Parachute Simulator Project

Edos Osazuwa
Supervisor: Dr. Anna clarke
Technion - Aerospace Engineering Department

July 6, 2024

Contents

Abstract

This report details the development and implementation of a parachute simulator, designed for educational purposes. The system includes physical components and software for measurement, providing a comprehensive tool for students to learn about parachute dynamics and control mechanisms.

1 Introduction

The parachute simulator project aims to create a hands-on learning tool for students to understand the principles of parachute operation and control. The simulator mimics the physical experience of controlling a parachute via a seated setup with handles connected to rubber bands. This report covers the system's design, components, and educational benefits.

2 System Description and Educational Usefulness

2.1 System Description

The simulator consists of a seat, two handles connected to the ceiling via rubber bands, and a measurement system that captures the force applied on the handles. The forces are measured using load sensors connected to an Arduino Uno and processed with an HX-711 amplifier and Load cells.

2.2 Educational Usefulness

This simulator provides students with a practical understanding of parachute mechanics and control systems. By manipulating the handles, students can experience how changes in force affect parachute dynamics. This hands-on approach enhances theoretical learning and provides valuable insights into real-world applications.

3 Components

3.1 Arduino Uno

The Arduino Uno is a microcontroller board used for reading sensor data and controlling outputs. It is the central component of the measurement system, providing the interface between the sensors and the computer.

3.2 HX-711

The HX-711 is a precision 24-bit analog-to-digital converter (ADC) designed specifically for weigh scales and industrial control applications. It is widely used to amplify and convert the small analog signal from a load sensor (also

known as a load cell) into a readable digital signal for microcontrollers such as the Arduino Uno.

3.2.1 Pin Configuration and Connections

The HX-711 module typically has the following pins:

- **VCC**: Power supply pin, typically connected to a 2.6V to 5.5V power source.
- **GND**: Ground pin, connected to the ground of the system.
- **DT (DOUT)**: Data output pin, connected to the digital input pin of the Arduino .
- **SCK (SCK)**: Serial clock input pin, connected to the digital output pin of the Arduino .
- **Rate**: Optional pin to set the data rate (10Hz).

3.2.2 Connecting the HX-711 to the Arduino Uno

To connect the HX-711 to the Arduino Uno, follow these steps:

- Connect the **VCC** pin of the HX-711 to the 5V pin of the Arduino.
- Connect the **GND** pin of the HX-711 to the GND pin of the Arduino.
- Connect the **DT (DOUT)** pin of the HX-711 to a digital input pin on the Arduino .
- Connect the **SCK (SCK)** pin of the HX-711 to a digital output pin on the Arduino .

3.3 Load Sensor

A load sensor, or load cell, is a transducer that converts force into an electrical signal. The electrical signal can be a change in resistance, voltage, or current depending on the type of load cell. For the parachute simulator project, strain gauge load cells are commonly used.

3.3.1 Load Sensor Configuration and Connections

A typical load cell has four wires:

- **Red (E+)**: Excitation+ or VCC, connected to the E+ pin on the HX-711.
- **Black (E-)**: Excitation- or GND, connected to the E- pin on the HX-711.
- **White (A-)**: Output- or signal-, connected to the A- pin on the HX-711.
- **Green (A+)**: Output+ or signal+, connected to the A+ pin on the HX-711.

3.3.2 Connecting the Load Sensor to the HX-711

To connect a load sensor to the HX-711 module, follow these steps:

- Connect the **Red (E+)** wire of the load sensor to the E+ pin on the HX-711.
- Connect the **Black (E-)** wire of the load sensor to the E- pin on the HX-711.
- Connect the **White (A-)** wire of the load sensor to the A- pin on the HX-711.
- Connect the **Green (A+)** wire of the load sensor to the A+ pin on the HX-711.

3.3.3 Working Principle

When a force is applied to the load cell, it deforms slightly. This deformation changes the resistance of the strain gauges in the load cell, creating a small differential voltage. The HX-711 amplifies this small voltage and converts it to a digital signal that can be read by the Arduino. The Arduino then processes this digital signal to determine the applied force.

