Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Software for posturometric platform

Josef Vágner

Supervisor: Doc. Ing. Tomáš Haniš, Ph.D.
Field of study: Cybernetics and Robotics
May 2023

Acknowledgements

I would like to thank my supervisor Ing. Tomáš Haniš, Ph. D., for his advice and support while working on my bachelor project.

I would especially like to thank Mr. Laušman and Mr. Šťastný for their cooperation in this posturometric platform project. I also thank Ing. Jaroslav Bušek, Ph.D. for the advice he gave me and for constantly pushing me to improve my work. I also thank my high school, SPŠ na Proseku, for the space provided for the development of this project.

Finally, I would like to thank my family and friends for their support throughout my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, ...

Abstract

This thesis focuses on the design of software for a controller of a posturometric platform designed for a treadmill, which is used for posturographic analysis of the patient and his gait. The thesis proposes suitable algorithms and procedures for processing data from strain gauge sensors placed in the platform. The processed values are used to calculate the center of pressure trajectory and the weight applied to the platform. These are then analyzed to obtain parameters about the patient's gait and stability. The thesis further covers the design of the graphical user interface that is used to display the obtained values.

The result is an application that records, analyzes and displays the data from the posturometric platform.

Keywords: posturometric platform, posturography, treadmill, center of pressure, peak detection, filtartion, graphical user interface

Supervisor: Doc. Ing. Tomáš Haniš, Ph.D.
Katedra řídicí techniky,
Resslova 9,
Praha

Abstrakt

Tato práce se zabývá návrhem softwaru pro řidící jednotku posturometrické platformy určené pro běhací pás, která slouží k posturografické analýze pacienta a jeho chůze. Práce navrhuje vhodné algoritmy a procedury pro zpracování dat z tenzometrických senzorů umístěných v platformě. Zpracované hodnoty dále slouží k výpočtu trajektorie bodu zatížení a váhy působící na platformu. Ty jsou následně analyzovány k získání parametrů o pacientově chůzi a stabilitě. Práce se dále zabývá návrhem grafického uživatelkého rozhraní, které slouží k zobrazení získaných hodnot.

Výstupem je aplikace, která zaznamenává, analyzuje a zobrazuje data z posturometrické platformy.

Klíčová slova: posturometrická platforma, posturografie, běhací pás, bod zatížení, detekce vrcholů, filtrace, grafické uživatelské rozhraní

Překlad názvu: Software pro posturometrickou platformu

Contents

Figures

Tables

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Vágner Josef** Personal ID number: **499312**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Software for posturometric platform

Bachelor’s thesis title in Czech:

Software pro posturometrickou platformu

Guidelines:

The injuries of the musculoskeletal system and especially of the lower limbs is critical issue in sociate, with high impact on life quality and productivity. Early indication and physiotherapy process monitoring dramatically reduce recovery time. The primary objective of this thesis is to prepare technical equipment for posturometric analysis.
1. Get familiar with posturometric platforms and techniques.
2. Prepare methodology, test scenarios and evaluation process.
3. Implement framework with data acquisition, processing and evaluation.
4. Test execution and evaluation

Bibliography / sources:

[1] Helštýnová, Barbara, Analysis of Posturometric Data, Ostrava, 2012. VSB – Technical University of Ostrava.
[2] Bruštíková Jolana, Use of Tenzometric Platform in Ambulatory Therapy, CTU in Prague, Faculty of biomedical engineering, 2016
[3] Tichý, Lukáš, The effect of a six-week physiotherapeutic intervention in child patients with clubfoot evaluated with a change in ROM of ankle dorsiflexion and the quality of gait measured by 2D analysis, Prague, 2021. Charles Univerisity, Faculty of Physical Education and Sport

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Tomáš Haniš, Ph.D. Department of Control Engineering FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **31.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until:
by the end of summer semester 2023/2024

doc. Ing. Tomáš Haniš, Ph.D. Supervisor’s signature	prof. Ing. Michael Šebek, DrSc. Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student’s signature

List of notations

CoP Center of Pressure.

CoG Center of Gravity.

CoF Center of Force.

GUI Graphic User Interface.

OSI Open Systems Interconnection.

SPI Serial Peripheral Interface.

isoSPI Isolated Serial Peripheral Interface.

CS Chip Select.

CAN Controller Area Network.

IIR Infinite Impulse Response.

FIR Finite Impulse Response.

Chapter 1

Introduction

1.1 Motivation

Injuries of lower limbs are a very complex topic in medicine. After a patient goes through treatment for an injury, for example, a broken leg, it is almost always necessary to begin a rehabilitation process that will help eliminate long-term consequences. The problem is that rehabilitation is time-consuming and expensive. Sometimes, there are not enough rehabilitation workers for all patients [1], especially if the patient's injury is not complicated and needs regular training with correct feedback.

The doctor may ordinate a treadmill workout when the patient needs walking or running exercise. Correct moves and techniques are critical factors in this form of rehabilitation and are often supervised by a specialist who guides the patient toward better results. This exercise can also require specialized equipment, such as treadmills with sensors that measure patient movement and stability. These devices can be costly and if the rehabilitation center is already equipped with standard treadmills, they would need to replace the standard treadmill or buy a new one equipped with sensors.

The solution would be a retrofit treadmill platform equipped with sensors (e.g. posturometric platform) that can provide helpful feedback to the patient and inform the specialist about the patient's progress. This means that conventional treadmills could be upgraded with the sensor platform, which could help expand this type of rehabilitation.

1.2 Thesis goal

This thesis aims to develop software for a posturometric platform. More specifically, develop an application for the platform's control device that

displays data from the platform. This includes proposals of the methods and algorithms needed for calculating parameters that are crucial for posturometric platforms.

1.3 Outline of thesis

The first part of the thesis aims to clarify how posturometric platforms work and the architecture of the posturometric platform project for which the results of this work will be used. After that, I will describe the process, methods, and algorithms which I will use when implementing the application. The last part aims to implement the application and test how the proposed algorithms and methods perform on a real prototype platform.

Chapter 2

Posturometric analysis

Before starting, let us find out more about the topic of posturography. It is necessary to explore what this field of study involves, the existing solutions are and their use in practice. This information will then provide the foundation for the posturography platform project.

2.1 Posturography

Posturography is a diagnostic method to determine the patient's ability to maintain balance. It is a noninvasive method based on measuring the patient's Center of Gravity (CoG) under static or dynamic conditions [2]. Because measuring CoG could be challenging, general posturometric platforms instead measure Center of Pressure (CoP) as an approximation of CoG [3]. This method is used to detect, monitor and treat stability and postural disorders. It can be used for short-term monitoring within a single exercise and long-term monitoring with statistical evaluation.

2.1.1 Static posturography

Static posturography is based on measuring the patient's CoP when the patient is standing with different alignments of the feet and with closed or opened eyes. For example, we can observe the difference in patient stability when his feet are vertical side by side and when his/her feet are turned 45 degrees away with his toes apart [4]. Sensor data can be plotted for easier analysis. There are three main types of graphs: statokinesiogram, stabilogram, and harmonic analysis [5]. Examples are shown in Figure 2.1.

The statokinesiogram shows the patient's CoP in the xy plane, where the coordinate x shows movement from left to right and the coordinate y shows movement from front to back. The stabilogram shows the movement of CoP

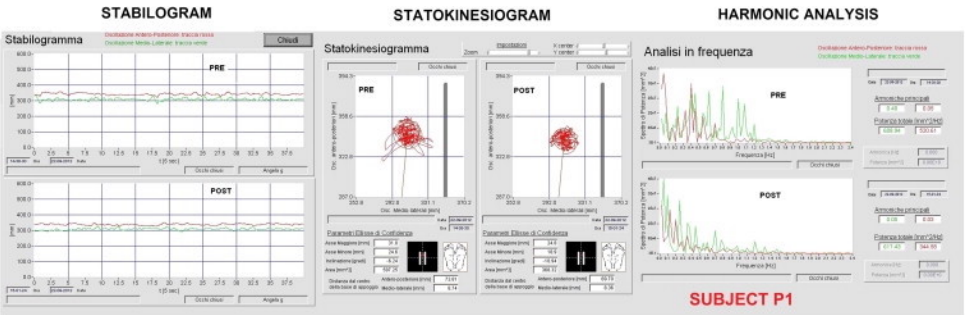

Figure 2.1: Example of static posturography plots [5]

as a function of time [5]. Both methods can help detect abnormalities and analyze the patient's health condition. Harmonic analysis shows a frequency decomposition of the posturography data.

2.1.2 Dynamic posturography

Dynamic posturography uses techniques similar to static posturography, but platforms can be motorized. During this examination, the patient is standing on a moving platform which can be supplemented with equipment for visual and audio stimulation. This arrangement allows doctors to measure the patient's stability reactions to changes in their environment [2]. Some of these movements or visual changes are shown in Figure 2.2.

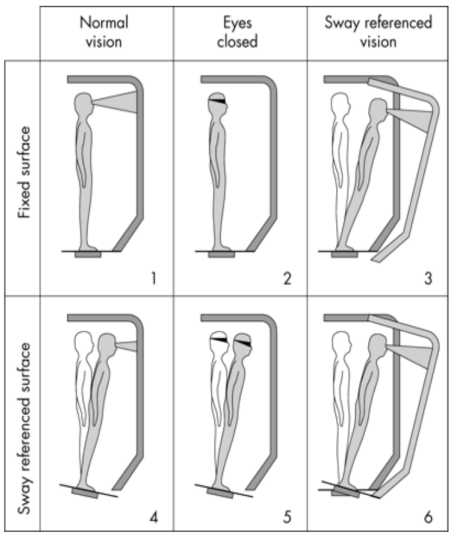

Figure 2.2: Example of dynamic posturography stages [6]

As an extended part of dynamic posturography, we can consider using of treadmills as a moving platform. Here, we can examine the patient's ability to maintain stability and coordination while walking or running.

2.2 Types of posturographic platforms

In the present day, there are a variety of platforms used in rehabilitation centers, hospitals, and clinics. We can also find some for home use. These platforms can come in various sizes, different functionality, and different price ranges.

2.2.1 Static boards

An example of a static board for rehabilitation at home or in the clinic can be the Nintendo Wii Balance Board (Figure 2.3) [7]. This board was initially meant to be a game accessory for the Nintendo Wii so that players could control a game by balancing on it. It was designed with four strain gauge sensors and wireless communication. Subsequently, it was discovered that it could be used to measure CoP in rehabilitation applications. Due to its good accuracy, small size, and low price, it became pretty popular [2].

Figure 2.3: Nintendo Wii Balance Board used for rehabilitation [7]

2.2.2 Dynamic boards

Smart Balance Master is a stabilometric board aimed at dynamic posturography analysis. Today, it is mainly used as a rehabilitation device at first created for NASA to study the effects of space travel on the human body [8]. This board has two separate sensor plates, one for each leg. It can perform rotational and transitional movements and simulate a changing environment with a screen in front of a patient [2].

Figure 2.4: Smart Balance Master [8]

2.2.3 Treadmills

Treadmills that use posturographic analysis methods used for rehabilitation are treadmills equipped with sensors that measure patient stability and performance during exercise. These treadmills greatly benefit lower limb exercises and cardiovascular system [9]. An example of a posturometric treadmill is shown in Figure 2.5.

Figure 2.5: Gait&Balance treadmill [10]

Some patients may benefit from exercising in water, where the aquatic environment provides the necessary support for seriously injured patients. Water also slows all movements, allowing patients to concentrate more on maintaining proper movements. Studies have shown an improvement in

the cardiovascular system, gait, stability, and even pain reduction when using underwater treadmills [11]. The underwater treadmills can be installed directly in a pool or submerged in a water tank (Figure 2.6).

