ");
      ESP_BT.print("Time");
      ESP_BT.print(";");
      ESP_BT.print("Distance");
      ESP_BT.print("\n");
```

```
    for(int i = 0; i<400 ; i++){
      ESP_BT.print(timeArr[i]);
      ESP_BT.print(";");
      ESP_BT.print(valueArr[i]);
      ESP_BT.print("\n");
      Serial.println(i);
      lcd.setCursor(0,1);
      lcd.print (".");
      delay(100);
      if (timeArr[i]==0){
        Serial.print("DONE" );
        lcd.setCursor(0,1);
        lcd.print ("Sending Done!");
        break;
        }
      }
      lcd.setCursor(0,1);
      lcd.print ("Sending Done!");
      Serial.print("DONE" );
}
void setup() {
    Serial.begin(115200); // Starts the serial communication
    ESP_BT.begin("ESP32_Board"); // Bluetooth interface name -> will show up on
        your phone
    lcd.begin(16, 2);
    lcd.clear();
    lcd.print("ESP32_Board");
    pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
```

```
}
void loop(){
  if (ESP_BT.available() > 0){
        inputByte = ESP_BT.read();
        if(inputByte == 'A'){
        distanceMeasuring();
        }
          lcd.clear();
          lcd.print("Measuring... ");
          lcd.setCursor(0,1);
          lcd.print (distanceCm);

            Serial.println("Start data acquisition");
          if(inputByte == 'T'){
        sendData();
         }
        }
}
```

A.1.3 Arduino code for the inclined plane for the remote data acquisition

```
// ------------------------------------------------
// device used for tests: ESP32-WROOM-32D
// Experiment: Fundamental physics Pendulum experiment that is connected to a
    web page.
// Author: SALAG Aicha
// ------------------------------------------------
```

```
// SD card Libraries
#include "FS.h"
#include "SD.h"
#include "SPI.h"
// Libraries to get time from NTP Server
#include "WiFi.h"
// Library for Asynchrone WebServer
#include "AsyncTCP.h"
#include "ESPAsyncWebServer.h"
// Network confidentials
const char* ssid = "AndroidAP2398";
const char* password = "cytc7060";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
//---------------------------------------------------
// Define sound speed in cm/us
#define SOUND_SPEED 0.034 // sound Speed
//---------------------------------------------------

// Variables
int sizeArr;
char inputByte = 'd';
String state;
unsigned long StartMillis; //
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long measuringDuration = 3000; // the duration of measurement
    process
```

```
const unsigned long period = 30; // the duration between two measures
const int trigPin = 14; // Trigger pin
const int echoPin = 12; //Echo pin
int LED_BUILTIN = 2;
long duration; // elapsed time
float distanceCm;
String dataMessage;
String chartData;
String datatSend ;
String dataArr[500];

// Initialize WiFi
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

// Initialize SD card
void initSDCard(){
   if (!SD.begin()) {
    Serial.println("Card Mount Failed");
    return;
  }
```

```
  uint8_t cardType = SD.cardType();

  if(cardType == CARD_NONE){
    Serial.println("No SD card attached");
    return;
  }
  Serial.print("SD Card Type: ");
  if(cardType == CARD_MMC){
    Serial.println("MMC");
  } else if(cardType == CARD_SD){
    Serial.println("SDSC");
  } else if(cardType == CARD_SDHC){
    Serial.println("SDHC");
  } else {
    Serial.println("UNKNOWN");
  }
  uint64_t cardSize = SD.cardSize() / (1024 * 1024);
  Serial.printf("SD Card Size: %lluMB\n", cardSize);
}

// Write to the SD card
void writeFile(fs::FS &fs, const char * path, const char * message) {
  Serial.printf("Writing file: %s\n", path);

  File file = fs.open(path, FILE_WRITE);
  if(!file) {
    Serial.println("Failed to open file for writing");
    return;
  }
```

```
  if(file.print(message)) {
    Serial.println("File written");
  } else {
    Serial.println("Write failed");
  }
  file.close();
}

// Append data to the SD card
void appendFile(fs::FS &fs, const char * path, const char * message) {
  Serial.printf("Appending to file: %s\n", path);

  File file = fs.open(path, FILE_APPEND);
  if(!file) {
    Serial.println("Failed to open file for appending");
    return;
  }
  if(file.print(message)) {
    Serial.println("Message appended");
  } else {
    Serial.println("Append failed");
  }
  file.close();
}
String processor(const String& var){
  Serial.println(var);
  return String();