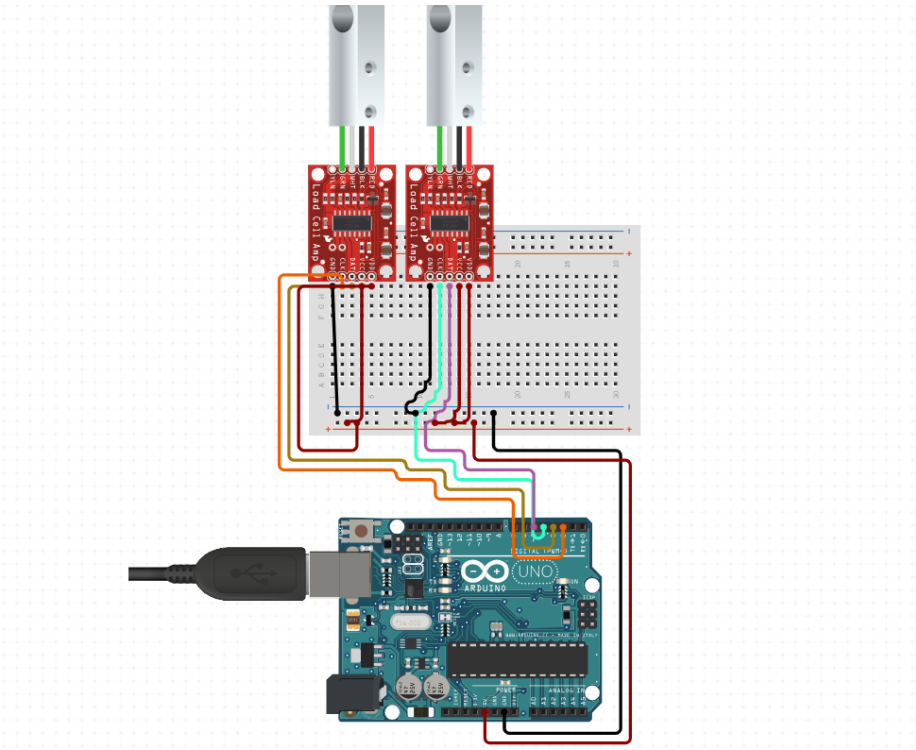

Figure 1: Connection diagram for HX-711 and load cell with Arduino Uno

By integrating the HX-711 with the load sensor and Arduino, the parachute simulator can accurately measure and display the forces applied to the handles, providing a realistic simulation experience.

4 Setup Bugs and Troubleshooting

This section will be filled with initial setup needed and all the bugs and troubleshooting steps encountered during the project. Examples of common issues include sensor calibration problems, signal noise, and software integration challenges.

4.1 HX-711 Library Setup

To facilitate the communication between the Arduino and the HX-711 ADC module, a dedicated library is used. The HX-711 library simplifies the process of reading data from the load sensor. Follow these steps to download and set up the HX-711 library in the Arduino Integrated Development Environment (IDE).

4.1.1 Downloading the HX-711 Library

The HX-711 library can be easily downloaded and installed through the Arduino Library Manager.

1. Open the Arduino IDE.
2. Navigate to **Sketch** → **Include Library** → **Manage Libraries...**.
3. In the Library Manager, type **HX711** in the search bar.
4. Locate the **HX711 by olkal** library in the search results.
5. Click the **Install** button next to the library name.

Alternatively, the library can be downloaded from the GitHub repository and installed manually:

1. Visit the HX-711 GitHub repository: `https://github.com/olkal/HX711_ADC`.
2. Download the ZIP file of the repository by clicking the **Code** button and selecting **Download ZIP**.
3. Open the Arduino IDE.
4. Navigate to **Sketch** → **Include Library** → **Add .ZIP Library...**.
5. Select the downloaded ZIP file to install the library.