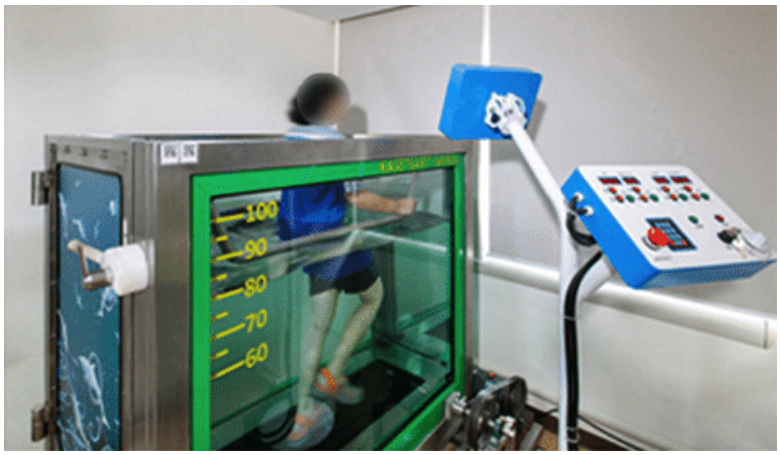

Figure 2.6: Underwater treadmill with tank [11]

Chapter 3

System architecture

The goal is to create a retrofit alternative to the already available posturometric platforms that could help expand this form of rehabilitation. This means that a whole treadmill does not need to be created but only the platform with sensors. This platform could function as an upgrade to traditional treadmills.

This thesis is part of the posturometric platform development, in which I develop software for a control device. Other parts are in the hands of my colleagues, Adam Šťastný and David Laušman. Mr. Šťastný is responsible for the sensor equipment, and Mr. Laušman develops the mentioned control device and handles long-term data analysis.

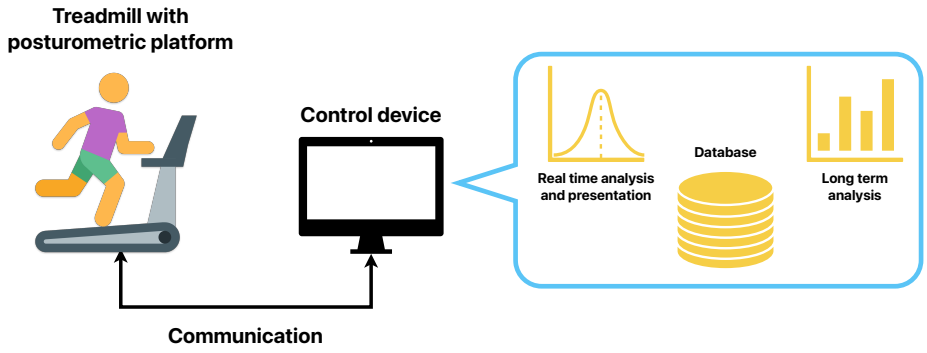

Figure 3.1: System architecture

In this chapter, I will describe the project architecture and the decisions that our team made in common parts of the project that have an impact on my work. What type of sensors do we use on the platform and what type of communication do we use. Due to the close collaboration between me and Mr. Laušman on the software itself, where he is responsible for the long-term data analysis, I will also describe what programming language and frameworks we decided to use. The product architecture is summarized by a diagram in Figure 3.1.

3.1 Type of sensor equipment

The first thing we need to decide is which sensors to use for the posturomentric platform. With sensors, we need to be able to detect the patient's movement on the platform and also monitor the weight that loads the platform. We considered methods that include capacitive panels, resistive panels, and a matrix of strain gauges. Capacitive and resistive panels are sensory devices designed to detect the contact position on the panel and are used mainly in touch screens [12]. Strain gauges are sensors that measure the load to which they are subjected.

3.1.1 Projected capacitive panels

Projected capacitive (p-cap) panels are based on measuring the changes in the electric field caused by a conductive object (like a finger) touching the panel [12]. The panel is composed of arranged pairs of electrodes. These electrodes generate an electric field between them, and when a finger is close to the pair, the electric field changes [12]. This change triggers the controller, which determines the position of the contact. Since every pair of electrodes functions as an individual capacitive touch sensor, the panel has no problem handling multi-touch events. An illustration of how the p-cap panel works is shown in Figure 3.2.

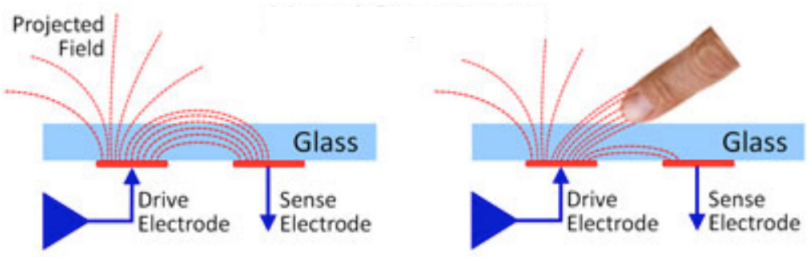

Figure 3.2: P-cap panels illustration [12]

Unfortunately, these panels require contact with a conducive object in order to work. That is why most smartphones cannot be used with gloves, usually made from nonconductive materials. That means that if we want to use these panels on the posturometric platform, we would need to modify the belt so that it can work with the p-cap panel. That can also cause unwanted touches and other problems. Another problem is that the p-cap panels cannot measure weight and may not survive a high impact on the running patient.

3.1.2 Resistive panels

Resistive panels are based on two separate conductive plates, where the upper plate is usually flexible, so when the user presses the panel, the upper plate is pushed to the lower plate and forms a connection that can be measured to determine the position of the press (Figure 3.3) [12]. Because this type of panel depends only on a force that pushes two plates together, there is no need for a conductive object, so the panel works with any non-sharp object [12].

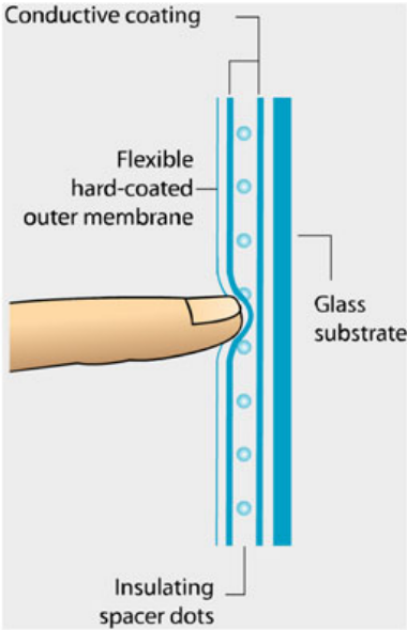

Figure 3.3: Arrangement of conductive plates in resistive plates [12]

Older resistive panels do not support multitouch because the contact position was calculated from voltage dividers that form when the plates touch each other. The second contact would change the voltage dividers, and the contact position would not be calculated correctly. The newer resistive panels consist of small individual segments arranged in a grid and therefore support multitouch. Each segment works individually, so the calculation is not corrupted, when the contact occurs in two separate segments [12].

Resistive panels seem like a great option for our application because they are cheap and can detect any non-sharp object touching them, so materials between the treadmill platform and the patient's feet will not be a problem. The main disadvantages of resistive panels are their low durability and the inability to measure weight on the platform [12].

3.1.3 Strain gauge

A strain gauge sensor is a specially modified resistor that is basically a long flat wire attached to a base material, as shown in Figure 3.4. This arrangement results in a significant change in the length of the wire when the support material bends. A change in the length of the wire changes its resistance, which can be measured and processed to a strain value applied to the base material [13]. Strain gauge sensors can be used in different configurations to measure strains in different directions.

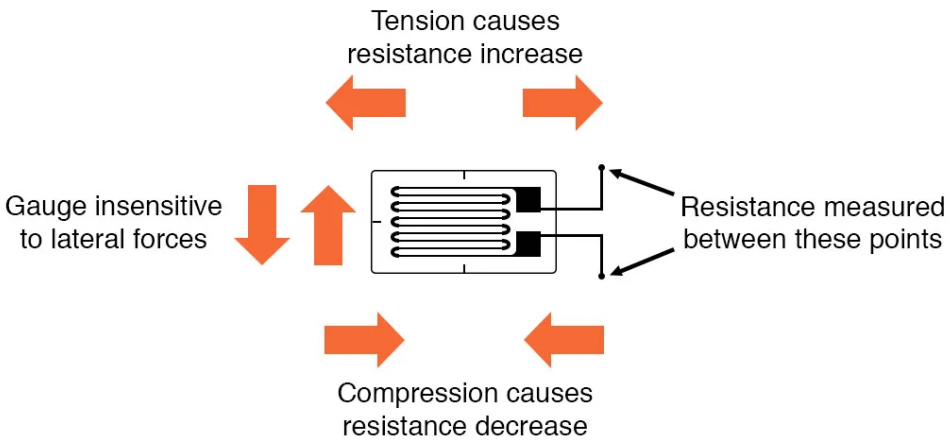

Figure 3.4: Strain gauge illustration [14]

The typical configuration of the strain gauges is a full Wheatstone bridge. This arrangement allows for a more precise measurement of the strain applied to the support material. The measurement is based on the measurement of the output voltage V_{out} when the known voltage V_{in} is applied.

3.1.4 Selected type of sensors

Using strain gauge sensors to measure patient's position on the platform may not be as accurate as the panels mentioned above. However, these methods do not provide information about the weight distribution on the platform. In addition, touch-sensitive panels are inconvenient for large panels because they must be custom-made and more durable. There is also a need for the contact of the conductive object with the p-cap panels, which would be difficult to maintain with a conventional treadmill. All these disadvantages are not present with strain gauges. Strain gauges are durable and can provide information on the weight distribution applied to the platform. For these reasons, we decided to use strain gauges.

3.2 Communication type

This part focuses on choosing a communication type that will be used for data transfer between the posturometric platform and the control unit. In the early stage of this project, we had to decide between wireless and wired communication. However, because our platform does not have a battery and in the future, we plan to make an underwater version of our platform, wireless communication is a dead-end for us. Traditional wireless communication does not perform well in a water environment, so we would need to keep our communication module away from the water. Since we already need a power supply cable between the control device and the platform, it is not a problem to use wire communication.

Because the platform is located near a treadmill motor that will generate high noise, we must find a suitable wired type of communication resistant to interference. The communication distance might be around tens of meters. For these reasons, we will only consider types of communication resistant to interference and with a longer communication range. The standard for these requirements is the type of communication that uses differential signalling.

3.2.1 Differential signals

Unlike standard single-ended signalling, differential signalling uses two complementary voltage signals to encode one information signal. Both voltage signals have the same amplitude but different polarities over the common-mode voltage. The receiver gets information by simply making a difference in the pair of cables. These "balanced" voltage levels mean that differential signalling is more resistant to interference because interference affects both voltage signals equally, so the resulting difference will not change [15].

3.2.2 RS485

TIA/EIA-485-A, which is known as RS485, is one of the basic communication standards that uses differential communication. The TIA-485 standard specifies the receiver and transmitter parameters on the multipoint bus. It is not a protocol, so communication handling is not specified. At the bare minimum, it requires one pair of twisted cables that connect two or more devices arranged in chains on the bus and the termination at both ends of the bus [16]. In terms of the Open Systems Interconnection (OSI) reference model, TIA485 specifies only the physical layer and is used by protocols of higher layers [16].

