Ole Einar Kværnsveen Belboe
Elias Loe Eritsland
Admir Meta

Next generation handgrip trainers

Bachelor's thesis in Electrical Engineering: Automation and Robotics
Supervisor: Roya Doshmanziari
Co-supervisor: Roxanne Jackson, Steffi Knorn, Damiano Varagnolo
May 2023

Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Ole Einar Kværnsveen Belboe
Elias Loe Eritsland
Admir Meta

Next generation handgrip trainers

Bachelor's thesis in Electrical Engineering: Automation and Robotics
Supervisor: Roya Doshmanziari
Co-supervisor: Roxanne Jackson, Steffi Knorn, Damiano Varagnolo
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Kunnskap for en bedre verden

Department of Engineering Cybernetics

IELET2920 - Bachelor Thesis: Automation and Robotics

Next generation handgrip trainers

Author:
Ole Einar Kværnsveen Belboe
Elias Loe Eritsland
Admir Meta

Date: May 22 2023

Abstract

This thesis primarily focuses on the design process undertaken to develop a handgrip trainer, with “measurement data” and process said data. The trainer incorporates piezoresistive pressure sensors for the purpose of accurately measuring grip strength. A comprehensive description of the measurement circuit is provided later in the thesis. Furthermore, the thesis details the 3D-printed model that houses the sensors, enabling users to apply pressure during squeezing intuitively.

Significant emphasis is also placed on the design process of the Kalman Filter. This filtering technique plays a vital role in reducing noise interference in the grip strength measurements. Moreover, the Kalman Filter is specifically tailored to estimate the ”Fatigued muscle mass”, a hidden variable in the dynamical model describing the process that lacks a direct measurement method. The thesis outlines the design methodology employed to effectively utilize the Kalman Filter for filtering the sensor signal.

Summary

This thesis explains the development process for a handgrip trainer which can collect measurements, log data and accurately describe grip dynamics. This means that the developed handgrip trainer must be able to measure the active muscle mass, as well as estimate the fatigued muscle mass in the contraction. This should be done for all the fingers (excluding the thumb), in order to accurately describe the grip dynamics for each finger. Therefore the handgrip trainer contains four separate sensors, one for each finger. The trainer device is modelled and 3D printed to incorporate the sensors inside itself, with pressure pins to squeeze on them. This creates intuitive pressure points while ensuring proper pressure distribution over the sensors. Each sensor's circuit consists of a piezoresistive sensor, an operational amplifier, a resistor and a 9V battery. When combined these give a linear relationship between pressure and sensor output. These measurements are collected with an ESP32 that logs the data to a PC over serial communication. This collected data set is then used to estimate the parameters for the mathematical model that describes the grip strength dynamics. This is done via a least squares estimator. The estimated model is then further used to help generate the fatigued muscle mass state that cannot be measured directly. This is done by implementing a Kalman filter in the ESP32. This filter reduces the noise from the measured active muscle mass and generates an estimate of the fatigued muscle mass. Additionally, this report discusses the challenges present in the development process and discusses future solutions, aiming to improve the future design of a handgrip trainer that supports accurate measurements and describes grip dynamics.

Denne oppgaven forklarer utviklingsprosessen for en håndgrepstrener som kan samle inn målinger, loggføre data og nøyaktig beskrive grepets dynamikk. Dette betyr at den utviklede håndgrepstreneren må være i stand til å måle den aktive muskelmassen og estimere den slitne muskelmassen i sammentrekningen. Dette skal gjøres for alle fingrene (unntatt tommelen) for å nøyaktig beskrive grepets dynamikk for hver finger. Derfor inneholder håndgrepstreneren fire separate sensorer, én for hver finger. Treneren er modellert og 3D-printet for å inneholde sensorene inne i seg selv, med trykkstifter for å klemme på dem. Dette skaper intuitive trykkpunkter samtidig som det sikrer riktig trykkfordeling over sensorene. Hver sensors krets består av en piezoresistiv sensor, en operasjonsforsterker, en motstand og et 9V-batteri. Når de kombineres, gir de en lineær sammenheng mellom trykk og sensormålinger. Disse målingene samles inn med en ESP32 som logger dataene til en datamaskin via seriellkommunikasjon. Denne innsamlede datasettet brukes deretter til å estimere parametrene for den matematiske modellen som beskriver grepstyrkens dynamikk. Dette gjøres ved hjelp av en minste kvadraters estimator. Den estimerte modellen brukes deretter videre til å hjelpe med å generere den slitne muskelmasse-tilstanden som ikke kan måles direkte. Dette gjøres ved å implementere et Kalman-filter i ESP32. Dette filteret reduserer støyet fra målt aktiv muskelmasse og genererer et estimat av den slitne muskelmassen. I tillegg diskuterer denne rapporten utfordringene som oppstår i utviklingsprosessen og drøfter fremtidige løsninger, med sikte på å forbedre fremtidig utforming av en håndgrepstrener som støtter nøyaktige målinger og beskriver grepets dynamikk.

Table of Contents

List of Figures

List of Tables

Listings

1 Introduction

Handgrip trainers are tools that can be used to improve handgrip strength, which is an essential component of physical health. As explained by J. Depp [10], grip strength is just as important for everyday life as it is for athleticism. It is additionally reported to be an important predictor of good health, muscular endurance, dexterity and overall strength [8] and has been used in a variety of clinical areas for multiple purposes [1]. Therefore, a reliable measurement of grip strength is a necessity, from both a fitness and a rehabilitation point of view. However, even though there are many different handgrip trainers on the current market, none of them are equipped with a measurement system that can provide accurate data on handgrip strength.

This thesis will therefore document the research and development of a handgrip trainer with a measurement system that can provide accurate and reliable data on handgrip strength. The project in this thesis will therefore aim to explain the design, development, and testing of this handgrip trainer with these features.

2 Literature review

2.1 Introduction

As described in “Section 1”, there is a gap in the market for handgrip trainers that document and measure data. As such, the literature on possible devices for this project is quite scarce. This is shown further in the thesis, where the physical model is designed from scratch. The only literature source found was from the project work which this thesis was built upon [9].

However, the second part of the project, the Kalman filter, is a well-known estimation algorithm with a myriad of published articles, books, and research papers aiming to improve both the theoretical background of the algorithm and its implementation on different architectures. Therefore, this review will be more focused on the literature behind Kalman filters, drawing a comparison between the literature used by the group and other sources.

2.2 Background

The “Kalman filter and Bayesian filtering” notebook by Labbe Jr. [11] was used by the group as the main literature source in order to provide an intuitive introduction to Kalman filters. This particular source boasts a hands-on approach to filtering, using Python as a medium to implement the theory behind the Kalman filter. Additionally, it is easily accessible due to it being open-source. The book presents a clear and understandable flow of information, particularly in its introduction to the Kalman filter. The concept of Kalman filtering is introduced through α-β filters and α-β-γ filters, continuing further to discrete Bayes filters. The method of introduction is quite intuitive, presenting these filters as solutions to practical problems, and demonstrating their strengths, as well as their flaws in solving said problems. The Kalman filter is then presented as a solution to the given flaws from the previous algorithms. This method of presentation makes sense from a logical standpoint and builds a strong foundation for understanding the underlying mathematical framework behind the model. Based on this method of presentation, the Kalman filter has been described thoroughly in “Section 3.3”. The limitations of the filter are also explained, making way for the presentation of the Extended Kalman filter and the Unscented Kalman filter. Given that these implementations are not necessary for the project at this stage, however, it has been decided that they will not be included in this short literature review.

A very important point to note is the Kalman filter design chapter in the notebook [11]. In this chapter, a meaningful and theoretically simple evaluation criterion is presented, without the need for rigorous mathematical proof. This criterion has been used and discussed further in “Section 7.5”.

2.3 Discussion

When compared to a more classical textbook on Kalman filtering, such as the Kalman filtering textbook by Andrews and Grewal [4], there are many advantages to the chosen approach by the group. The lack of mathematical rigor makes the content easy to understand for beginners. The presentation of the problems and the solutions are also straightforward: a problem is presented, and a solution is carefully crafted to suit the specifications. This gives an intuitive explanation of the solution by building it in a procedural manner. These explanations resonate with an engineering mindset, where issues often appear during the construction of a solution, and thus require the engineer to modify the solution in order to solve said issues.

However, there are certain disadvantages to this approach. There are some parts of the explanations which are accepted without proof, and this leads to an incomplete understanding of the subject. This is one of the advantages of the textbook by Andrews and Grewal: it provides a full explanation of the underlying theory, providing derivations of the equations, as well as detailed performance metrics for the filter. In this regard, the notebook by Labbe Jr. falls short, as it is meant as an introductory book to Kalman filters. In order to gain a complete understanding of the Kalman filter, a more specialized textbook is needed.

Furthermore, the notebook by Labbe Jr. contains an example implementation of the Kalman Filter in Python. While this implementation is straightforward, it cannot be implemented in real-time, since the library framework in Python is quite large and can therefore not be implemented in embedded hardware. A C/C++-based implementation is a necessity in this case. The textbook by Grewal and Andrews has the implementations in MATLAB but with an additional chapter on implementation methods, which helps in compressing the mathematical framework into a library that can be used in a real-time application [4].

2.4 Conclusion

It is obvious that for the purposes of this project and the preliminary knowledge of the group regarding Kalman filters, the notebook by Labbe Jr. was an intuitive choice since it allows for a fast implementation of the Kalman filter while creating the basic intuition required to understand the filter's operation and performance metrics. However, other textbooks, such as the aforementioned Kalman filtering textbook by Andrews and Grewal [4] are necessary in order to provide a complete understanding of the subject.

3 Mathematical Background

In this section, the theoretical background for the project is described in detail. The section is divided into three main subsections:

- Mathematical model of hand grip mechanics.
- Parameter estimation of the model.
- Design of a Kalman Filter for state estimation.

The mathematical model section will provide an in-depth explanation of how muscle mass undergoes transitions between active, resting, and fatigued states. This model will serve as the fundamental basis for the other sections. The parameter estimation section will focus on the process of determining the best parameters for the model, based on how well they fit the gathered data. Moreover, the mathematical model will be used as a basis for the design of the Kalman Filter. Once the design of the filter is complete, the fitting parameters obtained in the parameter estimation section will be applied to the Kalman Filter.

3.1 Mathematical model

The mathematical model has been developed by Ross, Nigam and Wakeling [7], and is briefly explained in this section. It is based on the principle of compartmental modelling, where the different states are mapped as compartments of the entire physical model. In this part of the thesis, a 2-state model will be detailed.

3.1.1 2-state mathematical model

We use the following notation:

- m_a as the active muscle mass
- m_f as the fatigued muscle mass
- m_r as the resting muscle mass
- M as the total muscle mass

We assume that the total muscle mass can only be divided into these three states. This means that, at all times, the total muscle mass must be a sum of the fatigued, active, and resting muscle masses. This is shown in figure 1. Mathematically, the figure can be represented by:

$$m_a + m_f + m_r = M$$

By expressing the resting muscle mass as a function of the other variables, we can write:

$$m_r = M - m_a - m_f \tag{1}$$

In order to fulfill this model we need to write the differential equations describing the dynamics of the system. Here, a few assumptions for the system have been made:

- It is not possible for a fatigued muscle mass to become rested immediately. The opposite is not possible either.
- In order for a rested muscle mass to become active, brainpower has to be used. We assume this "control signal" is a boolean value of either 0 or 1. We denote brainpower as u.