4.1.2 Setting Up the HX-711 Library

Once the library is installed, it needs to be included in the Arduino sketch to be used. The following example demonstrates how to set up and use the HX-711 library in your Arduino code:

4.2 HX_711 Config - Moving average

4.2.1 What is moving average

The moving average is a statistical technique used to smooth out short-term fluctuations and highlight longer-term trends or cycles in data. It's commonly applied in time series analysis
Impact of Moving Average on HX711 Measurements:

Noise Reduction:
The HX711 can produce noisy readings due to electronic noise, environmental factors, and mechanical vibrations. Applying a moving average helps smooth these fluctuations, providing a more stable and accurate reading.

Trend Detection: It helps in identifying trends or consistent changes in weight, which can be crucial for applications requiring precise and stable measurements.

Reduced Sensitivity to Outliers:
A moving average can mitigate the effect of transient spikes or drops in readings, which might otherwise lead to incorrect interpretations of the data.

Latency:
While smoothing the data, moving averages introduce a lag or delay in the response to changes. This can be a trade-off between noise reduction and real-time responsiveness.

4.2.2 How to change the moving average

More setup options are available in the 'HX_711 config file' which can be found in the Arduino libraries folder inside the 'HX711_ADC'. the library can be found by searching the folder HX711_ADC (usually sits at Documents -> Arduino -> libraries). Inside the ADC folder , go into the 'src' folder, then , open the config file using text editor. inside the config file you will find a row that begins with 'define SAMPLES' , under this property, you can specify the moving average yo want from the values 2, 4 ,8, 16, 32, 64, 128. Note that there are additional configurations I did not explore.

4.3 Arduino Connection

If Arduino connection was not established properly do the following:
1. Open Device manager under Ports(COM & LPT).
2. Find the Arduino device. If Arduino shows under COM-XX , the connection has been established properly.