3.2.3 Isolated Serial Peripheral Interface

Serial Peripheral Interface (SPI) is a common type of communication between microcontrollers and peripheral devices, such as sensors or output devices. SPI is a full-duplex master-slave communication interface that uses separate Chip Select (CS) cables instead of slave addresses. However, SPI is not meant for long-distance communication and is not resistant to interference, so the Isolated Serial Peripheral Interface (isoSPI) was invented. IsoSPI is an upgraded version of SPI with an additional transmitter that converts SPI signals to differential signals. It was invented as a low-cost and low-level alternative to the Controller Area Network (CAN) bus for electronic car battery packs, where there is no need for a high-level bus interface but instead simple master-slave communication between the battery controller and the battery segments [17]. Figure 3.5 illustrates the isoSPI connection between devices.

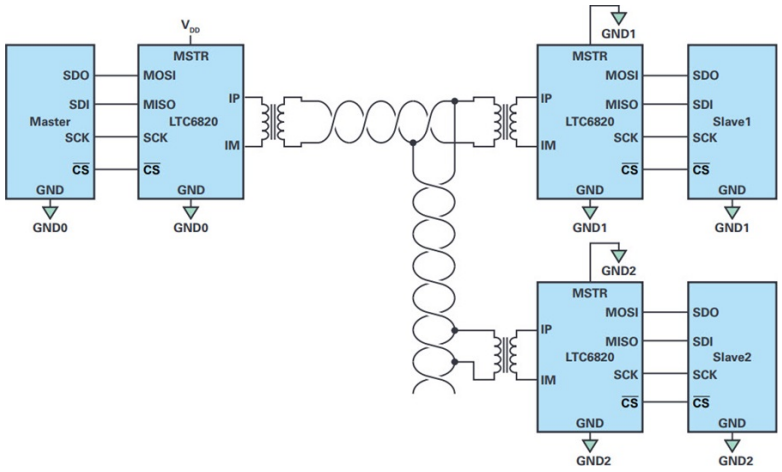

Figure 3.5: IsoSPI bus illustration [18]

3.2.4 Controller Area Network

CAN bus is a message-based protocol designed to allow devices (nodes) to communicate with each other without the need for a master node (multi-master bus). It was originally intended for the automotive industry, but it can be easily used in different applications [19]. Every massage on the bus has its arbitration id, which is used to prioritize massages. When two or more nodes try to send messages at the same time, the message with the highest priority is sent, and all other messages are postponed until the bus is free. CAN bus works in the broadcast mode so that every node on the bus receives all messages. All messages are represented as CAN data frames with the necessary data and parameters, such as the arbitration id and the time stamp [20]. Figure 3.6 illustrates the structure of the base CAN data frame.

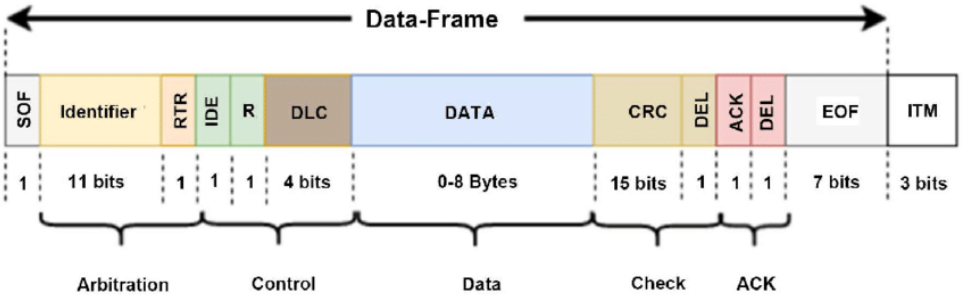

Figure 3.6: Base CAN data frame structure [20]

CAN data frames exist in base or extended versions. The extended version allows for more messages due to a larger arbitration id. The number of devices is not limited by the bus, but the number of different messages is limited by the size of the arbitration id [16]. For better noise resistance, CAN devices use a transmitter that transfers default CAN RX and TX signals to differential signals CAN HIGH and LOW. This approach allows high interference resistance, such as RS485, and only two wires are needed for communication [19]. Summarizing the benefits of the CAN bus [16]:

- message identification and real-time prioritization
- system wide data consistency
- multi master bus
- error detection
- automatic resend of unsend messages when bus is free
- resistant to interference
- easily expandable

These benefits result in a significant advantage over the RS485 standard and isoSPI.

3.2.5 Selected communication interface

All of mentioned communication interfaces could be used for the platform, and we decided to use the CAN bus. The main advantage is that the CAN bus is a communication protocol on its own, so we CAN use all of its features. It is also a well-known communication interface that means easy implementation and maintenance. It is also good to say that in our project we will not use any of mentioned communication interfaces on their speed limits. Another benefit is that the CAN bus is included in many available microcontrollers, making implementation even easier.

3.3 Software development framework

The software for the posturometric platform needs to have a Graphic User Interface (GUI) that would present data to users and a back-end part that would analyze the data from the sensors. The software is also divided into two parts, real-time and long-term data analysis, which are developed by me and Mr. Laušman, respectively. That sets a criterion for the programming language and framework that we choose. We need a programming language that meets the following parameters:

- is suitable for data analysis,
- allows rapid GUI application development,
- support CAN communication protocols.

We consider Python and C since these are programming languages that we are familiar with. C is much faster than Python but is not a good choice for the rapid development of GUI applications and data analysis. On the other hand, Python is a general use programming language widely used for data science, back-end development, AI and more. Python is an object-oriented programming language and has many libraries that extend his functionality. Both Python and C support CAN communication. That leads us to pick Python as the programming language for the control unit.

The most widely used Python frameworks for programming GUI applications include PySimpleGui, PyQt, and Flask. All offer similar features and have different pros and cons. We choose to use PyQt because, unlike others, PyQt has a very optimized module for working with graphs, which is crucial when displaying real-time data. PyQt library is built on a very popular Qt framework that runs on C++. That means that PyQt has very good overall performance and is cross-platform like Qt. Another benefit is that PyQt supports CSS-like style sheets, allowing high customization of default elements.

Chapter 4

Posturometric data analysis

In this chapter, I will describe the process of analyzing the data measured by the posturometric platform. To begin with, it is necessary to determine what values the analysis should focus on. The basic structure of the process is shown in Figure 4.1.

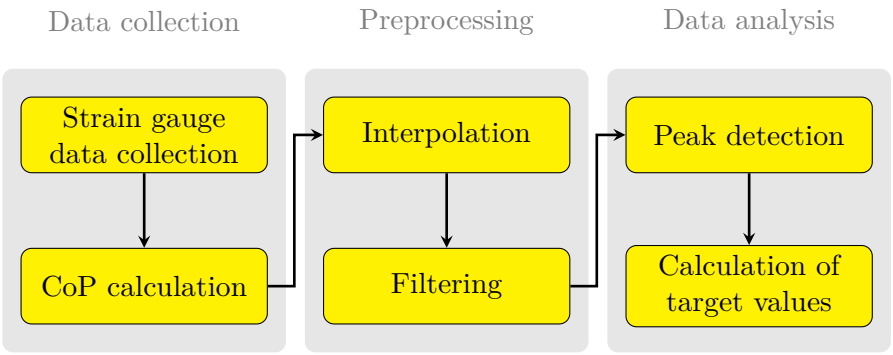

Figure 4.1: Posturometric data analysis diagram

All methods will be tested on a prototype platform from a classic treadmill, in which Mr. Šťastný has installed strain gauge sensors. Sensors are installed between the original platform's mounts and the platform itself. This arrangement resulted in using six strain gauge sensors, as shown in Figure 4.2.

4.1 Target values

As a source of information on the types of data that are useful when using the posturometric platform, I can rely on existing devices, covered in Chapter 2. Since the product is not aimed at any specific area of posturography, I decided to adapt the most common values about patient movement. The minimum that a typical posturometric platform or treadmill can do is measure the

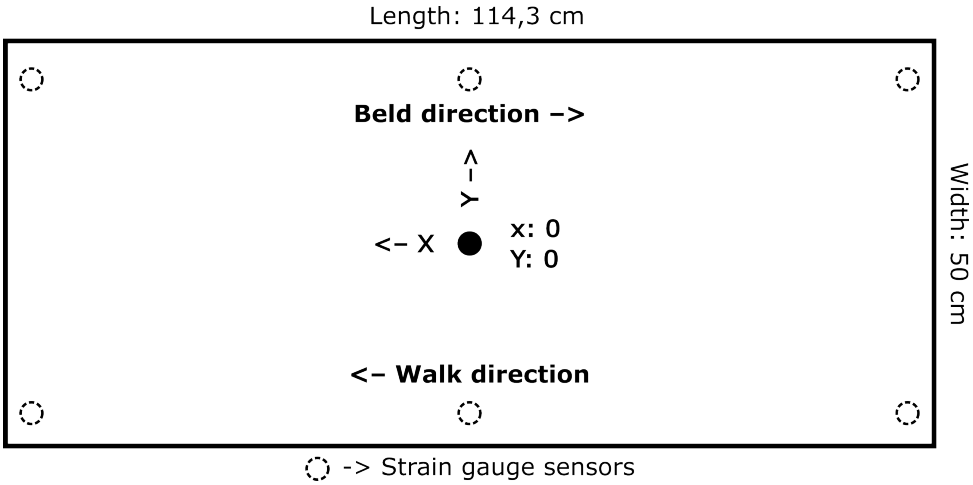

Figure 4.2: Prototype platform diagram

CoP trajectory and, from this determine the number of steps, the length and width of the steps, the patient's speed, and the walked distance. In the case of static posturometric platforms, they also measure the CoP trajectory and use it to determine patient stability.

4.2 Center of Pressure calculation

CoP calculation relies on values measured by the strain gauge sensor matrix placed under the platform. By default, the value measured by the strain gauge sensor is a voltage signal proportional to the strain applied on the sensor. Suppose that we have N strain gauge sensors, where the i-th sensor has a position in Cartesian coordinates x_i and y_i and measures the voltage U_i. The individual components of the CoP position are

$$\mathrm{CoP}_x = \frac{\sum_{i=1}^{N} U_i \cdot x_i}{U}, \tag{4.1}$$

$$\mathrm{CoP}_y = \frac{\sum_{i=1}^{N} U_i \cdot y_i}{U}, \tag{4.2}$$

where U is $\sum_{i=1}^{N} U_i$ [3]. In addition, calibration can be performed on strain gauge sensors to calculate the weight

$$m_i = (c_m)_i \cdot U_i, \tag{4.3}$$

that puts strain on the i-th sensor, where $(c_m)_i$ is the weight calibration constant of the i-th sensor. Equations 4.1 and 4.2 can be modified as

$$\mathrm{CoP}_x = \frac{\sum_{i=1}^{N} m_i \cdot x_i}{m}, \tag{4.4}$$

$$\mathrm{CoP}_y = \frac{\sum_{i=1}^{N} m_i \cdot y_i}{m}, \tag{4.5}$$

where m is $\sum_{i=1}^{N} m_i$. Unfortunately, these equations do not account for measurement inaccuracies and the possible inhomogeneity of the prototype platform. Therefore, calibration is necessary to reduce the effects of possible inaccuracies.

A complete platform calibration can be performed by placing a known weight on a specific position on the platform and recording what the sensors measure. These values can be substituted into equations 4.3, 4.4 and 4.5, leaving only the constants x, y and c_m unknown. After repeating this process multiple times for different positions and weights, the least squares fitting algorithm can be used to find unknown constants.