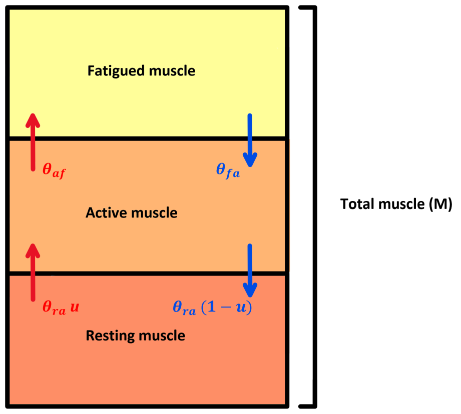


Figure 1: The model for muscle mass, and how u and the θ parameters affect the different muscle mass types

We denote all the parameters which characterize the transition from one state to another by θ:

- θ_{ra} represents the rate by which resting muscle mass becomes active
- θ_{af} represents the rate by which active muscle mass becomes fatigued
- θ_{fa} represents the rate by which fatigued muscle mass becomes active
- θ_{ar} represents the rate by which active muscle mass becomes rested

The measurements to be taken are denoted as z. They constitute the measurement model. Additionally, it is assumed by the sensor setup that only the active muscle mass can be measured. This setup is further detailed in "Section 4.1". The complete model, with both the process and the measurements, can then be written as:

$$\dot{m_a} = \theta_{ra} \cdot u \cdot m_r - \theta_{af} \cdot m_a + \theta_{fa} \cdot m_f - \theta_{ar} \cdot (1-u) \cdot m_a \tag{2}$$
$$\dot{m_f} = \theta_{af} \cdot m_a - \theta_{fa} \cdot m_f \tag{3}$$
$$z = m_a \tag{4}$$

Equation (2) can be combined with (1) in order to get:

$$\begin{aligned}\dot{m_a} &= \theta_{ra} \cdot u \cdot (M - m_a - m_f) - \theta_{af} \cdot m_a + \theta_{fa} \cdot m_f - \theta_{ar} \cdot (1-u) \cdot m_a \\ &= \theta_{ra} \cdot u \cdot M - \theta_{ra} \cdot u \cdot m_a - \theta_{ra} \cdot u \cdot m_f - \theta_{af} \cdot m_a + \theta_{fa} \cdot m_f - \theta_{ar} \cdot (1-u) \cdot m_a \\ &= (-\theta_{af} - \theta_{ar} - (\theta_{ra} - \theta_{ar}) \cdot u) \cdot m_a + (\theta_{fa} - \theta_{ra} \cdot u) \cdot m_f + \theta_{ra} \cdot M \cdot u\end{aligned}$$

Therefore, the equations describing the system dynamics can be written as:

$$\dot{m_a} = (-\theta_{af} - \theta_{ar} - (\theta_{ra} - \theta_{ar}) \cdot u) \cdot m_a + (\theta_{fa} - \theta_{ra} \cdot u) \cdot m_f + \theta_{ra} \cdot M \cdot u \tag{5}$$
$$\dot{m_f} = \theta_{af} \cdot m_a - \theta_{fa} \cdot m_f \tag{6}$$
$$z = m_a \tag{7}$$

These equations will be used further in the next sections.

3.1.2 Discrete 2-state mathematical model

In order to simulate this model, as well as simplify the parameter estimation part of the project, discretization via Euler's method can be used (forward Euler), due to its simplicity and adequate accuracy. The forward Euler's method is defined as:

$$y_{k+1} = y_k + h \cdot f(t_k, y_k) \tag{8}$$

where k is the sample number, h is the sampling time and $f(t_k, y_k)$ represents the differential equation to be discretized, namely, the right-hand side of equations (5) and (6). Applying forward Euler to these equations yields:

$$\begin{aligned}m_a[k+1] &= (1 - h \cdot \theta_{af} - h \cdot \theta_{ar} - (h \cdot \theta_{ra} - h \cdot \theta_{ar}) \cdot u[k]) \cdot m_a[k] \\ &\quad + (h \cdot \theta_{fa} - h \cdot \theta_{ra} \cdot u[k]) \cdot m_f[k] + h \cdot \theta_{ra} \cdot M \cdot u[k] \\ m_f[k+1] &= h \cdot \theta_{af} \cdot m_a[k] - (1 - h \cdot \theta_{fa}) \cdot m_f[k]\end{aligned}$$

In this model, the θ-parameters can be substituted with ϕ-parameters. The substitutions are:

$$\begin{aligned}\phi_{af} &= 1 - h \cdot \theta_{af} \\ \phi_{ar} &= h \cdot \theta_{ar} \\ \phi_{ra} &= h \cdot \theta_{ra} \\ \phi_{fa} &= 1 - h \cdot \theta_{fa}\end{aligned} \tag{9}$$

Therefore, the final discretized 2-state mathematical model is:

$$m_a[k+1] = (\phi_{af} - \phi_{ar} - (\phi_{ra} - \phi_{ar}) \cdot u[k]) \cdot m_a[k] + (1 - \phi_{fa} - \phi_{ra} \cdot u[k]) \cdot m_f[k] + \phi_{ra} \cdot M \cdot u[k] \tag{10}$$
$$m_f[k+1] = (1 - \phi_{af}) \cdot m_a[k] - \phi_{fa} \cdot m_f[k] \tag{11}$$

The measurement model remains unchanged from (7).

3.1.3 3-state mathematical model

The 3-state model uses the same notation as the model in "Section 3.1.1", with an additional state m_c, which denotes the cramped muscle mass. The introduction of this third variable means that both the static and dynamic models of the system must be updated. The total muscle mass is calculated as:

$$m_a + m_f + m_r + m_c = M$$

Equation (1) becomes thus:

$$m_r = M - m_a - m_f - m_c \tag{12}$$

The dynamic model reflects this change as well. In this model, it is not possible for the cramped muscle mass to become tired immediately. In addition, the cramped muscle mass cannot become active at will; it can only become rested. The ratio of this transition is denoted as θ_{cr} (for the transition from cramped to resting) and θ_{rc} (for the transition from resting to cramped). An active muscle mass can however become cramped. The ratio of this change is denoted as θ_{ac}. At the same time, the system of differential equations introduced in 3.1.1 gets a new state, namely m_c.

$$\dot{m_a} = \theta_{ra} u m_r - \theta_{af} m_a + \theta_{fa} m_f - \theta_{ar}(1-u) m_a - \theta_{ac} m_a \tag{13}$$
$$\dot{m_f} = \theta_{af} m_a - \theta_{fa} m_f \tag{14}$$
$$\dot{m_c} = \theta_{ac} m_a - \theta_{cr} m_c \tag{15}$$
$$z = m_a + b m_c \tag{16}$$

Equation (13) can be combined with (12) in order to get:

$$\begin{aligned}\dot{m_a} &= \theta_{ra} u (M - m_a - m_f - m_c) - \theta_{af} m_a + \theta_{fa} m_f - \theta_{ar}(1-u) m_a - \theta_{ac} m_a \\ &= \theta_{ra} u M - \theta_{ra} u m_a - \theta_{ra} u m_f - \theta_{ra} u m_c - \theta_{af} m_a + \theta_{fa} m_f - \theta_{ar}(1-u) m_a - \theta_{ac} m_a \\ &= (-\theta_{af} - \theta_{ar} - \theta_{ac} - (\theta_{ra} - \theta_{ar}) u) m_a + (\theta_{fa} - \theta_{ra} u) m_f - \theta_{ra} u m_c + \theta_{ra} M u\end{aligned}$$

The equations describing the system dynamics can be written as:

$$\dot{m_a} = (-\theta_{af} - \theta_{ar} - \theta_{ac} - (\theta_{ra} - \theta_{ar})u)m_a + (\theta_{fa} - \theta_{ra}u)m_f - \theta_{ra}um_c + \theta_{ra}Mu \tag{17}$$
$$\dot{m_f} = \theta_{af}m_a - \theta_{fa}m_f \tag{18}$$
$$\dot{m_c} = \theta_{ac}m_a - \theta_{cr}m_c \tag{19}$$
$$z = m_a + bm_c \tag{20}$$

Equations (5), (6), (7) represent the 2-state model for hand grip dynamics, while (17), (18), (19) and (20) represent the complete model with the cramped muscle mass. This model has not been discretized but will serve as a comparison further in the thesis, in "Section 7.1".

3.2 Parameter estimation

The model described in "Section 3.1.1" is a theoretical model, which comprises the aforementioned θ-parameters, the states, the control signal (mapped in our model as brainpower) and the measurement function. However, these equations do not contain any known θ parameters. The problem of parameter estimation therefore arises. In the following section, the least-squares estimator has been introduced as a solution to this problem.

3.2.1 Least-squares estimator

The estimator chosen for this application is the least-squares estimator. This estimator uses the difference between the measured data and the physical system's data to estimate the unknown parameters of the system. The parameters which minimize the sum of this difference squared are the estimated parameters we need. Mathematically, this estimation principle can be written as:

$$\theta^* = \arg(\min(\sum(y_{data} - y_{model}(\theta))^2)) \tag{21}$$

The estimator yields a parameter list θ^* that represents the model that best fits the data set. $y_{model}(\theta)$ is the model generated from our initialization parameters θ. y_{data} is the data set, received from the measurement model described by (7).

This problem and its solution have been detailed in "Section 5.1". The equations used in the parameter estimation algorithm are the same as those described in "Section 3.1.1". From this estimation problem, the parameters shall be calculated.

This estimator has been applied to the discretized mathematical models since they make the estimation problem easier to solve by setting clear upper and lower bounds for the parameters. Therefore, the estimation algorithm will be applied to the models described by equations (10) and (11) for the 2-state model. For this model, there are four parameters to be estimated.

Note that the identifiability analysis for the systems has not been made. Instead, the results from the previous work completed by the supervisors of this project have been reused. Due to these results, it is known that the 2-state system is globally identifiable, while the 3-state system is not globally identifiable. If the total muscle mass is unknown, then neither of the systems is globally identifiable. This knowledge will be further used in "Section 5".

3.2.2 Constraints for the system

As mentioned in "Section 3.2.1", the ϕ parameters in the discretized model are used, since the constraints for the least squares estimation problem can be calculated easily. These ϕ parameters cannot be negative, since having muscle masses that are negative does not make sense from a physical point of view. Therefore, both the states and the parameters must be positive.

If the ϕ parameters are negative, the rate of conversion between active and fatigued muscle masses does not make sense. In addition, in order to have a stable discrete-time system, these parameters must be smaller than 1. In order to enforce these parameters in a more explicit manner from a programming point of view, the Routh-Hurwitz criterion has been used. The reasons behind this decision are detailed in "Section 5.2".