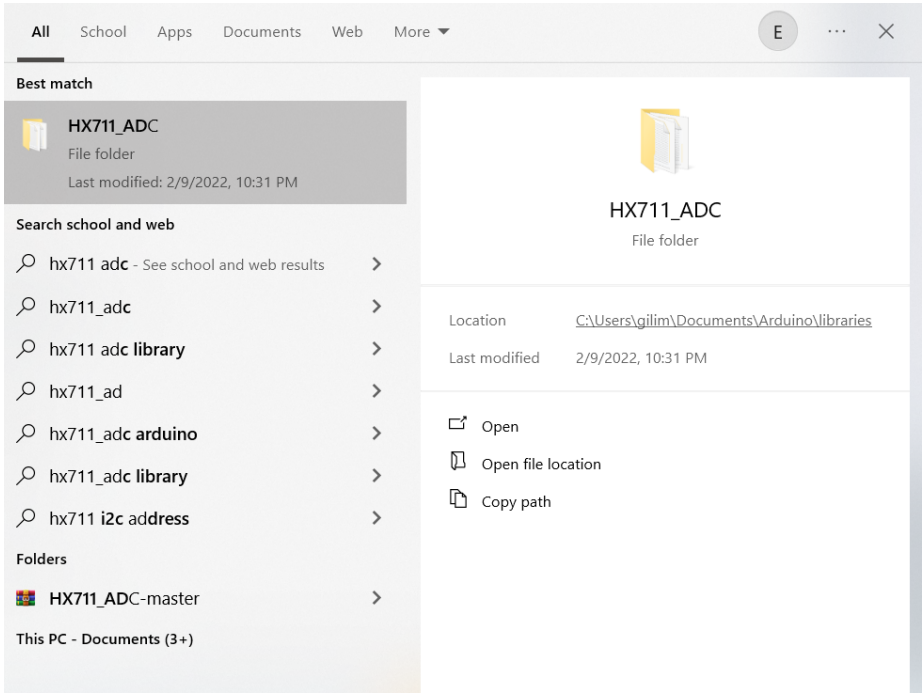

Figure 2: HX711_ADC folder

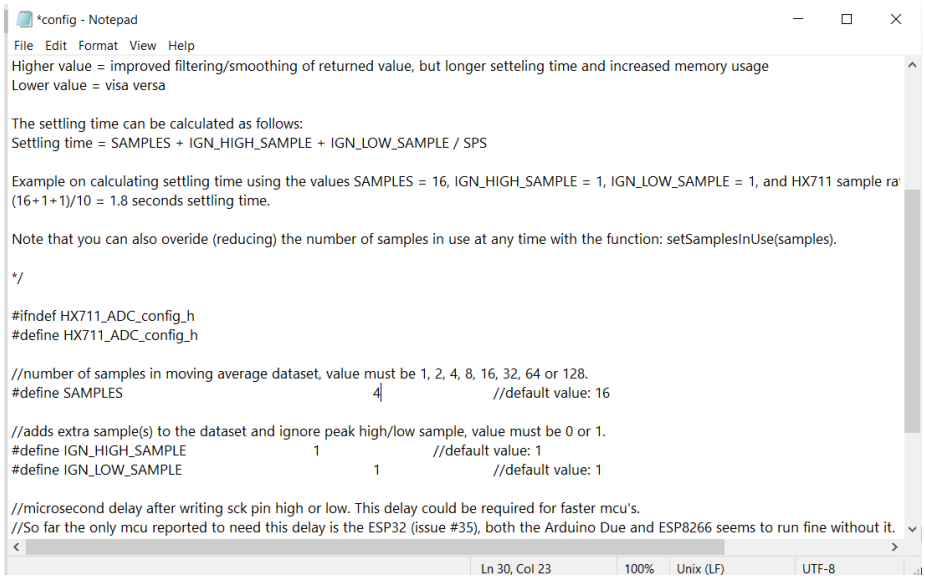

Figure 3: HX711_ADC config file

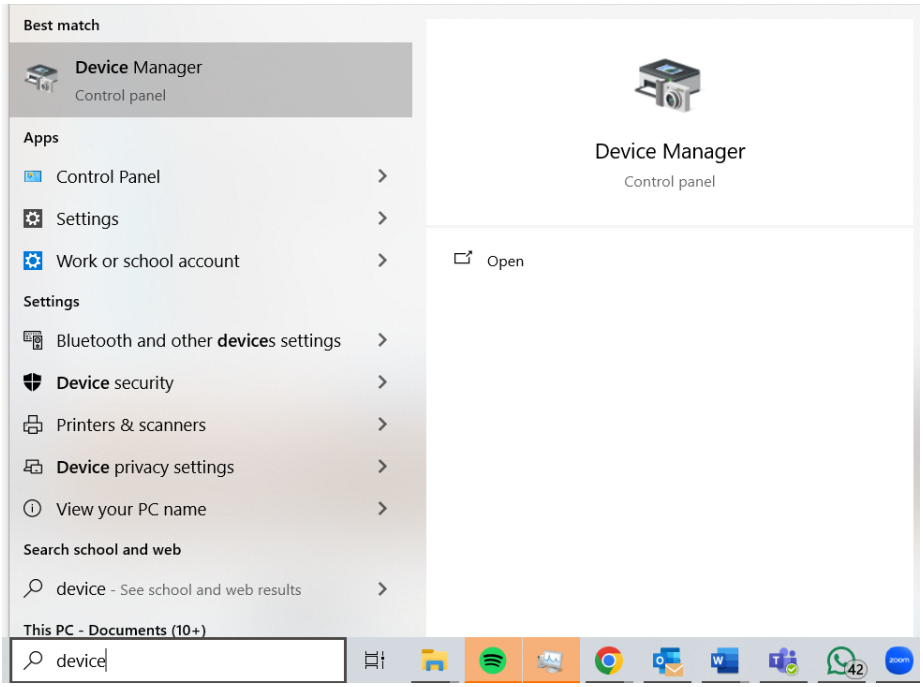

Figure 4: Device manager bug (1)