}
```

```
void distanceMeasuring(){
   digitalWrite(LED_BUILTIN, HIGH);


   int i = 0;
   StartMillis = millis();
   startMillis = millis();
   while(millis()-StartMillis <=measuringDuration){
               digitalWrite(trigPin, LOW);
               delayMicroseconds(2);
               digitalWrite(trigPin, HIGH);
               delayMicroseconds(10);
               digitalWrite(trigPin, LOW);
               currentMillis = millis();
               duration = pulseIn(echoPin, HIGH);
               distanceCm = duration * SOUND_SPEED/2;
               dataMessage = String(currentMillis) + ";" + String(distanceCm )+
                   "\r\n";
               //Append the data to file
               appendFile(SD, "/dataTrolleyExperiment.txt", dataMessage.c_str());
               Serial.println(distanceCm);
               Serial.println(i);
               chartData = String(currentMillis)+ "," + String(distanceCm);
               dataArr[i] = chartData;
//              server.on("/trolleyPosition", HTTP_GET, [](AsyncWebServerRequest
    *request){
//              request->send_P(200, "text/plain", chartData.c_str());
//              });
               i++;
```

```
   }


   sizeArr = i;
   state = "DONE";
}
void sendData(){
  int i = 0;
  while( i <sizeArr){
     datatSend = dataArr[i];
       server.on("/trolleyPosition", HTTP_GET, [](AsyncWebServerRequest *request){
               request->send_P(200, "text/plain", datatSend.c_str());
               });
              Serial.print(i);
               delay(30);
               i++;
    }
  }
void setup() {
   Serial.begin(115200);
   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
   initWiFi();
   initSDCard();
    // If the data.txt file doesn't exist
    // Create a file on the SD card and write the data labels
    File file = SD.open("/dataTrolleyExperiment.txt");
    if(!file) {
      Serial.println("File doesn't exist");
      Serial.println("Creating file...");
```

```
      writeFile(SD, "/dataTrolleyExperiment.txt", "Measuring instance, Distance
          \r\n");
    }else {
      Serial.println("File already exists");
    }
    file.close();
    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SD, "/index.html", "text/html");
    });
    server.serveStatic("/", SD, "/");
    server.on("/start_trolley", HTTP_GET, [](AsyncWebServerRequest *request){
    state = "START";
    request->send(SD, "/trolleyExperiment.html", String(), false, processor);
    });
    server.begin();
}
void loop() {
     inputByte = Serial.read();
     digitalWrite(trigPin, LOW);
     delayMicroseconds(2);
     digitalWrite(trigPin, HIGH);
     delayMicroseconds(10);
     digitalWrite(trigPin, LOW);
     currentMillis = millis();
     duration = pulseIn(echoPin, HIGH);
     distanceCm = duration * SOUND_SPEED/2;
     if (distanceCm >= 8 && distanceCm <= 12 ){
      Serial.print("distance equals to " );
      Serial.println (distanceCm);
```

```
      distanceMeasuring();
      }
     if(state == "START"){
        sendData();
    }
}
```

A.2 HTML code

A.2.1 HTML code for the home page

<!DOCTYPE html> <html> <head> <meta charset=”utf-8”> <title>Physics Experiment</title> <link rel=”icon” type=”image/png” href=”pendulumIcon.png”> <link href=”style.css” type = ”text/css” rel = ”stylesheet”> </head> <body>   <h1 class=”title”>Remote Data Acquisition for Physics Experiments</h1>   <h3 class=”header2”>Pendulum Experiment </h3> <img src=”pendulum.png” class=”image”> <button id = ”pendulumElement” class=”button” placeholder = %STARTpendulum% > Pendulum experiment</button>   <h3 class=”header2”> Inclined Plane Experiment</h3> <img src=”inclinePlane.jpg” class=”image”> <button id = ”trolleyElement” class=”button” placeholder = %START-pendulum% > Inclined plane experiment</button>   This page is implemented to let users acquire data from the two fundamental physics experiments; the Pendulum experiment and the inclined plane experiment.   <h4>Contact us</h4> Email: a_salag@enst.dz / g_hamache@enst.dz   <script src = ”app.js”></script> </body> </html>

A.2.2 HTML code for pendulum experiment page

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Pendulum experiment </title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
    <link rel="icon" type="image/png" href="oscillationIcon.png">
    <script src="https://code.highcharts.com/stock/highstock.js"></script>
    <script src="https://code.highcharts.com/modules/exporting.js"></script>
</head>
<body>
    <h1 id = "title" class="title" > Pendulum experiment WEBPAGE </h1>
    <div id="pedulumChart" class="chartContainer"></div>
     <script src = "pendulumExperimentApp.js"></script>
</body>
</html>
```