Given a general second-order system, the Routh-Hurwitz criterion states that the system is stable if and only if the coefficients of the characteristic polynomial satisfy $a_i > 0$. In order to use this criterion, the system must first be linearized. By setting $u = 0$ and $u = 1$ in the system described by (10) and (11), we get the following equations for $u = 0$ and $u = 1$, respectively:

$$m_{a0}[k+1] = (\phi_{af} - \phi_{ar}) \cdot m_{a0}[k] + (1 - \phi_{fa}) \cdot m_{f0}[k] \tag{22}$$
$$m_{f0}[k+1] = (1 - \phi_{af}) \cdot m_{a0}[k] - \phi_{fa} \cdot m_{f0}[k] \tag{23}$$

$$m_{a1}[k+1] = (\phi_{af} - \phi_{ra}) \cdot m_{a1}[k] + (1 - \phi_{fa} - \phi_{ra}) \cdot m_{f1}[k] + \phi_{ra} \cdot M \tag{24}$$
$$m_{f1}[k+1] = (1 - \phi_{af}) \cdot m_{a1}[k] - \phi_{fa} \cdot m_{f1}[k] \tag{25}$$

From these equations, the following state space form can be written. The system matrices F_0 and F_1 are:

$$F_0 = \begin{bmatrix} \phi_{af} - \phi_{ar} & 1 - \phi_{fa} \\ 1 - \phi_{af} & -\phi_{fa} \end{bmatrix} \qquad F_1 = \begin{bmatrix} \phi_{af} - \phi_{ra} & 1 - \phi_{fa} - \phi_{ra} \\ 1 - \phi_{af} & -\phi_{fa} \end{bmatrix} \tag{26}$$

The input vector B is:

$$B = \begin{bmatrix} \phi_{ra} \cdot M \\ 0 \end{bmatrix} \tag{27}$$

Finally, given that our measurements only include the active muscle mass m_a (as shown from (7)), the measurement matrix is:

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{28}$$

In order to find the characteristic polynomial, the transfer function for these 2 systems must be found. Given the state-space model for a generalized LTI system:

$$\dot{x} = F \cdot x + B \cdot u$$
$$z = H \cdot x + D \cdot u$$

the transfer function can be calculated as:

$$G(s) = H \cdot (s \cdot I - F)^{-1} \cdot B + D \tag{29}$$

The proof for this equation is trivial: we merely take the Laplace transformation of the model for the generalized LTI system, and find the ratio $\frac{y(s)}{u(s)}$. Setting these expressions in MATLAB as symbolic variables, we get the following transfer functions for $u = 0$ and $u = 1$:

$$G_0(s) = \frac{M \cdot \phi_{ra} \cdot (\phi_{fa} + s)}{s^2 + (-\phi_{af} + \phi_{fa} + \phi_{ar}) \cdot s + \phi_{af} + \phi_{fa} - 2 \cdot \phi_{af} \cdot \phi_{fa} + \phi_{fa} \cdot \phi_{ar} - 1} \tag{30}$$

$$G_1(s) = \frac{M \cdot \phi_{ra} \cdot (\phi_{fa} + s)}{s^2 + (-\phi_{af} + \phi_{fa} + \phi_{ra}) \cdot s + \phi_{af} + \phi_{fa} + \phi_{ra} - 2 \cdot \phi_{af} \cdot \phi_{fa} - \phi_{af} \cdot \phi_{ra} + \phi_{fa} \cdot \phi_{ra} - 1} \tag{31}$$

Note that these are the open-loop transfer functions. In order to use the Routh-Hurwitz criterion, the closed-loop transfer functions are needed. Remember that generally, the closed loop transfer function is related to the open loop transfer function as follows:

$$G_{cl}(s) = \frac{G_{sys}(s) \cdot G_{reg}(s)}{1 + G_{sys}(s) \cdot G_{reg}(s) \cdot G_{filter}(s)}$$

It is assumed there is no filter or regulator in the loop since we only need the stability of the system. Thus, the equation above reduces to:

$$G_{cl}(s) = \frac{G_{sys}(s)}{1 + G_{sys}(s)}$$

Given that the transfer function for the system can be reduced to a rational function ($G_{sys}(s) = \frac{n(s)}{d(s)}$, the closed loop function can be further reduced:

$$G_{cl}(s) = \frac{n(s)}{n(s) + d(s)} \tag{32}$$

The characteristic function is by definition:

$$n(s) + d(s) = 0 \tag{33}$$

Using equations (32) and (33), we get the following characteristic equations for our system:

$$\begin{aligned} P_0(s) = s^2 + (-\phi_{af} + \phi_{fa} + \phi_{ar} + M \cdot \phi_{ra})s + \phi_{af} + \phi_{fa} \\ - 2 \cdot \phi_{af} \cdot \phi_{fa} + \phi_{fa} \cdot \phi_{ar} + M \cdot \phi_{fa} \cdot \phi_{ra} - 1 \end{aligned} \tag{34}$$

$$\begin{aligned} P_1(s) = s^2 + (-\phi_{af} + \phi_{fa} + \phi_{ra} + M \cdot \phi_{ra})s + \phi_{af} + \phi_{fa} + \phi_{ra} \\ - 2\phi_{af} \cdot \phi_{fa} - \phi_{af} \cdot \phi_{ra} + \phi_{fa} \cdot \phi_{ra} + M \cdot \phi_{fa} \cdot \phi_{ra} - 1 \end{aligned} \tag{35}$$

Both these systems need to be stable. Therefore, all the coefficients of the polynomials must be larger than zero. This means that the constraints can be formulated as:

$$-\phi_{af} + \phi_{fa} + \phi_{ar} + M \cdot \phi_{ra} > 0 \tag{36}$$

$$\phi_{af} + \phi_{fa} - 2 \cdot \phi_{af} \cdot \phi_{fa} + \phi_{fa} \cdot \phi_{ar} + M \cdot \phi_{fa} \cdot \phi_{ra} > 1 \tag{37}$$

$$-\phi_{af} + \phi_{fa} + \phi_{ra} + M \cdot \phi_{ra} > 0 \tag{38}$$

$$\phi_{af} + \phi_{fa} - 2 \cdot \phi_{af} \cdot \phi_{fa} - \phi_{af} \cdot \phi_{ra} + \phi_{fa} \cdot \phi_{ra} + M \cdot \phi_{fa} \cdot \phi_{ra} > 1 \tag{39}$$

3.3 Kalman Filter

In order to design and implement a Kalman filter for this system, we need to extend the model described by (10), (11) and (7). This extension is composed of noise components, namely process noise (or disturbance) and measurement (sensor) noise. These are denoted w_1, w_2 and v, respectively, and we assume that:

$$w_1 \sim \mathcal{N}(0, \sigma_{w_1}^2)$$
$$w_2 \sim \mathcal{N}(0, \sigma_{w_2}^2)$$
$$v \sim \mathcal{N}(0, \sigma_v^2)$$

The model can then be written as:

$$\begin{aligned} m_a[k+1] &= (\phi_{af} - \phi_{ar} - (\phi_{ra} - \phi_{ar}) \cdot u[k]) \cdot m_a[k] \\ &\quad + (1 - \phi_{fa} - \phi_{ra} \cdot u[k]) \cdot m_f[k] + \phi_{ra} \cdot M \cdot u[k] + w_1 \end{aligned} \tag{40}$$

$$m_f[k+1] = (1 - \phi_{af}) \cdot m_a[k] - \phi_{fa} \cdot m_f[k] + w_2 \tag{41}$$

$$z[k] = m_a[k] + v \tag{42}$$

The Kalman filter consists of 3 steps: initialization, the prediction step and the update step. These steps are explained in detail below.

In the initialization step, an arbitrary value is given to $\hat{x}$ and $\hat{P}$, in order to be able to compute the prediction step. The chosen values are:

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \hat{P} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

These values initialize the filter, making the computation of the prediction step possible. This initialization, that has been chosen, assumes that the user is fully rested. During this step, a prediction will be calculated. This prediction is based on the knowledge we possess regarding the system. Our knowledge of the system can be expressed through the following 3 matrices:

- **The system matrix** F. This matrix represents the evolution of the system in time, given that there are no external inputs to the system.
- **The input matrix** B. This matrix represents the influence of inputs in the system; in other words, how external inputs influence the dynamics of our model.
- **The process noise matrix** Q. This matrix consists of the variance for each state, as well as the covariances between the states. For the system described by (10), (11) and (7), it is defined as:

  $$Q = \begin{bmatrix} \sigma_{w_1}^2 & \mathrm{Cov}(w_1, w_2) \\ \mathrm{Cov}(w_2, w_1) & \sigma_{w_2}^2 \end{bmatrix} \tag{43}$$

  An easier way to think of this matrix is as if it were our "belief" in the system. This means that, if the Q matrix has large values, the system is not trustworthy, and the predictions are not accurate.

These 3 matrices can be used to create the prediction block as shown below:

$$\bar{x} = F \cdot \hat{x} + B \cdot u \tag{44}$$

$$\bar{P} = F \cdot \hat{P} \cdot F^T + Q \tag{45}$$

Equation (44) uses the system matrix and the input matrix to generate a prediction $\bar{x}$ for the next time step. This prediction is normally distributed, which means a covariance matrix must also be calculated. This is what is computed through equation (45). This equation computes the "uncertainty" in our system by projecting the state covariance matrix $\hat{P}$ into the model space and then adding the process noise on top of that. Note that the predictions are generated in the model space. Both $\bar{x}$ and $\bar{P}$ from the equations above serve as the input for the update block, which is described next.

The update block uses the measurement model to improve upon the predictions generated by the prediction block. In order to do this, the measurement model must be described fully. This is achieved through the following two matrices:

- **The measurement matrix** H. This matrix represents the effect of the states on the measurements that are taken.
- **The measurement noise matrix** R. Similarly to the Q matrix described by (43), this matrix is also a covariance matrix. However, contrary to the Q matrix, it represents the measurement uncertainty. For the system described by (10), (11) and (7), it is defined as:

$$R = \sigma_v^2 \tag{46}$$

In order to fully convey the idea behind update block, a step-by-step explanation has been used. The first step in the update block is defining the residual. By definition, the residual is the difference between the measurement and the prediction. It is expressed by the following equation:

$$y = z - H \cdot \bar{x} \tag{47}$$

Note that the prediction must first be projected into the measurement space in order to compute the residual, given that the update is calculated in measurement space, while the prediction is calculated in the model space. In addition, given that this residual is a difference between normally distributed variables, a covariance matrix needs to be computed in order to express the uncertainty in the residual. This covariance matrix is termed the innovation covariance matrix and is defined as:

$$S = H \cdot \bar{P} \cdot H^T + R \tag{48}$$

This equation bears a striking resemblance to equation (45). The difference is that here, instead of the estimates being projected in the model space, it is the prediction that is projected in the measurement space, and then getting the measurement noise added on top.