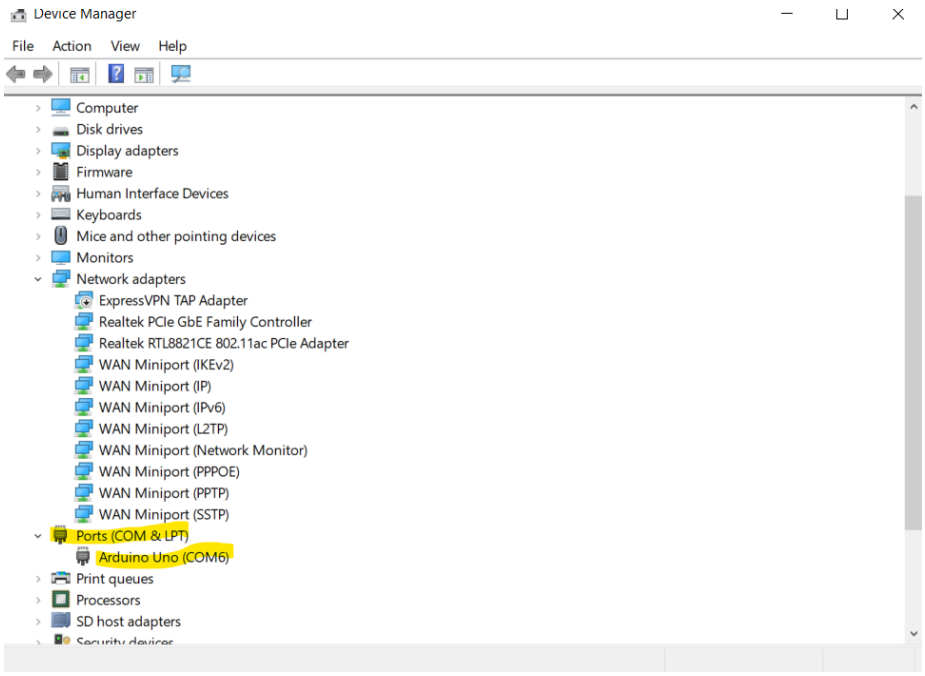

Figure 5: Device manager bug (2)

4.4 Setup Arduino IDE

Setup of Arduino IDE - the following steps should be preformed if any changes are needed for the core arduino code, note that if you are not familiar with the arduino language and just using the simulator for yourself, you can skip this part and go straight to the MATLAB GUI section.