A.2.3 HTML code for inclined plane experiment

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Pendulum experiment </title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
    <link rel="icon" type="image/png" href="oscillationIcon.png">
    <script src="https://code.highcharts.com/stock/highstock.js"></script>
```

```
    <script src="https://code.highcharts.com/modules/exporting.js"></script>
</head>
<body>
    <h1 id = "title" class="title" > Trolley Experiment Chart </h1>
    <div id="PositionChart" class="chartContainer"></div>
     <script src = "trolleyExperimentApp.js"></script>
</body>
</html>
```

A.3 CSS code

```
body{
    margin: 0;
}
.container{
    display: flex;
    width: 100vw;
    height: 100vh;
    /* justify-content: center;
    align-items: center; */
    font-family: 'Quicksand', sans-serif;
    font-weight: bold;
    font-size: 20px;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: 100px 1fr 1fr 100px;
    gap: 10px;
    padding: 10px;
```

```
    box-sizing: border-box;
}
.header{
    grid-column-start: 1;
    grid-column-end: 4;
    color: #2A3E59;
    padding: 10px;
    background-color: #C4EEF2;
    text-align: center;
}
.title{
    font-size: 30px;
}
.content-large{
    grid-row-start: 2;
    grid-row-end: span 2;
}
.content-small{
    grid-row-end: span 2;
    display: flex;
    align-items: center;
    justify-content: center;
}
.paragraph{
    text-align: center;
}
.footer{
    grid-column: 1/ span 3;
    display: flex;
```

```
    align-items: center;
    justify-content: center;
    background-color: #C4EEF2;
}
.image{
    flex: auto;
    height: 270px;
    width: 400px;
    align-content: center;
}
.header2{
    color: #2A3E59;
    text-align: center;
    text-align: center;
    padding: auto;
}
.button {
    display: inline-block;
    background-color: #C4EEF2;
    border: none;
    border-radius: 4px;
    color: black;
    padding: 16px 40px;
    text-decoration: none;
    font-size: 25px;
    margin: 2px;
    cursor: pointer;
    text-align: center;
    align-items: center;
```

```
    align-content: center;
    width: 400px;
}
```

A.4 JavaScript code

A.4.1 JavaScript code of the home page

```
document.getElementById("pendulumElement").addEventListener('click', function(e){
  window.open("/start_pendulum");
});
document.getElementById("trolleyElement").addEventListener('click', function(e){
  window.open("/start_trolley");
});
```

A.4.2 JavaScript code for the pendulum oscillation chart

```
var data = document.getElementById("data");
var chartT = new Highcharts.Chart({
    chart:{ renderTo : 'pedulumChart',
     type: 'spline'
     },
    title: { text: 'Pendulum oscillation' },
    series: [{
      showInLegend: false,
      data: []
    }],
    plotOptions: {
```

```
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: { text: 'Time (s)'},
  yAxis: {
    title: { text: 'Voltage (volt)'
  },
  credits: { enabled: false }
});
var i = 0;
refreshIntervalId =
setInterval(function ( ) {
  i+=1;
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
         console.log("i" + i);
      var chartInput = this.responseText;
         console.log(typeof chartInput);
         console.log(chartInput);
      var xyArray = chartInput.split(",");
         console.log(xyArray[0]);
         console.log(xyArray[1]);
         x = 0.001*parseFloat(xyArray[0]);
         y = parseFloat(xyArray[1]);
         console.log("length"+ chartT.series[0].data.length);
       if(chartT.series[0].data.length > 500) {
```

```
        chartT.series[0].addPoint([x, y], true, true, true);
      } else {
        chartT.series[0].addPoint([x, y], true, false, true);
       }
    }
    else{
      console.log("Error")
    }
    if(i == 400){
      clearInterval(refreshIntervalId);
  }
  };
  xhr.open("GET", "/pendulumVoltage", true);
  xhr.send();
 }, 30) ;
```

A.4.3 JavaScript code for drawing the chart

```
var chartT = new Highcharts.Chart({
  chart:{ renderTo : 'PositionChart',
  type: 'spline'
   },
  title: { text: 'Position Chart' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
```

```
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: { text: 'Time (s)'},
  yAxis: {
    title: { text: 'Position (cm)' }

  },
  credits: { enabled: false }
});
var i = 0;
refreshIntervalId =
setInterval(function ( ) {
  i+=1;
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {

    // console.log(this);
    if (this.readyState == 4 && this.status == 200) {
         // data = parseFloat(this.responseText);

         console.log("i" + i);
      var chartInput = this.responseText;
         console.log(typeof chartInput);
         console.log(chartInput);
      var xyArray = chartInput.split(",");
         console.log(xyArray[0]);
```