The most important part of the update block, the Kalman gain, is explained next. This gain serves as a weighting factor which decides whether the measurements, the predictions from the model or a linear combination of both are to be trusted. It is mathematically defined as:

$$K = \bar{P} \cdot H^T \cdot S^{-1} \tag{49}$$

Now that the Kalman gain is calculated, the updated estimates can be computed. We start by computing the state estimate mean. This needs to be a value between the state prediction and the measurement. The equation can be written as:

$$\hat{x} = \bar{x} + K \cdot (z - H\bar{x}) \tag{50}$$

It is obvious here that the Kalman gain K serves as a weighting factor. If we consider the scalar case, where $\bar{x}$ consists of one prediction, z consists of one measurement and $H = 1$, we can write:

$$\hat{x} = \bar{x} + K \cdot (z - \bar{x})$$
$$\hat{x} = (1 - K) \cdot \bar{x} + K \cdot z$$

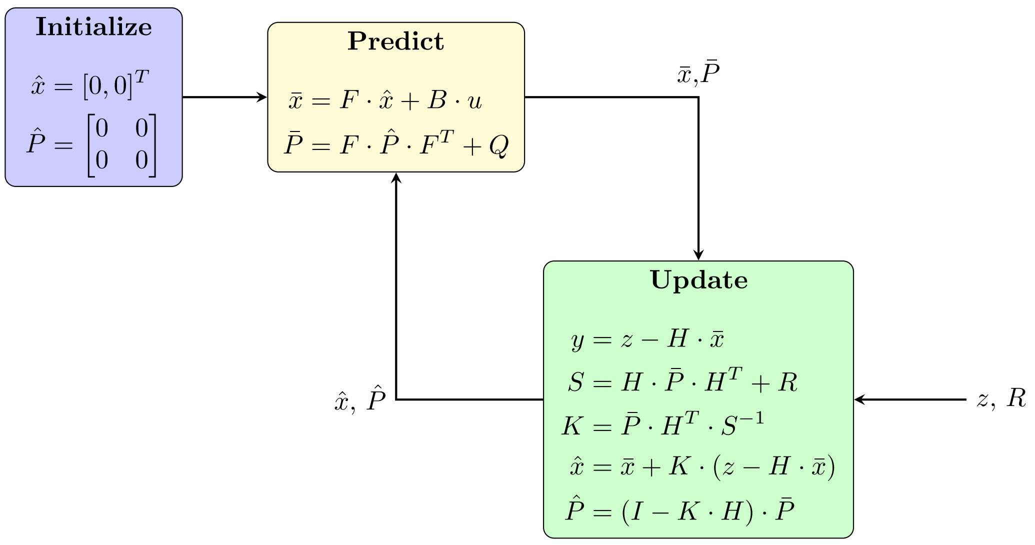
As we can see, for $K = 1$, the state estimate is the same as the measurement. The filter eliminates the predictions from the computed estimates. Conversely, for $K = 0$, the state estimate consists solely of the prediction, and the filter removes the measurements from the estimates. This logic is valid for the multivariable case as well. The only difference is that vectors have to be used instead of scalars, and the Kalman gain becomes a vector as well.

The state estimate $\hat{x}$ is normally distributed. This means that the state covariance matrix must also be computed. The equation to compute this matrix is given by:

$$\hat{P} = (I - K \cdot H) \cdot \bar{P} \tag{51}$$

Equation (51) is the final equation of the update block. The estimate mean $\hat{x}$ generated by (50) and the estimate covariance matrix $\hat{P}$ generated by (51) are the outputs we are looking for since they describe the system fully. Remember that the measurement model only includes the active muscle mass; the fatigued muscle mass cannot be measured. However, the Kalman filter can generate estimates for the fatigued muscle mass based on the system's model, effectively giving us the complete information we need from all the states.

The figure below is a diagram showing how the Kalman filter algorithm works step by step:



Initialize
$\hat{x} = [0,0]^T$
$\hat{P} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
Predict
$\bar{x} = F \cdot \hat{x} + B \cdot u$
$\bar{P} = F \cdot \hat{P} \cdot F^T + Q$
$\bar{x}$,$\bar{P}$
Update
$y = z - H \cdot \bar{x}$
$S = H \cdot \bar{P} \cdot H^T + R$
$K = \bar{P} \cdot H^T \cdot S^{-1}$
$\hat{x} = \bar{x} + K \cdot (z - H \cdot \bar{x})$
$\hat{P} = (I - K \cdot H) \cdot \bar{P}$
$\hat{x}$, $\hat{P}$
z, R

Figure 2: Kalman filter algorithm.

3.3.1 The Joseph equation

Given the discrete nature of the Kalman filter that is being implemented, as well as the hardware it is being implemented on, some measures have to be taken regarding the floating-point approximation from the floating-point unit (FPU) in the circuit. Given the nature of the approximations happening in the FPU, failsafes need to be designed in order to ensure that these approximations do not influence the performance of the filter. One of the aspects where these failsafes need to be implemented is the state covariance matrix $\hat{P}$ with dimensions 2×2. This matrix consists of the state variances in the primary diagonal and the covariances in the secondary diagonal. It is obvious that this matrix must be symmetric (the covariances must be equal since they describe the variance between the same 2 states). If these matrices are not symmetric, the filter shall diverge. Thus, it is important to ensure that the state covariance matrix $\hat{P}$ is symmetric. In order to ensure this, the Joseph equation is used, and (51) is transformed into:

$$\hat{P} = (I - K \cdot H) \cdot \bar{P} \cdot (I - K \cdot H)^T + K \cdot R \cdot K^T \tag{52}$$

This equation beholds the symmetrical nature of both $\bar{P}$ and $\hat{P}$. The proof for this equation can be found in the "Kalman and Bayesian filtering" notebook [11]. Equation (52) has been used instead of equation (51) further in the project thesis.

4 Hardware setup and sensor design

4.1 Electronics

The electronic circuit consists of:

- GHF-10 - UNEO Inc.
- TL074CN - Texas Instruments
- ESP32 TTGO t-display - LILYGO
- 47kΩ resistor
- 9v battery

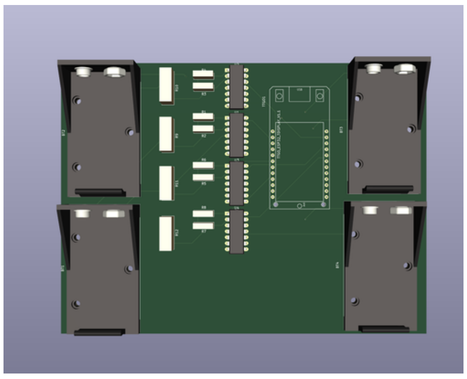
The pressure sensor, a GHF-10 by UNEO inc, was chosen for its relatively cheap price, and its desirable operating force, from 0kg to 50kg. There is a linear relationship between pressure and voltage, according to the sensor documentation [12] when the sensor is paired with an OP-amp. Hence, selecting an OP-amp circuit for the measurement circuit was a logical decision.

It was opted to utilize the TL074CN OP-amp (Operational Amplifier), manufactured by Texas Instruments, owing to its four-in-one capability which was deemed suitable for the intended purpose. The first design was tested with only a single sensor, and it was a success. However, it was found, after integrating the rest of the sensors, that it is essential for every sensor to have its own OP-amp and external power source. Due to the shared voltage source and ground of the TL074CN, only one OP-amp in the package could be used at a time for this project. The decision to persist with the TL074CN was based on the fact that these components had already been acquired in the beginning phases of the project, hence it was deemed practical to use them.

The resistor is paired with the OP-amp to form an inverting OP-amp. In this setup, a 47KΩ resistor has been determined to be a favourable balance between measurement range and noise reduction.

To provide power for each OP-amp, 9V batteries were selected as the power source. This choice is attributed to their affordability and ease of integration within the circuit.

The ESP32 TTGO t-display breakout board from LILYGO was selected as the microcontroller for its expandable features, including WiFi, Bluetooth, USB-C, and integrated screen, which is useful for quick visualization and confirmation of functionality during testing.


(a) Circuitboard

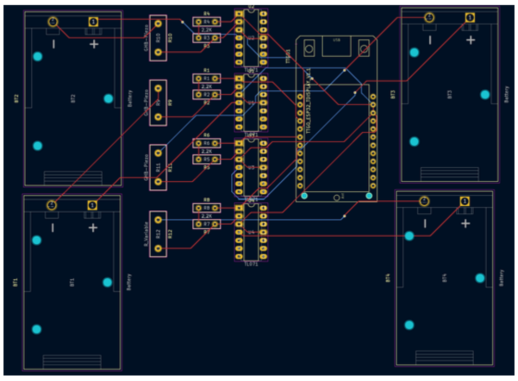

(b) Circuitboard traces

Figure 3: The PCB design for the circuit. PCB includes all necessary components

The PCB (Printed Circuit Board) was designed accordingly, to accommodate the four 9v batteries and OP-amp packages, along with the ESP32 TTGO and resistors. Two resistors in series were used for each OP-amp at some time during testing, but only one resistor is used in the final design. There are still remnants of the series resistor on the PCB, this can easily be fixed by bridging one of the resistor spots for each OP-amp with a wire.

4.2 Previous device

As highlighted in “Section 2”, the availability of literature on handgrip strength measurement devices is limited. Therefore this thesis is a continuation of the design described in the “AME Project” [9], which will be briefly discussed in this section.

The starting point for this thesis is “Prototype 2” developed within the “AME Project”. This prototype comprises a 3D-printed cylinder with three equally spaced cut-outs. These cut-outs serve the purpose of accommodating a balloon obtained from a blood pressure cuff. By threading the balloon through these cut-outs, the overall size of the device can be adjusted. When the device is squeezed, the pressure inside the balloon changes, which can be measured. This pressure variation is expected to exhibit a correlation with handgrip strength.

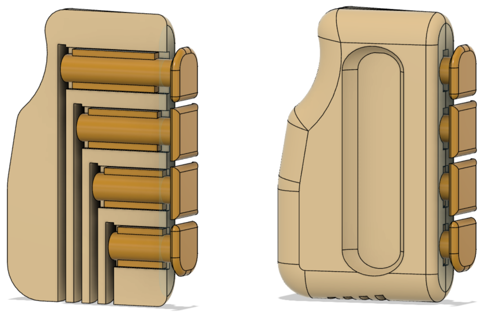
Although the prototype serves as the initial foundation for this thesis, it is important to highlight that the device has undergone several modifications and iterations. These changes have been made to improve functionality and performance. In “Section 7.3” of this thesis, a detailed account of the different iterations and enhancements made to the device will be provided, outlining the evolution and refinement of the handgrip trainer design.

4.3 Physical design

The physical model was created in Autodesk Fusion 360, a 3D modeling and design program, and was inspired by standard spring-loaded handgrip trainers. The model was made to have 4 individual intuitive pressure points acting like large pogo pins (spring loaded pins) shown in figure 4. Sensors were placed behind the pins so that any press on the pins would lead to a sensor output. The design iterations, and the desitions behind them will be discussed in “Section 7.3”.

The sensors are inserted from the holes on the bottom of the model. There are four holes, one for each sensor. The one furthest to the front of the model is for the bottom sensor, and the one furthest from the front is the top sensor. The holes are sized to fit the breadboard (prototyping board) wires connecting the sensor to the circuit.

To provide the pins with progressive resistance and to distribute the pressure across the entire sensor, rubber foam was inserted between the pins and the sensor. This technique results in a consistent and dependable output from the sensor.



(a) Trainer model (b) Cross section

Figure 4: The final handgrip trainer. This model uses pogo pins to press the sensors embedded in the device.

4.4 Kalman Filter implementation

The code for the ESP32 TTGO consists of mainly the filter code. This code was implemented using the struct class. This was to make multiple filters for all the sensors, easier and cleaner.

The following code snippet presents a non-functioning simplified version of the filter code, designed to enhance readability. To find the full script, please check the "Appendix".