Suppose that $\mathbf{A}^{M\times N}$ is a matrix that holds measured values of N sensors, where the i-th row was measured at a known position CoP_x and CoP_y with a known weight m. The $\overrightarrow{\mathrm{CoP}}_x^{M\times 1}$, $\overrightarrow{\mathrm{CoP}}_y^{M\times 1}$ and $\vec{m}^{M\times 1}$ hold the known values CoP_x, CoP_y and m. The equations that are solved by the least squares fitting algorithm are

$$\mathbf{A} \cdot \vec{c}_m = \vec{m}, \tag{4.6}$$

$$\mathbf{B} \cdot \vec{x} = \overrightarrow{\mathrm{CoP}}_x \cdot (\mathbf{B} \cdot \vec{1}_N), \tag{4.7}$$

$$\mathbf{B} \cdot \vec{y} = \overrightarrow{\mathrm{CoP}}_y \cdot (\mathbf{B} \cdot \vec{1}_N), \tag{4.8}$$

where $\mathbf{B}_{ij} = \mathbf{A}_{ij} \cdot (\vec{c}_m)_j$, $\vec{1}_N$ is the N dimensional vector of ones and $\vec{c}_m^{N\times 1}$, $\vec{x}^{N\times 1}$ and $\vec{y}^{N\times 1}$ are the values we are looking for with calibration.

4.3 Data preprocessing

Data preprocessing is a crucial step before analyzing the collected data. The preprocessing stage is the preparation of the data so that they come in specified format and with specific properties, which make future analysis easier. In this chapter, I will focus on data interpolation and filtration.

Interpolation is needed because the sensor values are read and processed as they arrive on the CAN bus. That results in a variable sampling frequency, which could cause problems in the filtration. Filtration aims to minimize noise that may affect the data.

4.3.1 Interpolation

Interpolation is the process of estimating the values between two already known data points. It is based on an algorithm that creates a function that passes through already known data points. The function is then used to estimate the desired values. Interpolation can be used to make predictions or fill in missing data points.

I use interpolation as a process to obtain data points at a constant sampling rate. Another approach could be the synchronization of sensors. However, since the measurement process involves the calculation of the CoP position, whose speed depends on the CPU load of the control device, this approach would be complicated. I use an *interp* [21] function from Python's Numpy library for interpolation. Numpy uses optimized algorithms written in C, so the performance is much better than the pure Python implementation. When testing the prototype platform, the average sampling frequency that the control unit can achieve around 300 Hz. That is more than enough for further analysis, so I decided to lower the sampling frequency for interpolation to 200 Hz. That gives enough precision for further analysis and also reduces computational demand.

4.3.2 Filtering

In this section, I will explain the process of designing a filter that I would use on values that come from interpolation (CoP_x, CoP_y, and m). Since these values would be displayed and processed in real time, the filter also need to perform in real time. The biggest limitation when choosing a filter type comes from the fact that the values need to stay in sync after the filtration because they depend on each other. That means that the filter must have no or linear phase shift. Zero phase-shift filtering can be achieved with forward-backwards filtering, as implemented in Matlab's *filtfilt* function [22]. Unfortunately, this approach cannot be performed in real time, so the only option is to use linear phase shift filters. These filters have a constant delay, called a group delay, which is applied to the filtered data [23]. The group delay differs for every filter, so I will need to use the same filter for all three values. Figure 4.3 shows the data (in the frequency domain) measured on the prototype platform with a speed set to 9 km/h, while running on it.

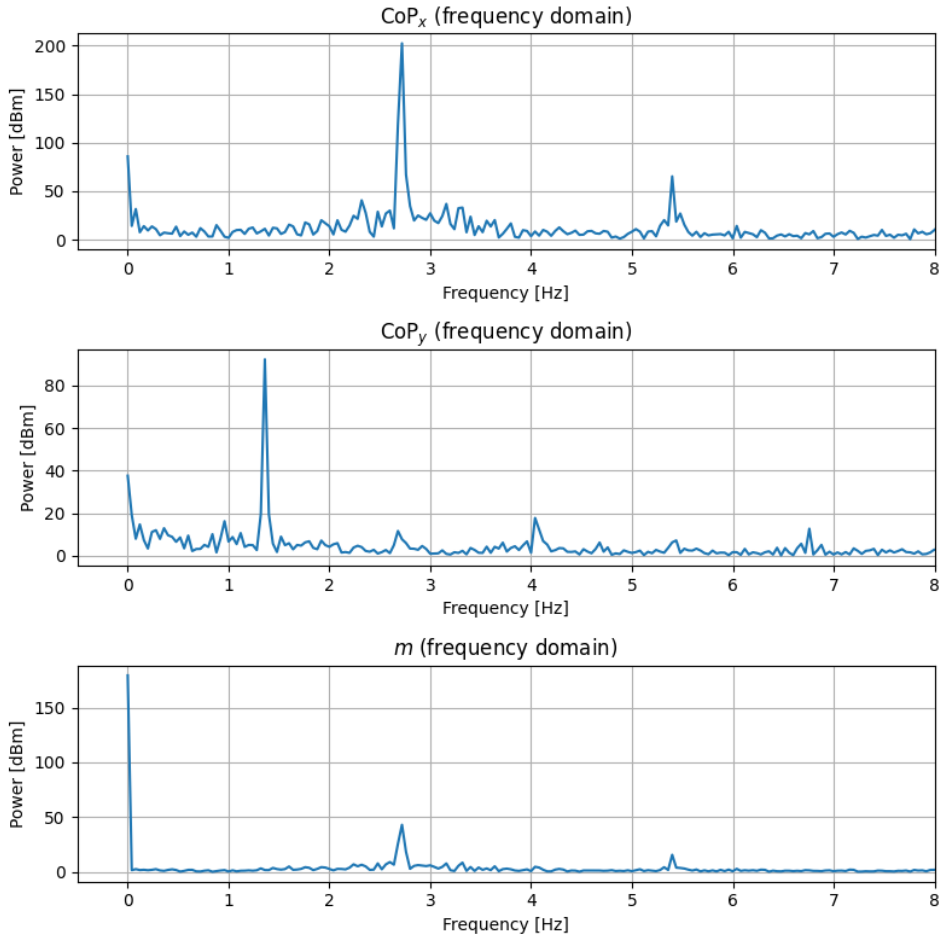

Figure 4.3: Data in frequency domain

Human running frequency can reach up to 15 km/h, and it is estimated that on average it took 1250 steps to run a km [24]. This results in an estimated step frequency for a fast run of around 5 Hz. Since the posturometric platform is aimed at rehabilitation, it is safe to say that the average running speed would be below 15 km/h (fast run), and so the step frequency would be around 4 Hz at maximum. That statement is confirmed in Figure 4.3, where CoP$_x$ has a dominant frequency below 3 Hz (the treadmill speed was set at 9 km/h). This means that the filter should have minimal effect on frequencies equal to and below 4 Hz. For this reason, I decided to use a low-pass filter. For designing and implementing the filter, I use Python's Scipy library, which has functions both for performing the filtration itself and also to help design the filter.

There are two main types of filters: Infinite Impulse Response (IIR) and Finite Impulse Response (FIR). IIR filters are typically easier to implement and have a lower delay, but they are very hard to design with a linear phase shift. On the other hand, the FIR filter has a linear phase shift when the

impulse response of the filter is symmetric [25]. The functions available in the Scipy library help to design a FIR filter that meets the specified criteria. The popular filter design method is a widow function like Butterworth, Chebyshev, and Kaiser [25]. I decided to use the Kaiser window method. The Scipy signal module has the function *kaiserord* [26] that determines the β parameter for the Kaiser window and the width of the filter. The parameters that I need to specify were

- cutoff frequency – frequency where the magnitude of the filter frequency response is -3 dB,
- width of transition between the pass band and the stop band,
- ripple – upper bound for the deviation (in dB) of the magnitude of the filter frequency response from that of the desired filter.

The ideal filter should filter out all frequencies above the maximum running frequency, so this means anything above 5 Hz, but the real filter is not ideal, so there will be a transition phase between the pass band and the stop band. I decided to set the cutoff frequency to 5 Hz so that the transition phase would start right after 4 Hz. I also want the gain in the pass band to vary by no more than 0.5 %, which corresponds to a ripple around 43 dB. The last parameter is the width of the transition phase, which I experimentally set to 3 Hz. This value has a great influence on the group delay value, so that a shorter transition means larger group delay, so I choose a compromise that suits the filter requirements.

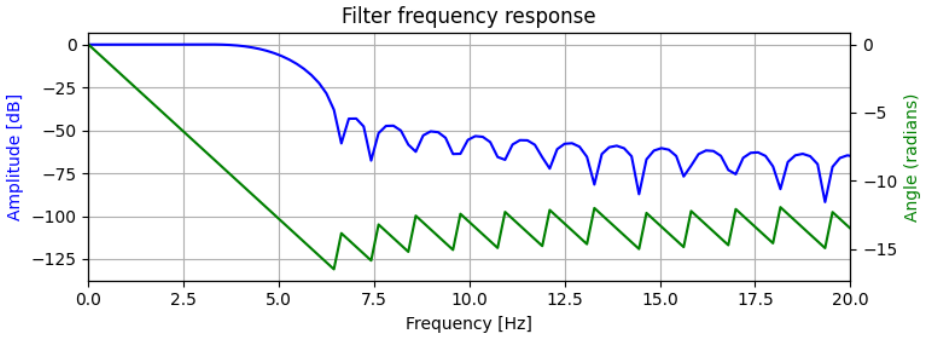

Figure 4.4: Designed filter impulse response and phase

Then I used the established parameters to calculate the filter itself with the *kaiserord* and *firwin* [27] functions of the Scipy signal module. The frequency response of the designed filter and the filter phase are shown in Figure 4.4. Then I apply the filter with the *lfilter* [28] function to the raw data with zero initial state for the filter. The difference between raw and filtered data are shown in Figure 4.5. Upon closer inspection of the filtered data compared to the raw data, it is clear that the mentioned group delay is present and that the filter helps smooth out the data.