4.4.1 Setup arduino

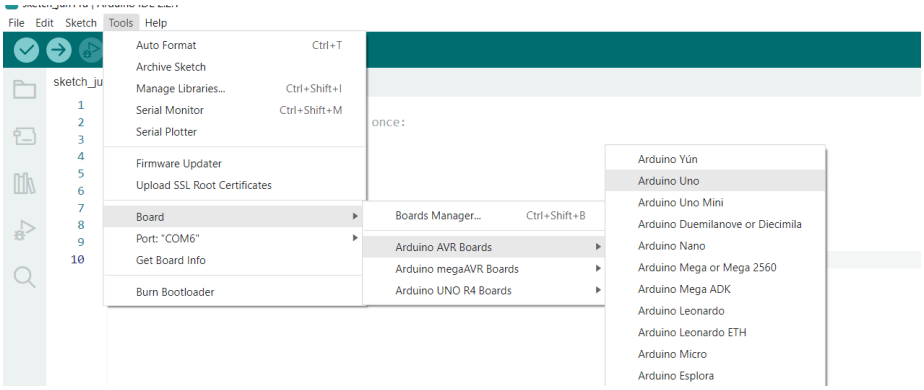

Figure 6: Arduino IDE - Device selection

4.4.2 Setup COM serial port

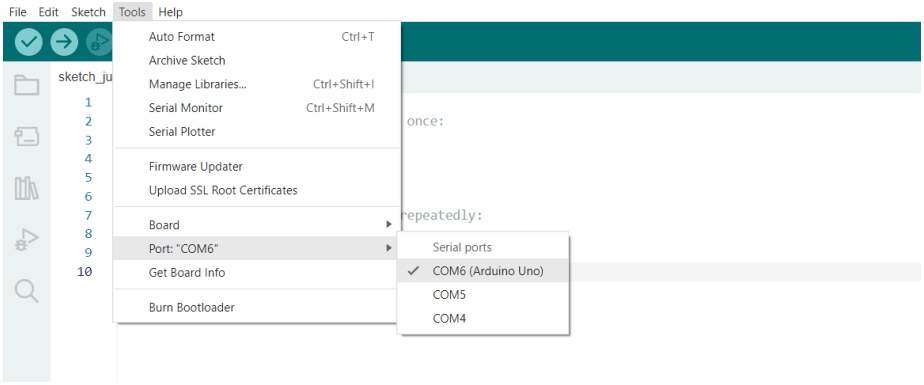

Figure 7: Arduino IDE - COM selection

4.4.3 Serial Window and Baud Rate

To manage the serial window preform the following (numbers in figure below) (1) - serial port icon on the right left will open the window at the bottom(2), to read correctly set baud rate(3) to 57600

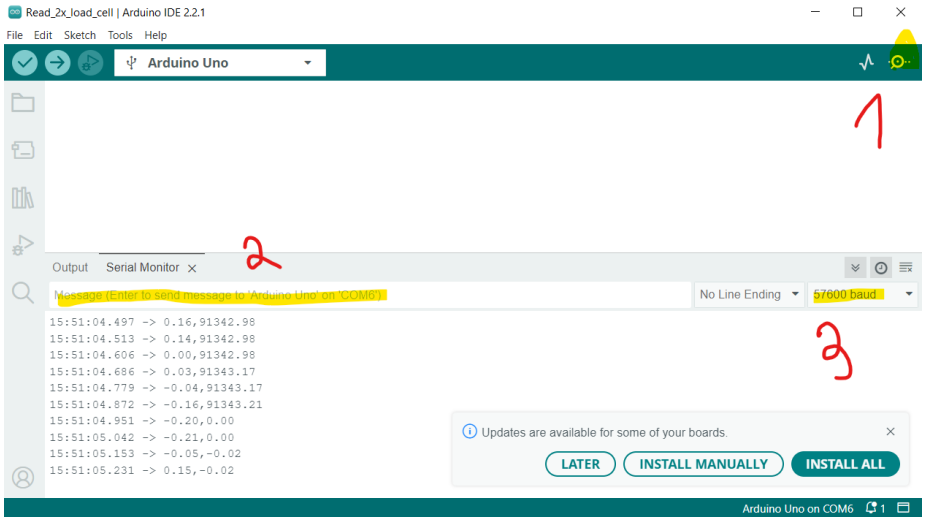

Figure 8: Arduino IDE

5 Arduino Code

The Arduino code for the parachute simulator is divided into two main parts: calibration and data collection. The calibration part is executed once to calibrate the system, while the read part runs continuously to collect data during operation.

5.1 Calibration code

The calibration process is essential to ensure accurate force measurements from the load sensors. During calibration, known weights are applied to the sensors to determine the calibration factor, which converts the raw sensor readings into meaningful force values. Arduino code snippet that shows the calibration procedure is attached under 'Calibration.ino'
If recalibration is needed, the following steps should be followed:
1. Open the Arduino IDE. 2. Load the 'Calibration.ino' snippit. 3. Connect the Arduino to your computer. 4. Upload the sketch to the arduino. 5. Open the serial port on the right hand side of the IDE window 6.follow the serial terminal instruction.

5.2 Read And Data Collection Code

Once the system is calibrated, the read function continuously collects data from the load sensors. This data represents the forces applied to the handles and is crucial for simulating the parachute controls. The following code snippet shows how data is collected and sent to a computer for further processing:

6 MATLAB Code

The MATLAB code for the parachute simulator is developed using the App Designer tool, which provides a user-friendly graphical interface for real-time data monitoring and control. The application includes several interactive buttons and a UI to facilitate the measurement process. The primary components of the MATLAB app are:

6.1 COM Input

An input box in which the COM serial number for the current connection should be defined.
It is importent to note that you sholud only include the integer of the COM and not the whole COMXX .

6.2 Start Button

The Start button initiates the data reading procedure. When pressed, the app begins collecting force data from the Arduino and displays it in real-time on the UI. This allows users to observe the forces applied to the handles as they interact with the simulator.