Listing 1: Simplified Kalman Filter structure

```
struct KalmanFilter {
    // Sensor setup
    int pin;
    int reading;

    // phi parameters
    const float M;
    const float phi_af;
    const float phi_ar;
    const float phi_ra;
    const float phi_fa;

    struct state {
        P<2, 2>;
        x<2, 1>;
    };

    // initializing matrixes
    state prediction;
    state estimate;

    F<2, 2>;
    y<1, 1>;
    z<1, 1>;
    S<1, 1>;
    K<2, 1>;
    const Q<2, 2>;
    const R<1, 1>;
    const H<1, 2>;
    const B<2, 1>;
    const I<2, 2>;

    void read()      {}

    void predict()   {}

    void update()    {}

    void readPredUpd()   {}
};
```

The only variable in the filter code that needs changing after filter initialization is the pin variable. This variable tells the filter which pin to read the sensor data from.

All of the ϕ parameters will remain constant for all the filters. They are therefore declared as such, as shown in the code snippet. The total muscle mass M is also a constant and is declared this way. These ϕ parameters would be replaced if new and better parameters are found.

Next, all the matrices are initialized. All of the matrices are declared, and the constant matrices are given their correct values. The prediction and estimate matrices are declared through a structure, to easily distinguish the different x and P matrices.

The filter code consists of four functions. The first is for reading the sensor. The second is for the filter's prediction block. The third is for the filter's update block. The last one runs the other three in their correct order. This means that it is only necessary to call upon the last function to update the filter output.

Listing 2: The Kalman Filter prediction block. This function depicts the difficulties of predicting u.

```
void predict() {
    bool flagDer = (z - z_prev)/sampletime > derThresh;
    bool flagLow = z > threshold;

    if (flagDer and flagLow) {
        F = FPressed;
        prediction.x = (F * estimate.x) + (B);
    }

    else {
        F = FReleased;
        prediction.x = (F * estimate.x);
    }

    prediction.P = (F * estimate.P * ~F) + (Q);
}
```

There are two different F - matrices for the system, one for activating the trainer (when $u = 1$), and one for the released trainer (when $u = 0$). The code snippet to the left shows how the models are switched. There is firstly a check for the derivative of the measurements and if it's bigger than a certain threshold. The threshold is a relatively low negative value, to filter out measurement noise. The second check is if the measured value is bigger than some threshold. These checks decide if we use the activation F - matrix, or the released F - matrix. They are also used as a replacement for u.

The remaining code primarily serves the purpose of debugging. A significant portion of it is dedicated to displaying sensor data on the screen attached to the ESP32 TTGO. This code generates a plot of the filtered sensor output, making it more convenient to verify the proper functioning of all sensors.

5 Experiments

5.1 Parameter estimation

As mentioned in "Section 3.2", a least squares estimator was used for estimating the parameters of the system. In this section, the data logging pipeline, as well as the MATLAB script running the estimator is detailed.

5.2 The parameter estimation pipeline

In order to run the least-squares estimator, data from the sensors must be logged. The solution for this is writing said data to the serial interface between the ESP32 TTGO development board and the computer running the simulations. Then, a Python script which is run from the computer side of the pipeline reads this data and saves it in a CSV file. The CSV file is then read by the MATLAB script running the parameter estimation algorithm. This entire flow of information is assured by 3 scripts:

- The *Pressure_sensor_test/src/main.cpp* script which receives the data from the sensor array. This script has, in addition to the serial interface data write, a moving average filter which reduces the noise from the sensor array. For the tests, a sample time of 10 ms was used. In some of the tests, this script has been substituted with the final *bachelor_esp_kalman/src/main.cpp* script which contains the full implementation of the Kalman filter.

- The *import_data_time_pls.py* script which covers the data receiving side. It has a timed loop where the serial data is written as a string with different separators (commas for different sensor measurements and semicolon for new sensor measurements at every sample period).
- The *fit_disc_ode_parameters.m* script which covers the MATLAB side of the CSV file. This file structures the data as a vector of measurements to be read. It then structures the objective function as the sum of the squared difference between an optimization expression and the data set. In this case, the optimization expression is the model described by (10) and (11), together with the measurement function described by (7). This objective function can then be solved for the ϕ-parameters that yield the least sum of the squares.

The ESP32 TTGO script which writes the data to the serial port and the Python script which reads the data from the serial port and structures it as a .csv file are quite straightforward. They are not detailed in this thesis but are a part of the Git repository attached in the "Appendix". Within the MATLAB script, however, there is a code section which explains the usage of the Routh-Hurwitz criterion from "Section 3.2.2". This code section is explained in the section below:

5.2.1 Parameter estimation code

The code that is described below is found in the *fit_disc_ode_parameters.m* script in the Git repository attached in the "Appendix". Through this code, parameter estimation is possible. We start by explaining the functions which are used by this script:

- The *disc_diff_eq.m* function file runs the mathematical model described by (10) and (11). It returns two vectors, where one has the values of the active muscle mass and the other has the values of the fatigued muscle mass.
- The *disc_theta_to_ode.m* function file runs the same mathematical model, but instead of using numerical values for the ϕ-parameters, it expresses them as optimization variables. This function will be used to build the expression to be optimized. It returns a vector of optimization expressions which gets compared to the data set that was logged from the hardware. An important note here is that this function only returns the optimization expressions which correspond to the active muscle mass; the fatigued muscle mass cannot be measured.

The code snippets below have been extracted from the script file, based on their importance to the estimator algorithm.

```
    %Starting point for the optimization problem
    phi_first_guess = [0.5, 0.5, 0.5, 0.5];
    % Grid for the total muscle mass
    M_size = 100;
    M = linspace(3, 40, M_size);
```

Listing 3: Initialization for the parameter estimation

The first line defines a starting point for the optimization problem. The reasoning behind this choice is further explained in "Section 7". The second and third lines set up the grid search by defining the total muscle mass as a number between 3 and 40. The size of the grid can be chosen arbitrarily: in our project, the grid size is 100 elements.

```
    % Defining the optimization variables
    phi = optimvar('phi', 4);
    % Preallocating memory for speed
    optim_y = optimexpr(1, N);
    sumsq = NaN(1, M_size);
    phi_estims = NaN(4, M_size);
```

Listing 4: Optimization variable setup and preallocation

Then, the optimization variables are defined. These are the ϕ-parameters which must be estimated for the logged data set. In order to make the code run faster, the memory for the computations has been preallocated. This is the purpose behind the three last lines in the snippet above: they allocate a vector of optimization expressions, a vector which registers the different sums of the squares for each muscle mass in the grid, and four different vectors holding the different optimal ϕ-parameters for each muscle mass in the grid.

```
for i = 1:M_size
    fprintf('Problem %i', i);
    %Building the optimization expression from the
    %disc_theta_to_ode function
    optim_y = fcn2optimexpr(@disc_theta_to_ode, phi, N, u_vec, M(i)
   );
    %Defining the objective function as the least squares method.
    obj = sum((y_data - optim_y).^2);
    %Defining the optimization problem
    %Tweaking the fmincon solver to evaluate the objective function
    %up to 9000 times
    opts = optimoptions(@fmincon, 'MaxFunEvals', 9e3);
    %Defining the optimization problem
    prob = optimproblem('Objective', obj);
    %Defining the constraints in order to force optimproblem
    %to use fmincon instead of lsqnonlin.
    prob.Constraints.cons1 = -phi(1) + phi(4) + phi(2) + ...
    (M(i)*phi(3)) >= 0.0001;
    prob.Constraints.cons2 = phi(1) + phi(4) - (2*phi(1)*phi(4)) +
   ...
    (phi(4)*phi(2)) + (M(i)*phi(4)*phi(3)) >= 1.0001;
    prob.Constraints.cons3 = -phi(1) + phi(4) + phi(3) + ...
    (M(i)*phi(3)) >= 0.0001;
    prob.Constraints.cons4 = phi(1) + phi(4) - (2*phi(1)*phi(4)) -
   ...
    (phi(1)*phi(3)) + (phi(4)*phi(3)) + (M(i)*phi(4)*phi(3)) >=
   1.0001;
    %Setting up the start point for the optimization problem
    phi_0.phi = phi_first_guess;
    %Solve the optimization problem via fmincon
    [phi_sol, sumsq(i)] = solve(prob, phi_0, 'Options', opts);
    %The optimal parameters are assigned to the optimal parameter
    %matrix for each iteration of the grid search
    phi_estims(:, i) = phi_sol.phi;
end

[min, min_index] = min(sumsq);
```

Listing 5: Grid search algorithm

In this code snippet, the estimator is defined. For each total mass in the grid, the optimization expression is built, as well as the objective function. The optimization problem is built around the objective function, through the usage of the *optimproblem* function. In addition, constraints have been defined for the problem. Since *optimproblem* chooses the solver for the optimization problem based on the form of the objective function, not defining the constraints poses a problem, since the solver will be implicitly changed by MATLAB. The solver used by the script without any explicit constraints is *lsqnonlin*. This solver does have its own constraints, but they do not necessarily match the stability criteria of the system. After the grid search is completed, the smallest element in the sum of squares vector is found, as well as its location in said vector. This smallest element in the sum of squares vector represents the ϕ-parameters and the total mass M which best fits the model to the data. In layman's terms, this means that we are looking for the parameter list and the total muscle mass that generates the least amount of deviation from the data. We then assume that these parameters are the ϕ-parameters and the total muscle mass M we seek.

5.2.2 Least squares estimator testing

Twelve tests were conducted by the project members, and 12 more tests were conducted by volunteers in order to check for bias in data collection and processing. The aim of the tests was to find estimates under different conditions. Therefore, the test duration varies from 2 minutes to 6 minutes. In order to fully characterize muscle fatigue, the contraction time was kept consistently over 30 seconds. If the contraction time was any lower, there would not be a clear fatigue trend in the data. The relaxation time was kept consistently over 30 seconds as well, in order to Additionally, the tests require the subjects to be fully relaxed at the test start. If they are not relaxed, the fatigued muscle mass will increase faster in the data, yielding incorrect results. The test bench used was the one described in "Section 4.1" and "Section 4.3".

The tests shown below are only a few of the tests that have been logged during the period the thesis work was conducted. The Git repository attached in the "Appendix" includes all the tests that were run, as well as the resulting parameters from each test.

Design tests The estimator tests presented below were conducted by the project group. Only 3 tests have been included in this thesis, the remaining 9 design tests are available in the Git repository in the "Appendix".

The estimation test in figure 5 was one of the first tests conducted. The data set that was logged is *test1.csv*. A moving average filter with a size of 20 was used to remove some of the noise from the measurement circuit. The least squares estimator managed to fit the data to the model very well in this test case.

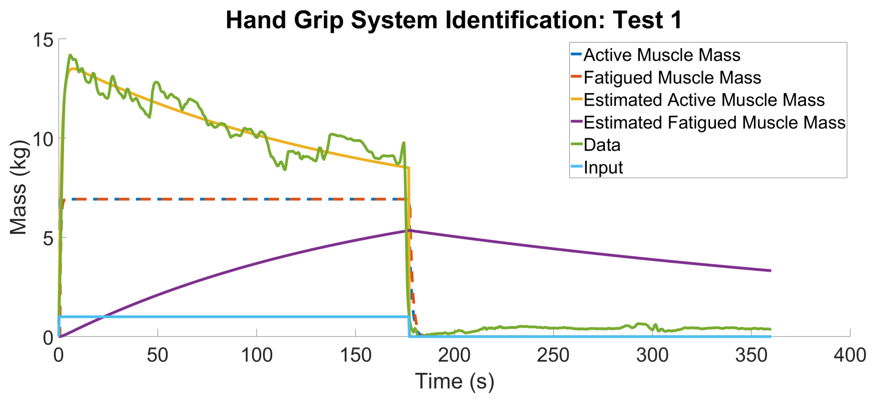


Figure 5: Data from test 1

Another test that was conducted by the group was *test2.csv*, shown in figure 6. In this test, the moving average filter was not used. This was done in order to see how the algorithm is influenced by noisy data. In this specific case, the estimator managed to fit the data into the model, albeit at a much worse rate. This is reflected in the least sum of squares, which is orders of magnitude higher than what was observed in the first test.