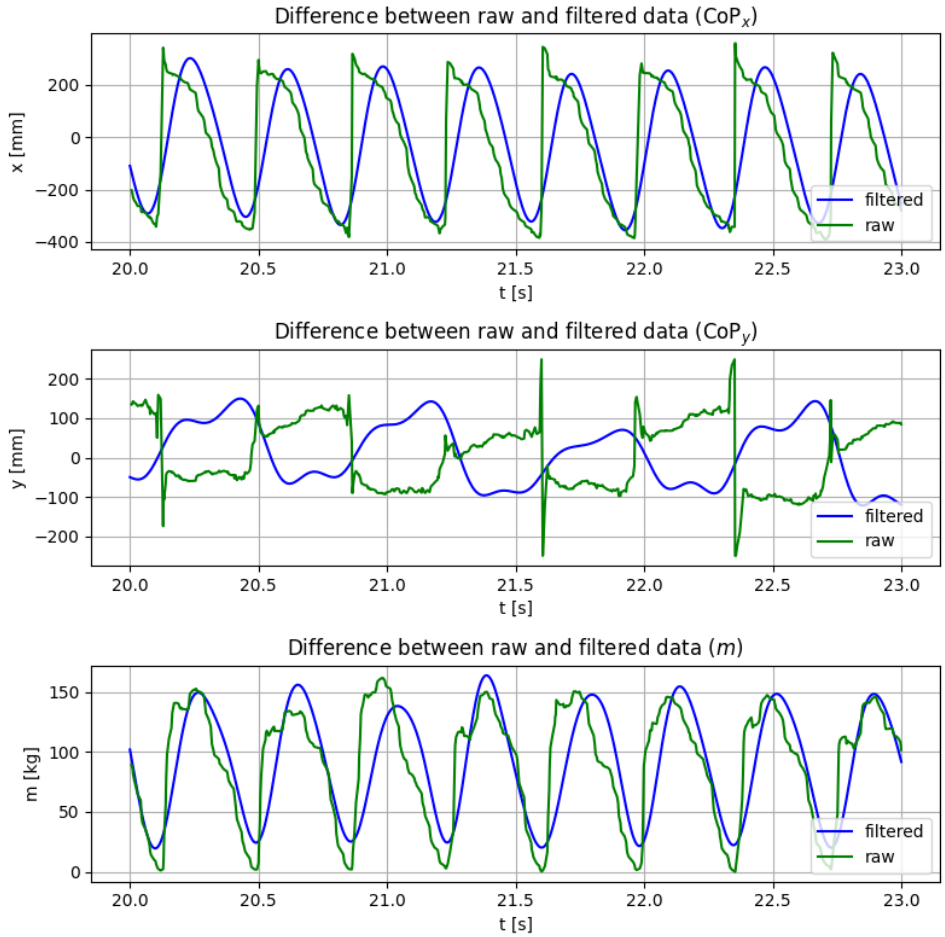

Figure 4.5: Difference between raw and filtered data

4.4 Static analysis

The static analysis is focused on monitoring patient stability when standing still with feet on marked positions. The crucial part for the doctor are the outcome graphs stabilogram and statokinesiogram, which displays data from one whole exercise or a combination of more previous exercises. That means that they are not supposed to be displayed in real time and that it is not necessary for doctors. The only functionality that can be used in real time is to show the position of CoP, with a circle defining the area where the position of CoP should be, to patients so that they can react and adjust their posture accordingly. The patients can then try to maintain posture so that the CoP position is in the middle of the marked area.

4.5 Dynamic analysis

Dynamic analysis focuses on monitoring the patient during movement, such as walking or running. The objective of the analysis is to isolate individual steps and count their characteristics, which could be used for further analysis or long-term monitoring. To isolate individual steps, the algorithm needs to know where the step starts and ends, which leads to a peak detection process.

4.5.1 Peak detection

Peak detection is a key part of dynamic analysis. I can inspect the individual steps with founded peaks and calculate the necessary parameters. For example, I can count the number of steps by the number of peaks in measured weight, where every peak represents the landing foot on a new step. The algorithm that I propose is inspired by the algorithm proposed in [29].
It's common practice to use the window detection method to detect peaks in real time [30]. The window is a portion of a certain width from the measured data points. The algorithm inspects the window for peaks and then moves the window by a specified offset. That means that there is no need to process all data points over and over again, which would be inefficient. The idea behind this process is illustrated by the diagram in Figure 4.6. Therefore, to make the algorithm work properly, it is necessary to set the window width to be lower than the length of the signal period, so two steps (two positive peaks) cannot occur in one window. The offset of window movement should be lower than the window size so that the algorithm would not miss any peak. I set the offset to half the size of the window.

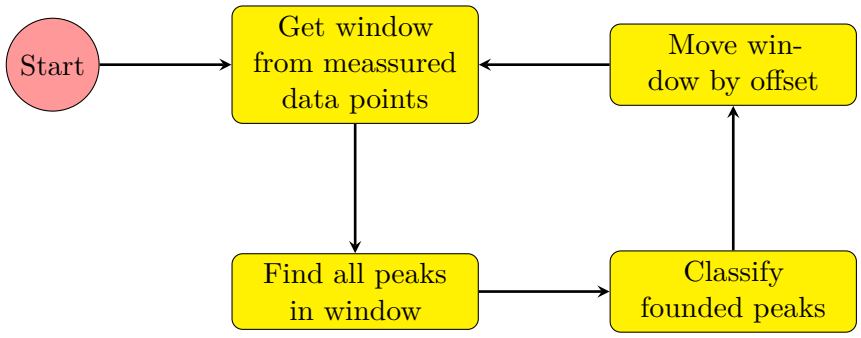

Figure 4.6: Peak detection diagram

The basic algorithm for peak detection is that every point whose neighboring values are smaller is labelled as a peak. The algorithm can be a very precise tool for step detection, but it can also detect unwanted peaks (false peaks) that are shown in Figure 4.7. This problem can be solved by introducing constrain conditions to the process, which can eliminate false detection. For the base peak detection, I ended up using the *find_peaks* [31] function from

Scipy's signal module. This function is written in C and is very well optimized and has more additional features, such as determining the height and width of the peak. I also tried implementing the base algorithm myself, but the pure Python implementation was slower in comparison to the Scipy version.

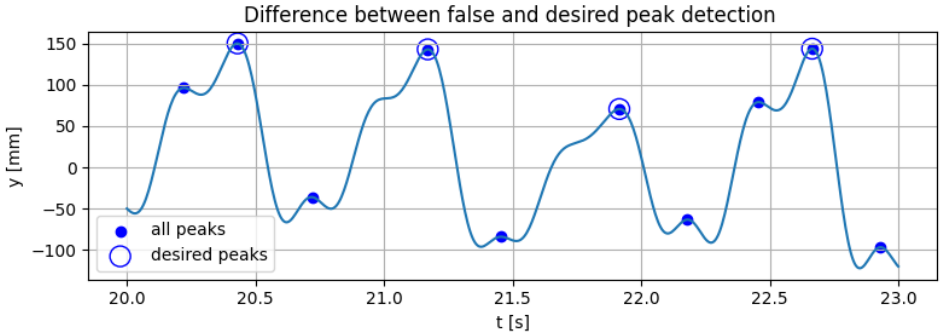

Figure 4.7: Difference between false and desired peak detection

Now, when I have peaks in the window, I can set a classification process that classifies which peak is positive and which is a false detection. The classification process is just a set of conditions that every peak has to meet to be classified as a positive peak and saved.

Now, when I have peaks in the window, I can set a classification process that classifies which peak is positive and which is a false detection. The classification process is just a set of conditions that every peak has to meet to be classified as a positive peak and saved. Suppose that Win is a set of data points in the window, $Peaks$ is a set of founded peak values in the window, $PosPeaks$ is a set of values of peaks already positively classified, and t returns the time of a given peak or point. Parameters

$$M_i = \begin{cases} \text{True}, & \text{if } Peaks(i) \geq max(Win) \\ \text{False}, & \text{otherwise} \end{cases}, \tag{4.9}$$

$$T_i = t(Peaks(i)) - t(PosPeaks(-1)), \tag{4.10}$$

$$S_i = -max(Peaks(i) - mean(PosPeaks(-N:-1)), \tag{4.11}$$

are counted for every peak in the window. The parameter M represents the maximality of the peak in the window, T is the periodicity of the peak, and S is the similarity of the peak, which uses N previous positively classified peaks. The classification process then uses these parameters and checks if they meet a certain threshold (δ_T, δ_S) or a certain value. The next problem occurs when there are two peaks that are really close to each other and are part of the same step. In this case, the first peak might be positively classified, and the second peak would be missed because of the periodicity parameter. This can be fixed by specifying that if the window finds a peak that is already

positively classified and also finds a peak whose value is greater than the positively classified peak, it will delete the positively classified peak and check the new peak. For easier representation, I call this function *PeakDualityCheck*. The classification process is summarized in Figure 4.8.

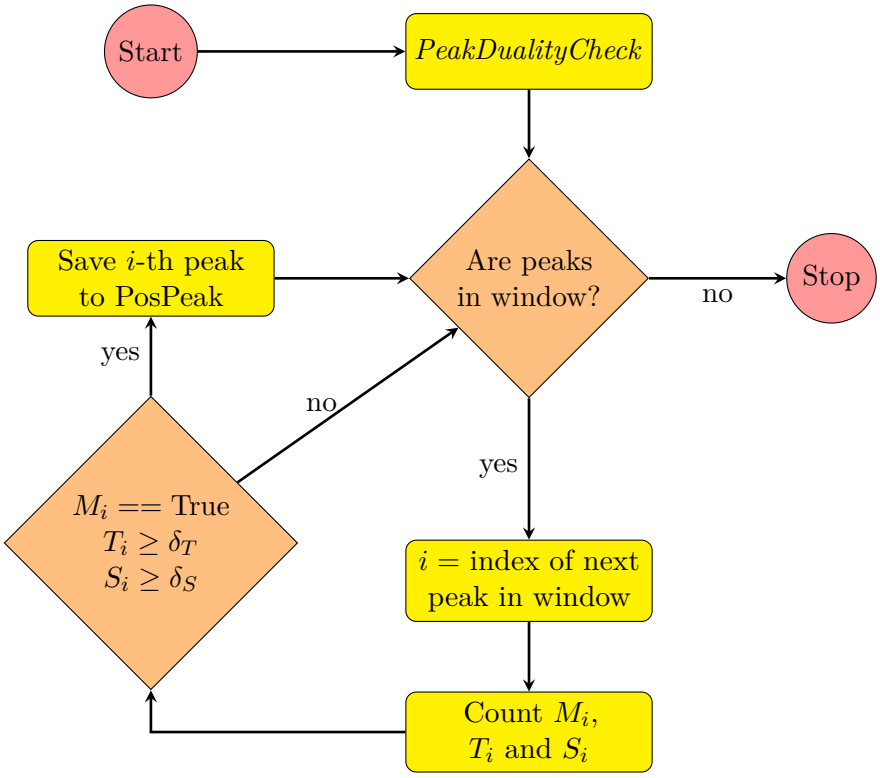

Figure 4.8: Peak classification diagram

Data type	Window width [ms]	δ_T [ms]	δ_S
CoP_x [mm]	200	200	-150 mm
CoP_y [mm]	400	400	-100 mm
m [kg]	200	200	-50 kg

Table 4.1: Parameters for algorithm

I use the described process for all data sets that I obtain from the sensors (CoP_x, CoP_y and m). The process can be used for rising and falling peaks (I need falling peaks only for CoP_x and CoP_y). To detect falling peaks, multiply the window by -1. Finally, I need to determine the parameters of the algorithm and test how it performs. For this purpose, I used the measured data that I plotted with the peaks found and then experimented with the parameters until I got a satisfactory result. The parameters that I use are specified in Table 4.1. Sections of the measured data with calculated peaks are shown in Figure 4.9.