```
            console.log(xyArray[1]);
            x = 0.001*parseFloat(xyArray[0]);
            y = parseFloat(xyArray[1]);
            console.log("length"+ chartT.series[0].data.length);
         if(chartT.series[0].data.length > 110) {
          chartT.series[0].addPoint([x, y], true, true, true);
        } else {
          chartT.series[0].addPoint([x, y], true, false, true);
         }
      }
      else{
        console.log("Error")
      }
      if(i == 100){
        clearInterval(refreshIntervalId);
    };
    };
    xhr.open("GET", "/trolleyPosition", true);
    xhr.send();
  }, 30) ;
```

Bibliography

[1] R. W. Habash and C. Suurtamm, "Engaging high school and engineering students: A multifaceted outreach program based on a mechatronics platform," *IEEE transactions on education*, vol. 53, no. 1, pp. 136–143, 2009.

[2] H.-J. Lee and H. Yi, "Development of an onboard robotic platform for embedded programming education," *Sensors*, p. 3916, 2021.

[3] M.-H. Phan and H. Q. T. Ngo, "A multidisciplinary mechatronics program: From project-based learning to a community-based approach on an open platform," *Electronics*, p. 954, 2020.

[4] A. El Moussaouy, M. El Hadi, A. Ouariach, R. Essaadaoui, D. Bria, and K. Laabidi, "Simple conception of photoresistor cell current-voltage characteristic measurement by arduino-based platform for educational purposes," in *International Conference on European Transnational Education*, (Maroco), pp. 348 – 356, Springer, 2020.

[5] R. McHugh and H. Zhang, "Virtual prototyping of mechatronics for 21 st century engineering and technology," in *Proceedings of the 2008 ASEE Midwest Conference*, pp. 1–12, 2011.

[6] Č. Kodejška, G. De Nunzio, R. Kubínek, and J. Říha, "Low cost alternatives to commercial lab kits for physics experiments," *Physics Education*, p. 597, 2015.

[7] S. Uğur and T. KIRINDI, "Using arduino in physics teaching: arduino-based physics experiment to study temperature dependence of electrical resistance," *Journal of Computer and Education Research*, vol. 7, no. 14, pp. 698–710, 2019.

[8] C. Lellis-Santos and F. Abdulkader, "Smartphone-assisted experimentation as a didactic strategy to maintain practical lessons in remote education: alternatives for physiology education during the covid-19 pandemic," *Advances in physiology education*, vol. 44, no. 4, pp. 579–586, 2020.

[9] T. P. Cabré, A. S. Vela, M. T. Ribes, J. M. Blanc, J. R. Pablo, and F. C. Sancho, "Didactic platform for dc motor speed and position control in z-plane," *ISA transactions*, vol. 118, pp. 116–132, 2021.

[10] T. Pierratos and H. M. Polatoglou, "Utilizing the phyphox app for measuring kinematics variables with a smartphone," *Physics Education*, vol. 55, no. 2, p. 025019, 2020.

[11] E. Adhitama and A. Fauzi, "A novel real-time data acquisition using an excel spreadsheet in pendulum experiment tool with light-based timer," *Physics Education*, pp. 035 – 002, 2018.

[12] G. Urban, "Jacob fraden: Handbook of modern sensors: physics, designs, and applications," 2016.

[13] Cytron Technologies, *User's Manual HC-SR04 ultrasonic sensor*, 5 2013.

[14] A. S. Huang and L. Rudolph, *Bluetooth essentials for programmers*. Cambridge University Press, 2007.

[15] D. DuRocher, *HTML & CSS QuickStart Guide: The Simplified Beginners Guide to Developing a Strong Coding Foundation, Building Responsive Websites, and Mastering the Fundamentals of Modern Web Design*. ClydeBank Media LLC, 2021.

[16] D. Flanagan, *Javascript: The Definitive Guide: Master the World's Most-used Programming Language*, vol. 1245. O'reilly, 2020.

[17] M. Haverbeke, *Eloquent JavaScript: a modern introduction to programming*. No Starch Press, 2018.

[18] J. N. Robbins, *Learning web design: A beginner's guide to HTML, CSS, JavaScript, and web graphics*. " O'Reilly Media, Inc.", 2012.

[19] D. Gourley, B. Totty, M. Sayer, A. Aggarwal, and S. Reddy, *HTTP: the definitive guide*. " O'Reilly Media, Inc.", 2002.

[20] R. Nicklin and J. Rafert, "The digital pendulum," *American Journal of Physics*, vol. 52, no. 7, pp. 632–639, 1984.