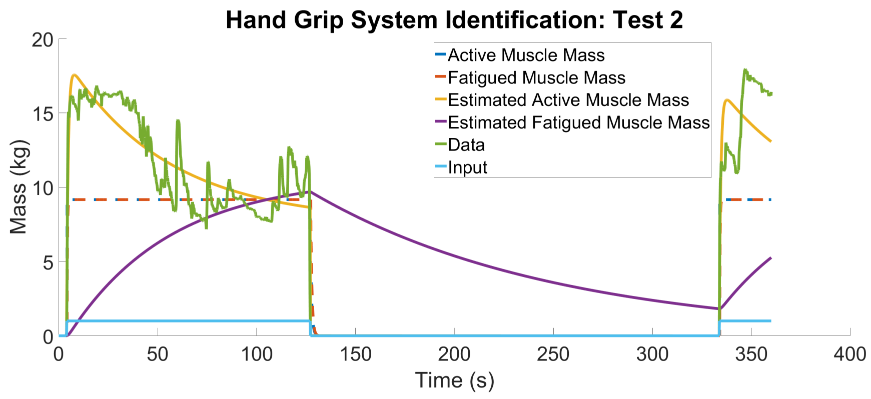


Figure 6: Data from test 2

Lastly, the data set logged in *test12.csv* is included in this preview of the tests. A moving average filter was used in this test as well, with a window size of 20, but here we notice that the estimator did not fit the data to the model. The reasons behind this breakdown in the algorithm, as shown in figure 7 will be further discussed in "Section 7".

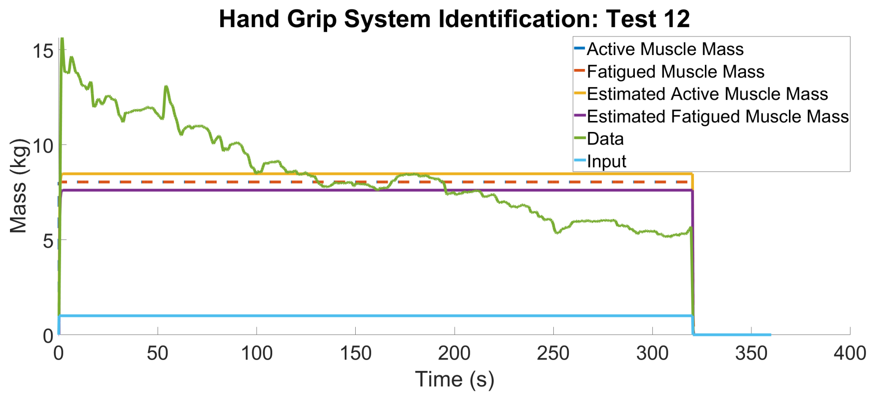


Figure 7: Data from test 12

Bias tests In addition to the tests above, 12 additional tests were conducted in order to check our data collection for potential bias. The users did not have any prior knowledge of the theory behind the system. In this thesis, only 3 tests have been included. The rest of the tests are in the Git repository attached to this thesis in the "Appendix", together with the associated parameter lists.

An important point to note is that these tests were not run with the moving average filter. Instead, the implementation of the Kalman filter described in "Section 4.4" was used. The parameter list inserted in the prediction block of the Kalman filter was the list from *test2.csv* since this parameter list yielded the best noise filtering.

The first bias test presented in figure 8 was one of the cases where the estimator reflected the trends in the data. This can be noticed by the constant drop in the active muscle mass during the test. However, the data is not fitted properly to the model due to noise. The fatigued muscle mass is also at reasonable levels.

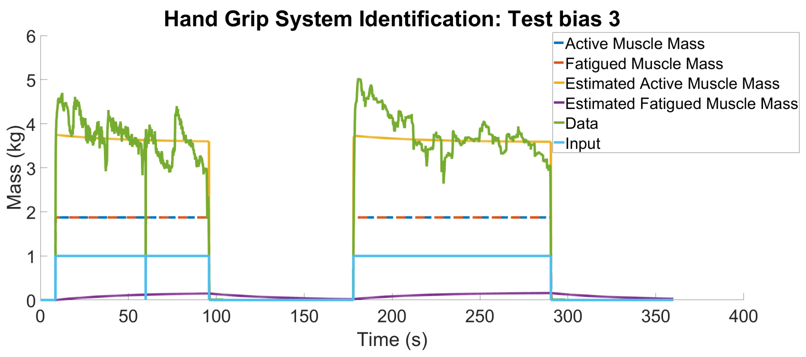


Figure 8: Data from unbiased user test 3

Another test is shown in figure 9. During the switch from contraction to resting, a disconnection happened in the hardware. However, setting this data in the estimator yields interesting results. The active muscle mass is fitted perfectly to the data, but the fatigued muscle mass does not make sense from a physical point of view.

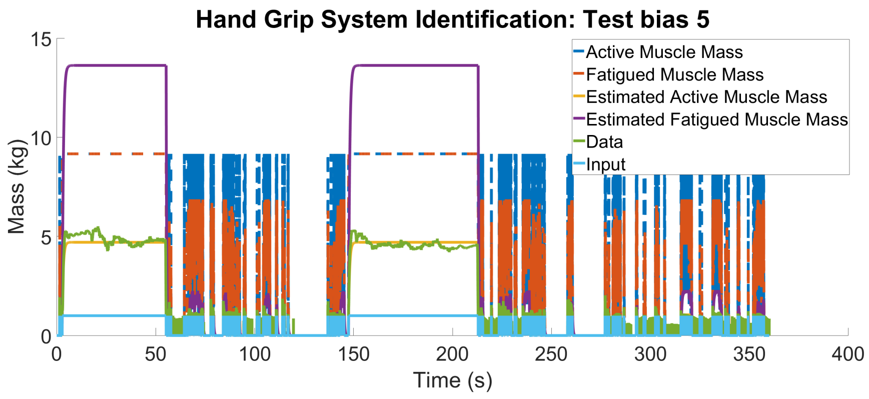


Figure 9: Data from unbiased user test 5

Lastly, the test shown in figure 10 shows another breakdown in the algorithm, in the same way as in figure 7. The estimator does not fit the data to the model, but instead returns a parameter combination that does not make sense. The reasons for this breakdown are identical to the one in figure 7 and are further elaborated upon in “Section 7”.

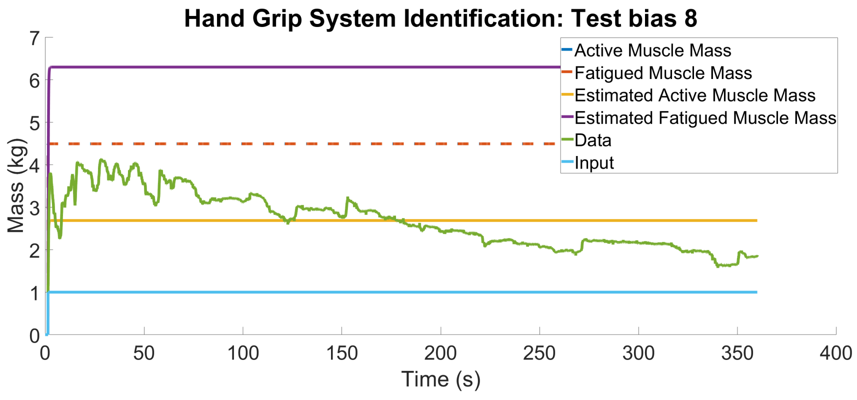


Figure 10: Data from unbiased user test 8

6 Results

The depicted circuit in figure 11 represents a functional proof-of-concept for the PCB design. It is connected in the same manner as the actual PCB wiring. However, it is worth noting that there are certain challenges encountered when working with a breadboard setup, which will be elaborated upon in “Section 7.2”. Despite these challenges, the circuit successfully establishes the desired linear relationship between pressure and voltage, validating its intended functionality.

The sensors are incorporated into the packaging of the device, in the final design of the 3D printed model, as shown in figure 11. This design choice provides the user with an intuitive grip and pressure points for squeezing. However, it is important to note that the device falls short of its intended ergonomic goals. The reasons behind this will be elaborated upon in “Section 7.3” of the thesis.

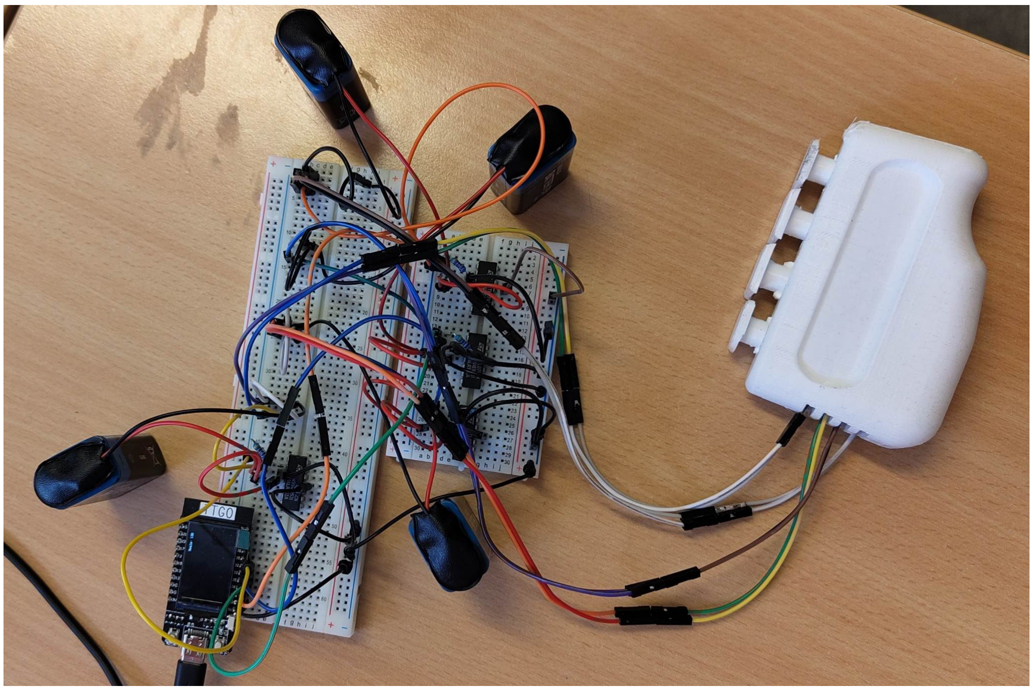


Figure 11: The physical setup

The circuit and handgrip trainer, depicted as the “ESP Pipeline” in figure 12, operate independently from the “Parameter Estimation Pipeline”. The sole purpose of the “Parameter Estimation Pipeline” is data collection and parameter estimation, which has already been accomplished during the project. Consequently, the “ESP pipeline” is capable of functioning independently without relying on the “Parameter Estimation Pipeline” for its operations.