In Figure 4.9 it is clear that the algorithm can correctly detect the desired peaks. Unfortunately, the algorithm does not perform very well when the

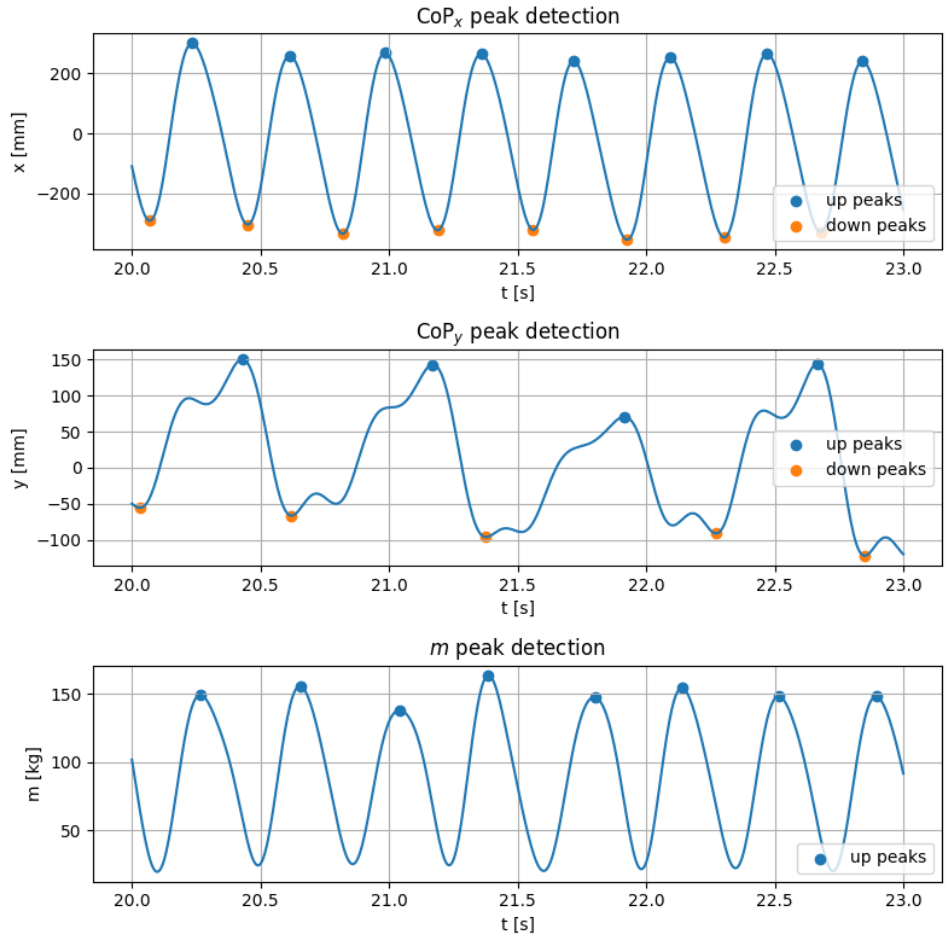

Figure 4.9: Difference between false and desired peak detection

patient suddenly goes down from the treadmill or stops the treadmill. In this case, an unwanted peak may meet the conditions and become positively classified. That may affect the peaks that follow. The solution is that the specialist who supervises the patient has to start the measurement (exercise) after the patient begins to walk or run and stop the measurement before the patient finishes the walk or run.

4.5.2 Step counting

The number of steps the patient takes throughout the exercise can be calculated in two ways. The first is based on the fact that when the patient is walking or running, the beginning of every step, the moment when the foot hits the platform, will correspond to a high peak in measured weight. That means that the number of steps can be estimated from the number of peaks in the weight data. The second method uses the CoP position. Specifically, the CoP position on the axis that is identical to the direction of movement of

the patient (CoP_x). The peaks on this axis also correspond to the beginning of the steps.

Both methods have similar results for running and fast walking, where the impact on the platform is significant at the beginning of each step. However the calculation with weight losses accuracy for slow walking, during which the weight is lower and some peaks might get missed. For this reason, I decided to use the CoP_x peaks to calculate the number of steps.

4.5.3 Step width, length and walked distance

The step width is an absolute value of the difference between the CoP_x upward peaks and the CoP_x downward peaks. The same principle works for the step length but with CoP_y peaks. For the calculation, I combine all the values of the peaks (up and down peaks), from one data set (CoP_x or CoP_y), into one array and sort them by time. From this, the desired values can be obtained by calculating the absolute values of the differences between all the peaks in the sorted array. The same process is used for both the length and width of the steps.
The distance that the patient has walk through the exercise, is equal to the sum of every step length.

4.5.4 Gait speed

The gait speed is proportional to the movement speed of CoP caused by the treadmill belt. The direction of the belt is opposite to the x-axis of the platform (Figure 4.2), so the movement speed of CoP_x is the required gait speed. Suppose that x are the values of CoP_x. The speed can be calculated as

$$v_i = (x_i - x_{i+1}) \cdot fs, \tag{4.12}$$

where fs is the sampling frequency of the signal. I then calculate the mean of only the positive speed values (negative values are caused by the leg switching during walking).

Chapter 5

Implementation

My goal is to create an application that should function as a proof of concept and utilize the algorithms and methods mentioned in Chapters 3.2 and 4 to calculate and present the results. The application can be divided into three main segments, communication (data collection), data analysis, and data presentation through the GUI. As described in Chapter 3.3, I use Python to write the application, CAN interface for communication, and PyQt as a GUI framework. For easier maintenance and testing, I take advantage of a configuration yaml file, which stores parameters used throughout the application and the algorithms.

5.1 Communication

This part of the application is responsible for handling communication with sensors, which includes collecting measured values, computing CoP position and weight, handling errors and commands such as the tare command. Tare is a necessary procedure to limit the effect of initial sensor preload. The taring process sends a defined command to all sensors, which then reply when the taring is done.

Python has a Python-can library that allows one to use the CAN interface with Python. The library has all the necessary features, such as setting CAN filters. I use a library notifier class [32] that starts a thread in which all incoming messages are distributed to the specified listeners. The listener is a class that takes the incoming message and processes it [32]. The library comes with several already specified listener classes. For example, listeners for logging, printing and saving messages are already prepared in the library [32].

I use the listener base class to create a custom listener that saves incoming messages with measured values into an array that holds the last value from all six sensors. When this array is full (the listener gathered value from every

sensor), the CoP position and weight calculation is performed as described in Chapter 4.2. The calculated values are then interpolated, filtered and stored in a buffer.

We have designed the basic CAN communication structure together with Mr. Šťastný, who is responsible for the communication on the sensor side. To ensure the unity of the designed messages, we decided that Mr. Šťastný would take care of it and that I would use these specifications.

5.2 Data analysis

The sensor data analysis algorithms are covered in Chapter 4. All algorithms and calculations presented are called periodic by the PyQt event handler. I can set the event handler to call specific functions at certain periods. In order to display the measured data smoothly, I decided to re-render the graphs in the application at 30 Hz.

For the dynamic analysis functions, I set the call period to the window shifts of specific data types (CoP_x, CoP_y and m). All accumulated data and specifications about the current exercise are stored in an object with the working title Episode. The Episode contains information about the patient, the start of the exercise, and buffers with measured data and established peaks. The Episode is stored and reset when changing the exercise type.

5.3 Graphic User Interface

For GUI design, I take an inspiration from [10]. The application consists of separate windows for each exercise type and a menu window that functions as a signpost between exercises. Since part of GUI focused on long-term data analysis is developed by Mr. Laušman, I prepare base classes and styles that Mr. Laušman can use. That will guarantee that the application style will be consistent across the entire application.

The menu window is the first thing the user sees after starting the control device. This window has a navigation bar and a settings bar. The buttons in the navigation bar redirect the user to the different exercise windows. In the settings bar, there is only one button, the tare button, but in the future it can involve a complex settings bar that could be customized for the end user. When the user switches to one of the exercises, the applications automatically initialize the Episode object for the particular exercise that starts the measurement. Figure 5.1 shows a screenshot of the menu window.

The window for dynamic exercise shown in Figure 5.2 is built around the main graph that shows the CoP position with a trajectory trail. Next to the

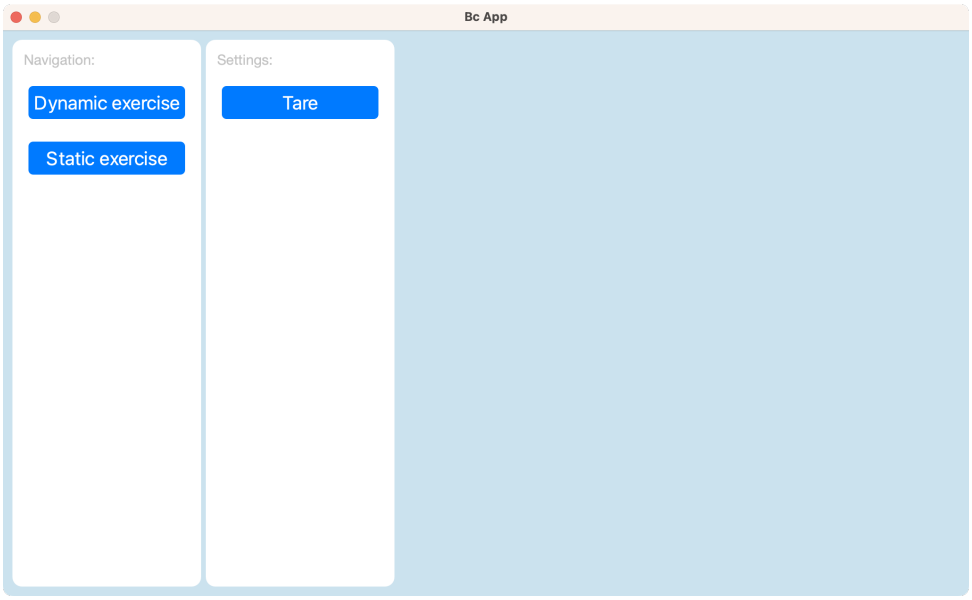

Figure 5.1: Application menu window

graph are slabs with a calculated number of steps, step length, width, walked distance, and patient's gait speed. On the top of the window is a top bar that is identical for every exercise window and holds a patient name, current time, name of the exercise, menu button and stop button. The menu button redirects the user to the menu window. The stop button stops the current exercise.

The static exercise window shows a graph similar to that of the dynamic exercise window, but the static exercise graph is more focused and has a marked area where the patient should keep the CoP position. The top bar is the same as in the other exercise windows. The static exercise window is shown in Figure 5.3.

All parts of the application are prepared for future customization that the end user (doctor) would need. The application graph all displays data at 30 Hz thanks to a very fast PyQtGraph library that is used to display the graphs.