6.3 Tare Button

The Tare button is used to define a new baseline for the sensors during the reading process. Pressing this button resets the current readings to zero, accounting for any drift or offset in the sensor measurements. This feature ensures that the force readings remain accurate and relevant throughout the simulation.

6.4 Save Button

The Save button allows users to save the collected measurements to a file. This functionality is essential for data analysis and record-keeping, enabling users to review and analyze the force data post-simulation. The data is typically saved in a format suitable for further processing or visualization.

6.5 Stop Button

The Stop button halts the data reading process. This button is crucial for controlling the simulation, allowing users to end the data collection when desired. Once stopped, the app can either reset for a new simulation session or shut down, depending on the user's needs.

6.6 Real-time Data Display

The UI within the MATLAB app provides real-time visualization of the force readings. As data is collected from the sensors, it is immediately displayed

on the screen, offering an intuitive and immediate understanding of the forces being applied. This real-time feedback is invaluable for both educational and experimental purposes.
A screenshot of the GUI is provided below:

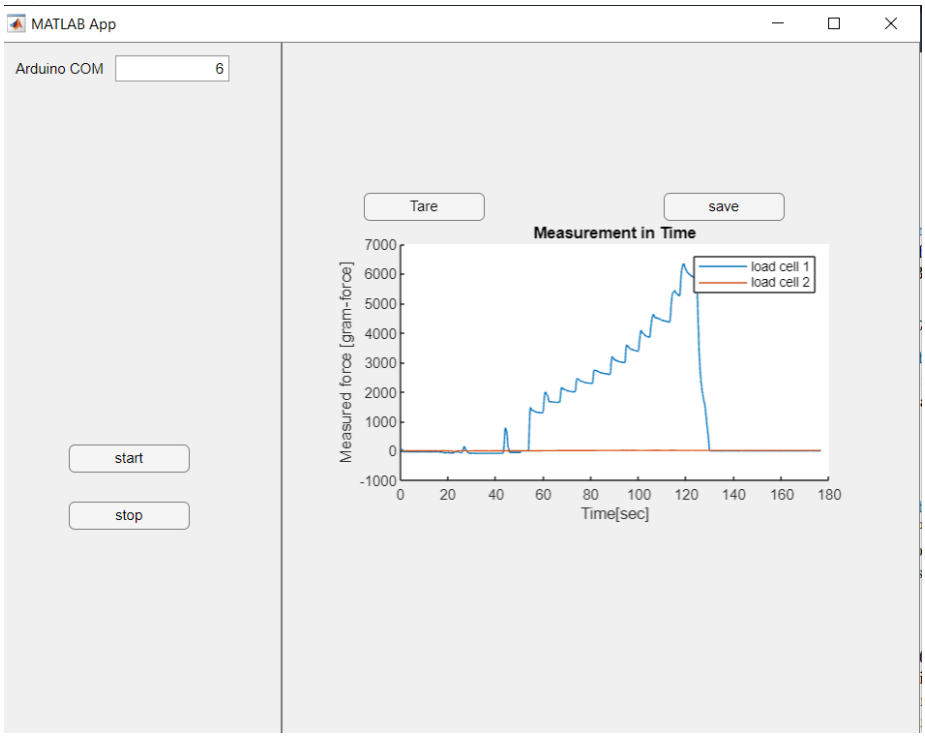

Figure 9: MATLAB app GUI

7 Conclusion

The parachute simulator project successfully integrates hardware and software to create a realistic and educational tool. The system provides practical experience and deepens understanding of parachute dynamics and control mechanisms.

References

[1] Arduino Uno Documentation. Retrieved from `https://www.arduino.cc/en/Main/arduinoBoardUno`

[2] HX-711 Datasheet. Retrieved from `https://github.com/sparkfun/HX711-Load-Cell-Amplifier/tree/V_1.1`

[3] Load Sensor Overview. Retrieved from https://www.gotronic.fr/pj-460.pdf

[4] HX-711 library. Retrieved from https://github.com/olkal/HX711_ADC