For data collection, the “Parameter Estimation Pipeline”, shown in figure 12 employs a straightforward Python script that interfaces with the handgrip trainer via serial communication. The script captures the measurement data and stores it in a CSV file for further processing. This measurement data comes from 2 sources:

- The *Pressure_sensor_test/src/main.cpp* script described in “Section 5.1”. This script generated the initial parameters by which the Kalman filter would initially be run. It implements a moving average filter in order to remove some of the noise present in the measurement circuit and writes the measurements to the PC as a CSV file.
- The *bachelor_esp_kalman/src/main.cpp* script described in “Section 5.1”. This script uses the Kalman filter implementation to reduce the noise from the measurements.

Subsequently, the data is imported into MATLAB, where a least squares estimator is employed to determine the appropriate ϕ parameters. This estimation process allows for the characterization of the system dynamics based on the collected data.

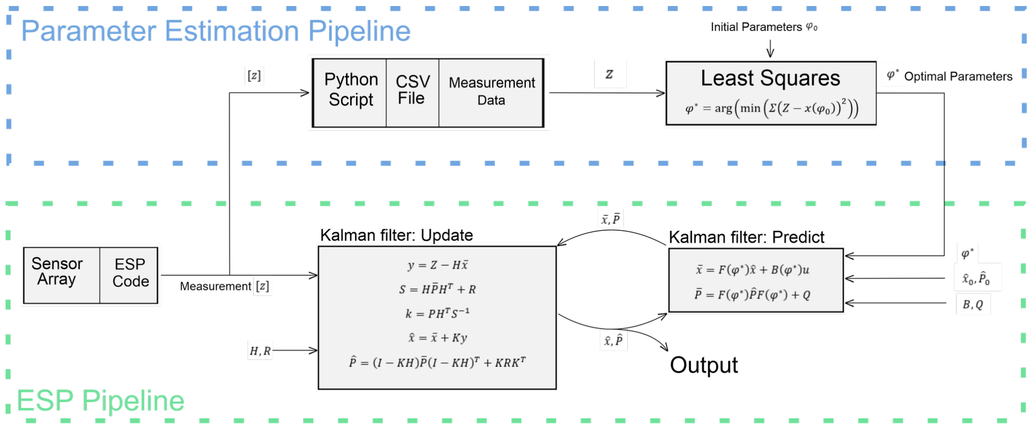


Figure 12: Pipeline from the sensor to output

As seen in figure 12, the Kalman filter is implemented in the "ESP Pipeline", which is a real-time embedded system. It is important to note that this script can be run after an initial set of ϕ-parameters has been estimated. This was detailed in "Section 5.1". As seen in figure 12, the estimates generated from the Kalman filter serve as the output we need. These estimates are the active muscle mass and the fatigued muscle mass, which fully describe the grip dynamics. Further in the project work, this output has been fed back to the estimation pipeline in an attempt to improve the estimates, as shown in figure 13.

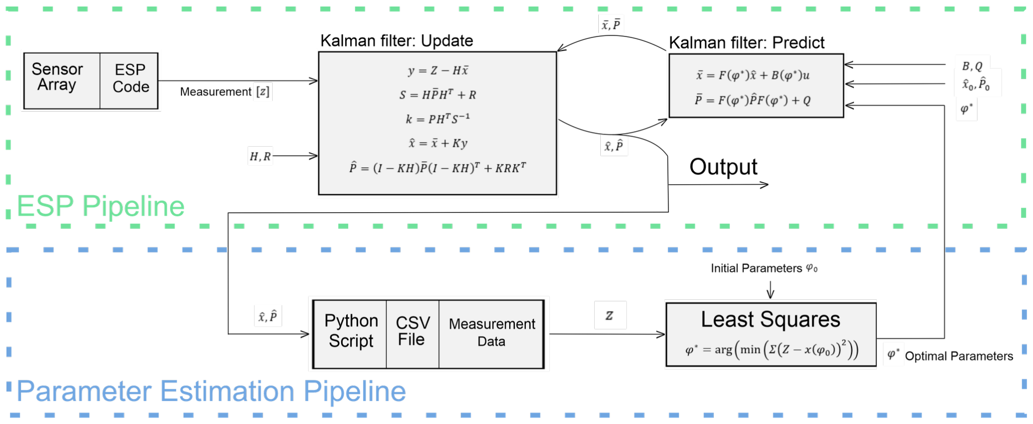


Figure 13: Real-time pipeline using the Kalman Filter for noise reduction

In this case, the Kalman filter has been used to reduce measurement noise, in an attempt to better fit the model to the data through the least squares estimator. Note that this pipeline requires a working ϕ-parameter set, otherwise, the Kalman filter cannot generate accurate predictions. This pipeline is the one currently implemented in the ESP32. The parameter set which best fits the model to the data set *test2.csv* was used, since the performance of the filter for this parameter set was deemed satisfactory.

Regarding the filter's performance, it is consistent with the expectations that the group had in the preliminary project. However, problems arise due to the slight non-linearity of the system, namely the switching between the active model and the resting model. Some possible solutions and suggestions for further work on this topic are described in "Section 7.5".

The Kalman filter output is shown in figure 14. As demonstrated by the serial plotter, the filter (gray) manages to remove a significant amount of noise from the measurement (orange). In figure

14 the measurement and Kalman filter estimate are practically overlapping. This is due to the Kalman filter being tuned in favor of the measurement. The fatigued muscle mass (red) shown in the figure however does not make physical sense. This result was consistent regardless of the ϕ-parameters that the prediction block uses. The reasons for this deviation have been explained extensively in “Section 7.5”.

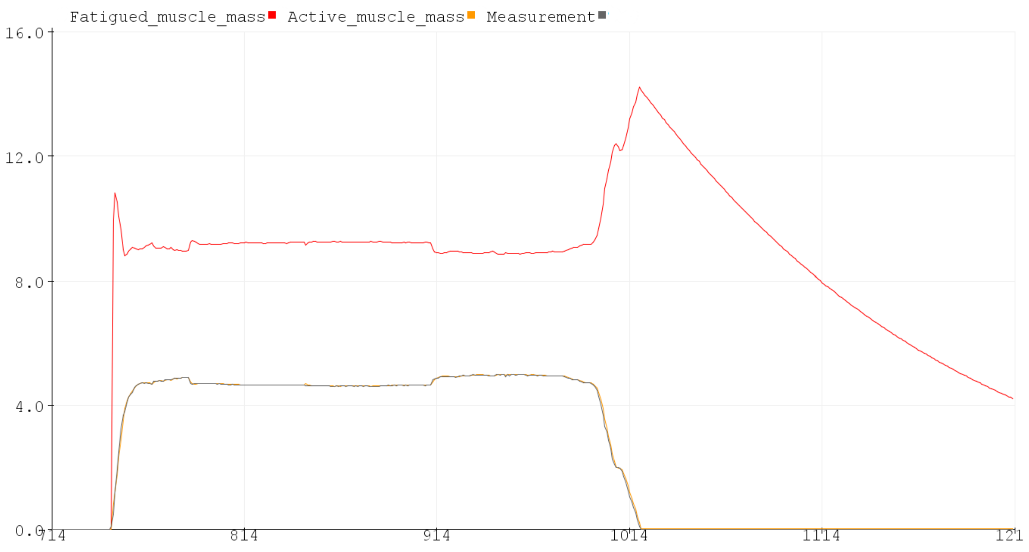


Figure 14: The output of the Kalman filter in the Serial Plotter

Additionally, a residual plot for the Kalman filter has been included in this section, in figure 15. The orange and gray lines represent the $[-3\sigma_{m_a}; 3\sigma_{m_a}]$ interval where the residuals must lie, given that they are normally distributed variables. It is important to note that this is an evaluation criterion based on the theory behind normally distributed variables. It is in no way descriptive of the total performance of the filter. This is further supported by the incorrect value of the fatigued muscle mass shown in figure 14.

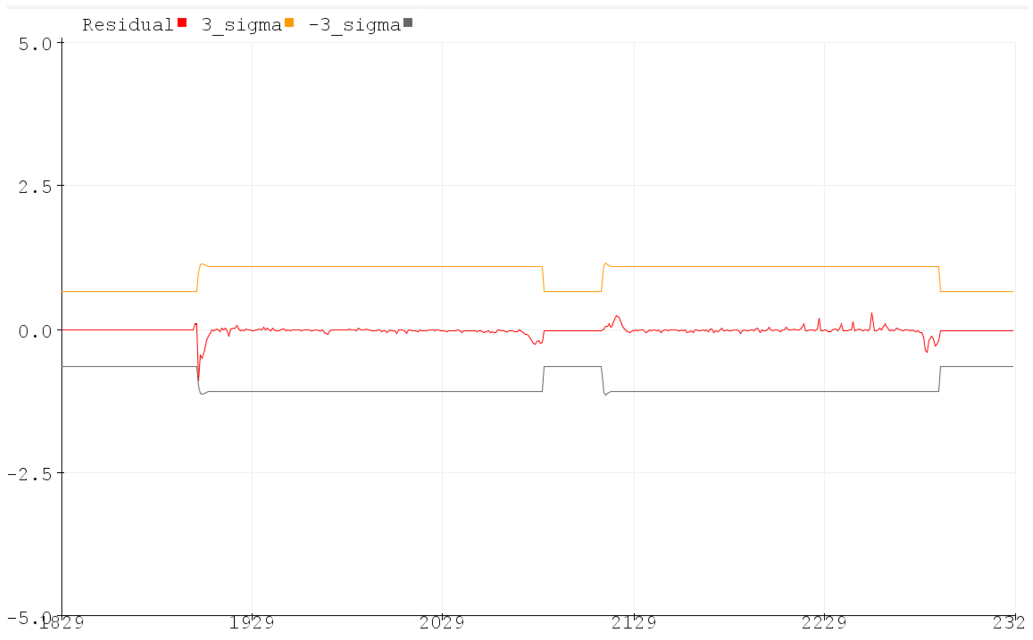


Figure 15: Kalman filter residuals in the Serial Plotter

7 Discussion

7.1 Mathematical model

The two-state mathematical model described in “Section 3.1.1” provides an accurate description of the muscle dynamics in a gripping motion. However, it does not account for the possibility of cramps in the muscle mass. In order to factor cramping into this model, the three-state mathematical model in “Section 3.1.3” must be used. It is important to note, however, that this model is not globally identifiable. This means that there are multiple solutions to the least squares problem defined in “Section 3.2” if one chooses to implement this model in the objective function. Additionally, the model must be discretized before it can be used. This is done in exactly the same way as the derivation in “Section 3.1.2”, where the forward Euler method was used.

Regarding the usage of the forward Euler method, there are certain disadvantages which limit the size of the sample time for the measurement circuit. Having a large sample time can lead to the algorithm becoming unstable. This can be avoided by setting a high sampling frequency in order to obtain more measurements, or by changing the discretization method. In this project, a high sampling frequency was set in order to fix this issue. The reasons for this choice are as follows:

- Setting a higher sampling frequency is a much easier solution than discretization via a more complicated method.
- The forward Euler method sets clear boundaries for the parameter estimation problem described in 3.2.
- Other numerically stable methods, such as the backward Euler method or the trapezoidal method, are implicit methods which require more computing power. They present an additional challenge of transforming the discrete system into a state-space form. Additionally, the constraints for the parameter estimation problem are not clearly defined through these methods.