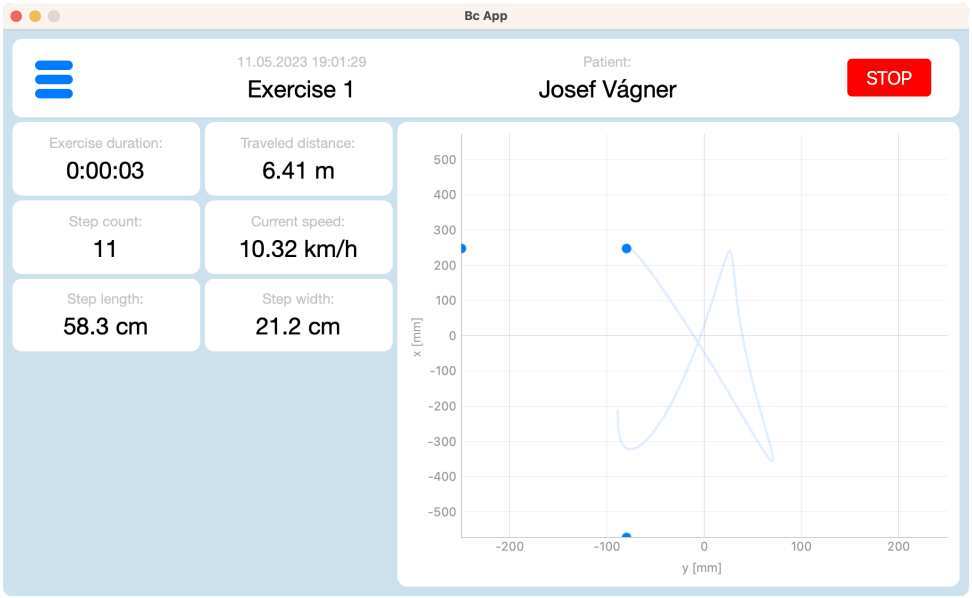

Figure 5.2: Application dynamic exercise window

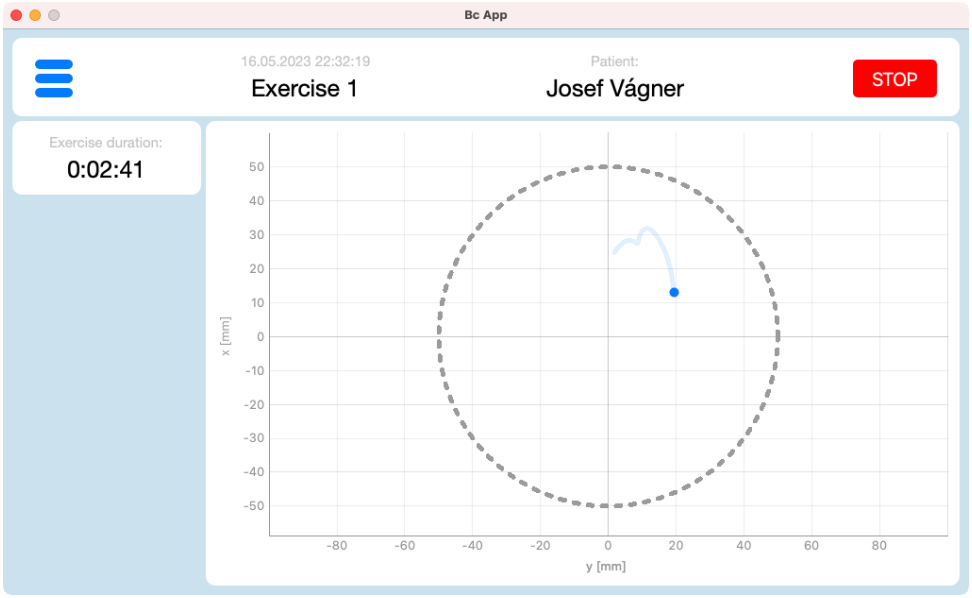

Figure 5.3: Application static exercise window

Chapter 6

Test execution and evaluation

I used the prototype platform mentioned in Chapter 4, in order to test the functionality and precision of the proposed algorithms. Unfortunately, the measurements presented in this chapter are only indicative and serve to demonstrate the functionality of the methods used. The main reason is the lack of stiffness of the platform. That causes the platform to bend between the mounting points, negatively impacting the measurements. This problem is very noticeable when walking on the platform when the flexing of the plate can be felt directly. To minimize this issue, I performed the calibration described in Chapter 4.2. To accurately place the calibration weights, I created a calibration template of the size of the platform with marked positions to place the weights. This template is shown in Figure 6.1.

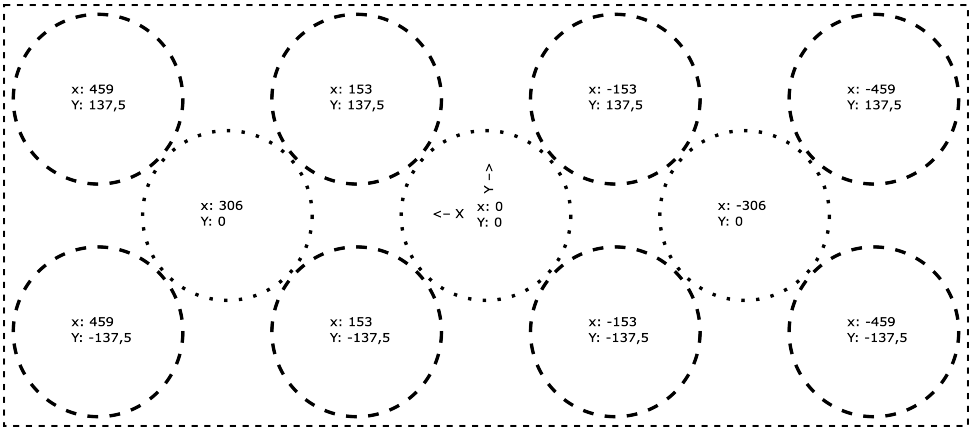

Figure 6.1: Calibration template

The platforms used on standard treadmills serve only as solid support on which the belt rides. The fact that the platform is made of softer material is typically not a problem unless you want to use such a platform for measurement. The location and mounting method of the strain gauges on the prototype platform can also cause measurement problems. For structural reasons, the strain gauges are mounted next to the platform mounting bolts.

That may cause the platform to partially rest on the mounting bolts, which may affect the measurement. A custom-made platform that is strong enough and designed specifically for this application should eliminate these problems.

To be as close as possible to real conditions. I took all the measurements with a live person. That may introduce slight inaccuracies in the testing process. For example, it cannot be determined with absolute certainty whether a person is standing or leaning in a precise position. However, since this is an indicative measurement and primarily a functional test, these distortions can be neglected. On the other hand, this procedure indicates how the current solution would perform in live operation.

6.1 Accuracy of CoP position and weight measurement

Determining the exact position of CoP and measuring the weight applied to the platform is essential for the later parts of the analysis of the patient's gait. If the resulting CoP position is very inaccurate, it will have a negative effect on the other calculated parameters.

I took the measurements by having the test person stand in a predetermined position on the platform. I then recorded the CoP position and the measured weight. I repeated this process for different positions on the platform. The measured data can be seen in Tables 6.1 and 6.2.

Real values		Measured values		
CoP_x [mm]	CoP_y [mm]	CoP_x [mm]	CoP_y [mm]	Δ_{CoP} [mm]
-153.0	-137.5	-241.1	-170.7	94.1
-153.0	137.5	-207.5	197.3	80.9
-306.0	0.0	-322.1	-4.4	16.7
-459.0	-137.5	-415.8	-161.9	49.6
-459.0	137.5	-362.8	159.3	98.6
0.0	0.0	26.7	41.7	49.5
153.0	-137.5	227.9	-162.0	78.8
153.0	137.5	237.7	152.7	86.1
306.0	0.0	292.4	14.6	19.9
459.0	-137.5	339.3	-142.4	119.8
459.0	137.5	320.0	167.7	142.3
Δ_{CoP} mean				76.0

Table 6.1: Data from testing CoP position measurement accuracy

Table 6.1 contains the data of the measured and real positions of CoP, where Δ_{CoP} is the distance between the measured and the real position. It can be seen that the largest deviation is 14.23 cm, and the average deviation

is 7.6 cm. Given the platform dimensions, this gives an average percentage deviation of 8.4 % (the maximum is 13.6 %). Considering the problems with the prototype platform, this is a satisfactory result.

Real m [kg]	Measured m [kg]	Δ_m [kg]
93.6	93.3	-0.3
	104.5	10.9
	118.6	25.0
	95.1	1.5
	88.9	-4.7
	75.3	-18.3
	99.5	5.9
	91.0	-2.6
	120.9	27.3
	90.8	-2.8
	94.1	0.5
$\lvert\Delta_m\rvert$ mean		9.1

Table 6.2: Data from testing weight measuring accuracy

Table 6.2 contains the measured and real weight data, where Δ_m is the difference between the measured and the real weight. The real weight of the test person was obtained using a personal scale. It can be seen that at some positions the measured weight is higher than the real weight and at positions lower. That may be due to the mentioned problems with the prototype platform. The largest measured weight difference is 27 kg, and the average weight deviation is 9.1 kg. Therefore, the average percentage difference between measured and actual weight is 9.7 %. That could be considered a satisfactory result, but very large deviations of around 25 kg may have a negative impact on further analysis.

6.2 Step counting accuracy

Step counting was tested by having the test person walk (run) on the treadmill at different speeds, during which the number of steps taken and the number of steps measured by the app were monitored. Both values, along with the set speed of the treadmill, are shown in Table 6.3.

It can be seen from Table 6.3 that the variation in the number of steps is within one of steps. That can be caused, for example, by observation errors. The accuracy of the proposed algorithm for step counting does not depend so much on the accuracy of the CoP position measurement. Since the algorithm works with peaks, it does not matter whether the measured data are slightly inaccurate. For these reasons, good results can be seen in this measurement despite the mentioned platform issues.

Set speed [km/h]	Counted steps	Measured steps
3	50	49
3	53	54
3	50	52
6	50	51
6	50	51
6	50	52
9	51	51
9	57	57
9	48	48

Table 6.3: Data from testing step counting accuracy

6.3 Gait speed accuracy

Due to limited options, the accuracy of the speed calculation was tested using a bicycle with a bicycle computer that measured the speed of the belt while the test person walked or ran on the belt. Measurements were taken for different treadmill speed settings. The measurements are shown in Table 6.4, where A is the difference between the speed measured by the bicycle computer and the speed calculated from the CoP position.

Measured speed [km/h]	Computed speed [km/h]	Δ_v [km/h]
3.4	3.5	0.1
3.2	3.3	0.1
3.4	3.3	0.1
6.6	6.4	-0.2
6.5	6.3	-0.2
6.4	6.4	0
9.7	9.9	0.2
9.6	9.8	0.2
9.6	9.4	-0.2

Table 6.4: Data from testing gait speed accuracy

The data in Table 6.4 show very good results for the gait speed measurements. Since the speed measurement is based on the calculation of the movement speed of the CoP position, this measurement is not much affected by the inaccuracy of the CoP position measurement. Even if the measured data differ from the real CoP position, the motion of the CoP position is preserved.

6.4 Step width, length and walked distance

During testing, I found that the negative effect of the bending of the platform was found to be more significant during walking and running than under static conditions. That negatively reflected in the measurements of distance traveled, step length, and step width. The custom platform mentioned above should eliminate this problem. Furthermore, the accuracy of this measurement could not be accurately determined because it was not possible to obtain accurate control values to check with the measured data.

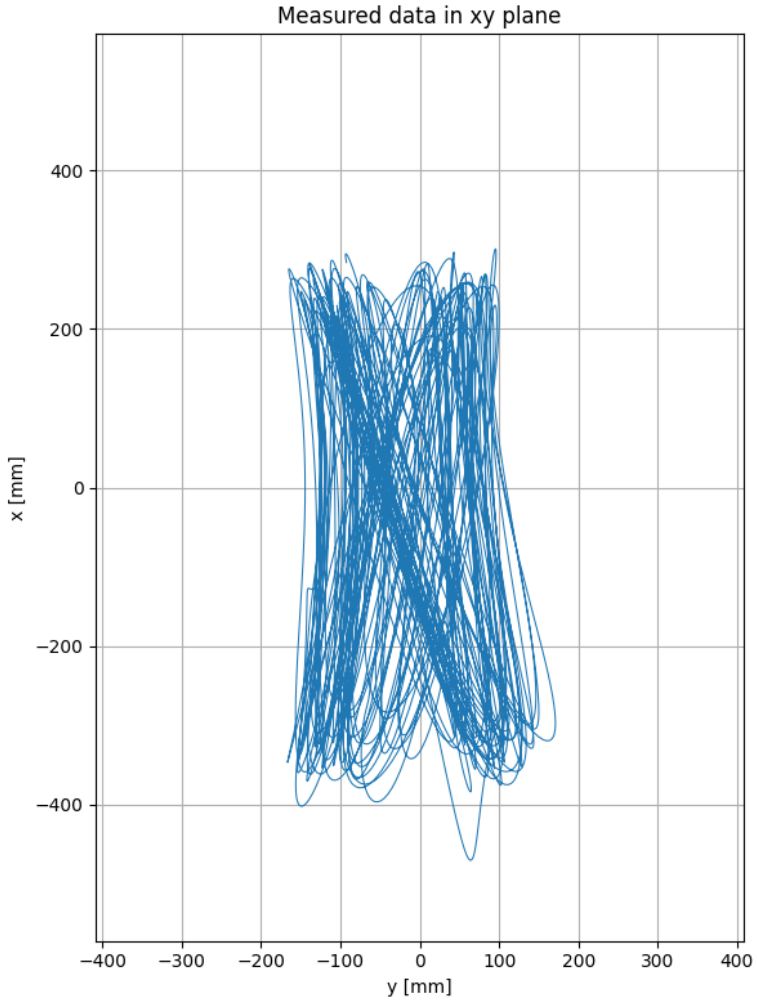

Figure 6.2: Calibration template

To verify the functionality, the measured data were plotted on a xy plot. The plotted data can be seen in Figure 6.2. From the figure it can be seen that the step width is around 20 cm and the step length is around 55 cm. These values approximately match the visual estimation made during the measurement.

Chapter 7

Conclusion and future work

The aim of this thesis was to create an application and the necessary software for the posturometric platform project. That includes the design of algorithms and procedures necessary to obtain the basic parameters of the patient's gait and stability. These goals were achieved.

The communication part of the application was designed to be responsible for collecting data from sensors that use the CAN bus. The collected data were then interpolated and filtered. A process was implemented to calculate the patient's CoP position and the weight applied to the platform.

After reviewing existing solutions, it was decided to analyze the basic parameters of the patient's gait: number of steps, walking speed, distance travelled, step width, and length. An algorithm was designed and implemented to recognize and analyze individual steps, which allowed the calculation of the specified parameters.

Finally, a user interface was designed to display the obtained values. Everything was implemented in Python. The testing was carried out on a real prototype platform with a control device created by my colleagues Mr. Laušman and Mr. Šťastný. The results of the tests were evaluated and, despite the problems caused by the low strength of the prototype platform, were satisfactory.

7.1 Future work

Future work includes further development and refinement of the proposed application and software, which now serves as a proof of concept. The aim of future work will be to complete the application in its final form.
Furthermore, everything needs to be tested on a custom-made platform for the posturometric platform project. This platform is already in the manufacturing process and should provide much better results.

Bibliography

[1] P. Arpaia, P. Cimmino, E. De Matteis, and G. D'Addio, "A low-cost force sensor-based posturographic plate for home care telerehabilitation exergaming", en, *Measurement*, vol. 51, pp. 400–410, May 2014, ISSN: 0263-2241. DOI: 10.1016/j.measurement.2014.01.031. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224114000451 (visited on Apr. 3, 2023).

[2] J. Bruštíková, "Use of Tenzometric Platform in Ambulatory Therapy", cs, bachelor thesis, Czech Technical University in Prague. Computing and Information Centre., Prague, Sep. 2016. [Online]. Available: http://hdl.handle.net/10467/67612 (visited on Apr. 4, 2023).

[3] T. Łukaszewicz, D. Kania, Z. Kidoń, and K. Pethe-Kania, "Posturographic methods for body posture symmetry assessment", en, *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 63, no. 4, pp. 907–917, Dec. 2015, ISSN: 2300-1917. DOI: 10.1515/bpasts-2015-0103. [Online]. Available: http://journals.pan.pl/dlibra/publication/97784/edition/84372/content (visited on Apr. 3, 2023).

[4] R. L. Kirby, N. A. Price, and D. A. MacLeod, "The influence of foot position on standing balance", en, *Journal of Biomechanics*, vol. 20, no. 4, pp. 423–427, Jan. 1987, ISSN: 0021-9290. DOI: 10.1016/0021-9290(87)90049-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021929087900492 (visited on May 7, 2023).

[5] M. Gallamini, "Treating Balance Disorders by Ultra-Low-Level Laser Stimulation of Acupoints", *Journal of acupuncture and meridian studies*, vol. 6, pp. 119–123, Apr. 2013. DOI: 10.1016/j.jams.2013.01.003.

[6] B. Helštýnová, "Analysis of Posturometric Data", cs, Accepted: 2012-07-11T07:50:06Z Publisher: Vysoká škola báňská - Technická univerzita Ostrava, M.S. thesis, Technical university Ostrava, Ostrava, 2012. [On-

line]. Available: http://dspace.vsb.cz/handle/10084/92986 (visited on Apr. 3, 2023).

[7] M. C. Beato, E. Morton, C. Iadarola, L. Winterberger, and N. Dawson, "Can the Wii Fit Balance Board be Used as a Fall Risk Assessment Tool among Poststroke Patients?", en, *Journal of Stroke and Cerebrovascular Diseases*, vol. 29, no. 2, p. 104 500, Feb. 2020, ISSN: 1052-3057. DOI: 10.1016/j.jstrokecerebrovasdis.2019.104500. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1052305719305841 (visited on Apr. 4, 2023).

[8] *Medical Devices Assess, Treat Balance Disorders | NASA Spinoff*. [Online]. Available: https://spinoff.nasa.gov/Spinoff2009/hm_2.html (visited on Apr. 4, 2023).

[9] *Treadmill Exercise Rehabilitation Improves Ambulatory Function and Cardiovascular Fitness in Patients With Chronic Stroke*, en. DOI: 10.1161/01.STR.0000181076.91805.89. [Online]. Available: https://www.ahajournals.org/doi/epub/10.1161/01.STR.0000181076.91805.89 (visited on Apr. 6, 2023).

[10] *AXELERO Gait&Balance*, pl-PL, Section: Aktualności, Oct. 2021. [Online]. Available: https://neurorehabilitacja.com.pl/aktualnosci/axelero-gaitbalance/ (visited on Apr. 6, 2023).

[11] C. Kittichaikarn and V. Kuptniratsaikul, "Design of an Underwater Treadmill System for rehabilitation of older obese adults: A pre-post study", *BMC Geriatrics*, vol. 19, Nov. 2019. DOI: 10.1186/s12877-019-1334-5.

[12] G. Walker, "A review of technologies for sensing contact location on the surface of a display: Review of touch technologies", en, *Journal of the Society for Information Display*, vol. 20, no. 8, pp. 413–440, Aug. 2012, ISSN: 10710922. DOI: 10.1002/jsid.100. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/jsid.100 (visited on Apr. 9, 2023).

[13] V. Goga, "Finite element model of the strain gauge For determining uniaxial tension", en, 2013, Accepted: 2016-06-27T06:33:11Z Publisher: Vysoká škola báňská - Technická univerzita Ostrava, ISSN: 1210-0471. [Online]. Available: http://dspace.vsb.cz/handle/10084/111747 (visited on Apr. 13, 2023).

[14] G. Hollings, *Strain Gauges: How they Work, Applications, and Types*, en. [Online]. Available: https://blog.endaq.com/strain-gauges-how-they-work-applications-and-types (visited on Apr. 13, 2023).

[15] P. Carsten, *The Why and How of Differential Signaling - Technical Articles*, en. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/the-why-and-how-of-differential-signaling/ (visited on Apr. 15, 2023).

[16] S. Aradi, T. Bécsi, and P. Gáspár, "Development of Vehicle On-board Communication System for Harsh Environment", *Acta Technica Jaurinensis*, vol. 6, pp. 53–63, Jan. 2013.

[17] J. Munson, *Enabling robust data communications within a high voltage BMS*, en, Jul. 2016. [Online]. Available: https://www.newelectronics.co.uk/content/features/enabling-robust-data-communications-within-a-high-voltage-bms/ (visited on Apr. 15, 2023).

[18] T. Brand, *Isolated SPI Communication Made Easy | Analog Devices*. [Online]. Available: https://www.analog.com/en/technical-articles/isolated-spi-communication-made-easy.html (visited on Apr. 15, 2023).

[19] H. Boland, M. Burgett, A. Etienne, and R. III, "An Overview of CAN-BUS Development, Utilization, and Future Potential in Serial Network Messaging for Off-Road Mobile Equipment", in Jul. 2021, ISBN: 978-1-83881-921-7. DOI: 10.5772/intechopen.98444. [Online]. Available: https://www.researchgate.net/publication/353499375_An_Overview_of_CAN-BUS_Development_Utilization_and_Future_Potential_in_Serial_Network_Messaging_for_Off-Road_Mobile_Equipment.

[20] A. Alshammari, M. Zohdy, D. Debnath, and G. Corser, "Classification Approach for Intrusion Detection in Vehicle Systems", *Wireless Engineering and Technology*, vol. 09, pp. 79–94, Jan. 2018. DOI: 10.4236/wet.2018.94007.

[21] *Numpy.interp — NumPy v1.24 Manual*. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.interp.html (visited on Apr. 29, 2023).

[22] *Zero-phase digital filtering - MATLAB filtfilt*. [Online]. Available: https://www.mathworks.com/help/signal/ref/filtfilt.html (visited on Apr. 30, 2023).

[23] A. V. Oppenheim, *Discrete-time signal processing / Alan V. Oppenheim, Ronald W. Schafer, with John R. Buck.* (Prentice-Hall signal processing series), eng, Second edition. Upper Saddle River, N.J: Prentice Hall, 1999, ISBN: 978-0-13-754920-7.

[24] W. W. K. Hoeger, L. Bond, L. Ransdell, J. M. Shimon, and S. Merugu, "ONE-MILE STEP COUNT AT WALKING AND RUNNING SPEEDS", en-US, *ACSM's Health & Fitness Journal*, vol. 12, no. 1, p. 14, Feb. 2008, ISSN: 1091-5397. DOI: 10.1249/01.FIT.0000298459.30006.8d. [Online]. Available: https://journals.lww.com/acsm-healthfitness/fulltext/2008/01000/one_mile_step_count_at_walking_and_running_speeds.7.aspx (visited on May 10, 2023).

[25] S. Mukherjee, R. Kar, D. Mandal, S. Mondal, and S. P. Ghoshal, "Linear phase low pass FIR filter design using Improved Particle Swarm Optimization", in *2011 IEEE Student Conference on Research and Development*, Dec. 2011, pp. 358–363. DOI: 10.1109/SCOReD.2011.6148765.

[26] *Scipy.signal.kaiserord — SciPy v1.10.1 Manual*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.kaiserord.html (visited on May 10, 2023).

[27] *Scipy.signal.firwin — SciPy v1.10.1 Manual*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.firwin.html (visited on May 10, 2023).

[28] *Scipy.signal.lfilter — SciPy v1.10.1 Manual*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.lfilter.html (visited on May 10, 2023).

[29] F. Gu, K. Khoshelham, J. Shang, F. Yu, and Z. Wei, "Robust and Accurate Smartphone-Based Step Counting for Indoor Localization", *IEEE Sensors Journal*, vol. 17, no. 11, pp. 3453–3460, Jun. 2017, Conference Name: IEEE Sensors Journal, ISSN: 1558-1748. DOI: 10.1109/JSEN.2017.2685999.

[30] D. Salvi, C. Velardo, J. Brynes, and L. Tarassenko, "An Optimised Algorithm for Accurate Steps Counting From Smart-Phone Accelerometry", in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, ISSN: 1558-4615, Jul. 2018, pp. 4423–4427. DOI: 10.1109/EMBC.2018.8513319.

[31] *Scipy.signal.find_peaks — SciPy v1.10.1 Manual*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html (visited on May 10, 2023).

[32] *Reading and Writing Messages - python-can 4.2.0 documentation*. [Online]. Available: https://python-can.readthedocs.io/en/stable/listeners.html (visited on May 11, 2023).