7.2 Electronics

7.2.1 Pressure sensor

The availability of the sensor has varied throughout the duration of the project. Other similar piezo resistor sensors should work but might need some new tuning.

7.2.2 Operational amplifiers

During the initial phases of the project, individually packaged OP-amps were tested and demonstrated comparable performance to the four-in-one package counterparts. However, due to the availability of the already acquired four-in-one packages, their usage was maintained throughout the project. Nevertheless, it is worth noting that the circuit and PCB design can be modified to accommodate individually packaged OP-amps, presenting opportunities for a more space-efficient and cost-effective circuit configuration.

7.2.3 Resistors

The current resistors were found to give a desirable measurement range and low noise, but they have not been subjected to extensive comparative testing. It is therefore recommended to conduct a more in-depth analysis of the resistors to identify a more suitable combination that can deliver an optimal measurement range while maintaining low noise levels.

7.2.4 Batteries

The current battery arrangement for the trainer is suboptimal. To improve the trainer size, smaller batteries such as Li-Po or Li-Ion batteries could be utilized, and paired with a boost circuit to address voltage sag that commonly occurs during battery discharge.

7.2.5 ESP32 breakout boards

In addition to the TTGO breakout board, alternative ESP32 microcontrollers could be employed for the project. However, it is important to note that different pinouts on these alternative breakout boards may render them incompatible as direct replacements for the current PCB. Consequently, modifications to the PCB design would be necessary to ensure compatibility with different ESP32 microcontrollers.

7.2.6 Tested circuit and PCB

The circuit was tested on a breadboard, and not the PCB. The breadboard was connected exactly like the PCB, but the tested breadboard setup was found to be highly susceptible to noise. This was primarily due to variations in wire angles and tension, leading to inconsistent noise levels. moreover, the sensors were prone to disconnection due to poor connections on the breadboard and its wires. To address these issues, it is strongly recommended to utilize the PCB and establish hard-wired connections. This approach will provide more reliable and stable connections, minimizing noise interference and ensuring a more robust system overall.

While utilizing a PCB offers significant advantages, such as improved reliability and compactness, it also comes with certain drawbacks. One such limitation is the inability to modify the physical PCB once it has been manufactured. The PCB design is specifically tailored to accommodate the listed parts mentioned in “Section 4.1”, including their specific pin-outs. Therefore, if alternative parts need to be used or any modifications are required, a new PCB would need to be manufactured to accommodate these changes.

7.2.7 Low-pass filter

To mitigate the noise from the sensor and OP-amp, the implementation of a simple low-pass filter can be considered. This filter would be effective in filtering high-frequency noise, thereby reducing the impact of extreme noise on the circuit. However, it is crucial to emphasize that design, testing and tuning of the low-pass filter are vital to achieve optimal noise-filtering performance. Thorough testing would involve evaluating the filter’s ability to preserve the desired signal while attenuating unwanted noise components. Proper tuning of the filter components would need to strike a balance between noise reduction and preservation of the relevant signal.

7.2.8 EMG sensor

As mentioned in “Section 3.2.1”, the system is not globally identifiable if the total muscle mass M is unknown. This means that we cannot find a unique parameter combination which yields a minimal sum of squares. In “Section 5.1”, a grid search was used in order to overcome this limitation. However, measuring the total muscle mass being used through EMG sensors can be an alternative approach to solving this problem. By modifying the measurement circuit, five measurements can be logged in the data logging script: one measurement for each finger, as well as the total mass being used at each time step. This solution was not used in this project because of its cost and its disadvantages regarding ergonomic design. Additionally, this modification of the measurement circuit would require an overhaul of the mathematical model.

7.3 Physical design

7.3.1 Preliminary design

The preliminary design concept for a handgrip trainer involved the use of a balloon wrapped around a cylinder, which would be inflated and then squeezed to generate internal pressure proportional to the handgrip strength. However, this concept was deemed impractical due to several anticipated issues:

- To ensure consistent measurements, it is crucial to inflate the handgrip trainer with the same pressure each time. This necessitates that the trainer be left undisturbed during the inflation process.
- The balloon introduces a potential challenge regarding its movement during the inflation and deflation process. This movement has the potential to create inconsistencies in the measurements obtained from the trainer
- Incorporating alternative sensors in the handgrip trainer, aside from the balloon pressure sensor, poses difficulties due to constraints related to size and the cylindrical design of the device.

The challenges anticipated in the previous design prompted the development of a new design approach that prioritizes the utilization of more rigid components. This shift in design aims to address the limitations and drawbacks identified in this earlier concept, with the goal of enhancing the overall functionality and performance of the handgrip trainer.

7.3.2 Device iterations

(a) Normal

(b) Transparent

Figure 16: First iteration based on "Prototype 2" from "AME project"

This first iteration of the model is heavily based on the one explained in "Section 4.2". The result is in the model shown above. This model incorporates a balloon inside of a 3D-printed frame with 3 pressure sensors on the inside shown as orange buttons in figure 16b, unlike the original that had the inflatable element folded through the frame.

(a) Balloon behind pressure points (b) Balloon in direct contact with sensors

Figure 17: Second iteration concept ideas

The second iteration represented a complete redesign, with the exception of it still using the balloon. In this iteration, the sensors were arranged on a single side of the device, allowing for individual measurements of each finger's grip strength. it is important to note that this design iteration did not include functional components but serves as a conceptual model for future development. Notably, this redesigned model prioritized improved ergonomics compared to the previous iteration, aiming to enhance user comfort and usability.

(a) Without showing balloon (b) With balloon

Figure 18: Third iteration

The third iteration of the handgrip trainer marked a significant advancement as it transitioned from a conceptual model to a functional prototype. Unlike the previous iterations, this model incorporated elements of rigidity and robustness. During testing, it was observed that the device had a tendency to slip out of the hand due to its flat and smooth surfaces. This discovery highlighted the need for further improvements to enhance the grip and overall usability of the trainer.

(a) Without showing balloon (b) With balloon

Figure 19: Fourth iteration

In the fourth design iteration, a notable improvement was the addition of a rail specifically intended to stabilize the thumb and the entire hand during gripping exercises. While this rail addressed the issue of hand slippage, it was discovered that it introduced discomfort for the users. Furthermore, the sharp corner edges of the sensor side were also identified as a source of discomfort. As a result, the fourth design fell short of meeting the desired standards. These findings emphasized the importance of refining the design to ensure user comfort and satisfaction.

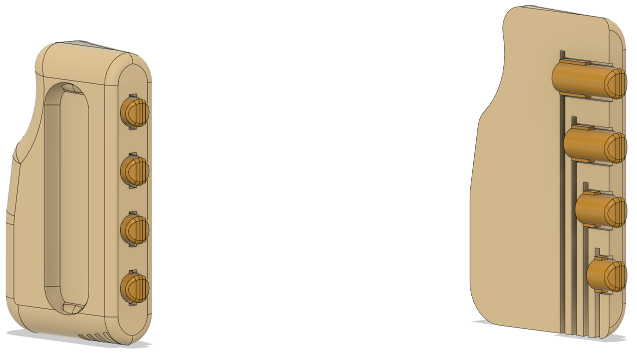

(a) Normal (b) Cross section

Figure 20: Fifth iteration

In the fifth design iteration, notable improvements were made in terms of ergonomics. However, it is important to acknowledge that the design still fell short of achieving optimal comfort. The new button design incorporated the sensors within the device, utilizing pogo-pins to apply pressure on them. Rubber foam was introduced to create a spring-like sensation and distribute pressure evenly across the sensors. This addition helped prevent the sensors from getting pressed on imperfections in the print and producing inaccurate readings.

One limitation of this design was the size of the pogo pins, which made the device less suitable for individuals with different hand sizes. The small size and surface area of the pogo pins also increases the likelihood of missing the intended squeeze area during gripping exercises.

(a) Normal (b) Cross section

Figure 21: Final design

The sixth and last iteration of the design saw big improvements on the pogo pins. The new pins have a larger surface area that makes the trainer easier to press, and more suitable for different-sized hands. The holes for the sensors are bigger, to make sure the wires fit

In the sixth and final iteration of the design, significant improvements were made to the pogo pins used in the device. The new pogo pins featured a larger surface area, which enhanced the overall usability and made the trainer easier to press. This modification addressed the previous limitation of the small pins, making the device more suitable for individuals with different hand sizes.

The increased surface area of the pogo pins provided a more comfortable and a more ergonomic gripping experience, allowing users to apply pressure more effectively and accurately. This enhancement contributed to the overall usability and functionality of the handgrip trainer, providing a better user experience and improving the reliability of the measurements.

Furthermore, the holes for the sensors were enlarged to ensure that the sensor wires could fit properly. This modification eliminated any constraints related to wire size and allowed for smoother installation and connection of the sensors.

Unfortunately, the current design of the trainer falls short of the intended ergonomic standards and significantly affects the user's ability to press the sensors. In particular, using the trainer in its intended orientation results in a weaker little finger output compared to when it is used upside down. This test alone reveals the suboptimal performance of the design, indicating the need for further development efforts to achieve a more optimal design that meets the intended ergonomic requirements. Therefore, continued development of the project should prioritize improving the design to ensure optimal user experience.

The rubber foam, utilized to provide the pins with a spring-like sensation and to distribute the pressure evenly across the sensors, had the drawback of tending to catch on the inside walls of the device, causing it to remain compressed. It is also worth noting that rubber foam tends to lose its ability to expand effectively with extended use. To overcome this challenge, a potential solution would be to create hollow rubber or silicone balls or cylinders that can be employed as springs instead of the current rubber foam solution. This approach may provide a more effective means of achieving the desired spring-like sensation while preventing the material from getting weaker over time and getting caught in the holder's interior walls.

In order to simplify the cleanup process during 3D printing, the current model has been designed with a "print-in-place" approach. However, this approach is not ideal when combined with the lack of mechanical expertise. The model is printed with the pogo pins already in place to minimize the need for additional supports that would otherwise be required to prevent the model from collapsing during printing. If the pogo pins were not pre-installed, the holes for either the sensors or the pogo pins would end up filled with supports. Removing these supports from the holes would be a tedious task. therefore, the model is printed with the sensor holes facing upwards and the pogo pins already incorporated into the design.

The process of removing the pogo pins can indeed be challenging, as it often requires significant force. To facilitate their removal, it is recommended to use a pair of pliers for added grip and leverage. However, it is important to note that there is a high risk of damaging or destroying the pogo pins during the removal process. It is therefore advisable to print a spare set of pogo pins beforehand. This ensures that replacements are readily available in case any of the original pogo pins become damaged during removal.

7.4 Parameter estimation

As seen in the results in "Section 5.1", the least squares estimator which was chosen performed poorly. One of the reasons for this is the hardware setup. As explained in "Section 7.2.6", the breadboard possessed a decreased signal-to-noise ratio. The decrease in this ratio is reflected in the parameters returned from the estimator. Some of these parameters are physically impossible, as noticed in the data set *test_bias4.csv*. A printed circuit board would eliminate much of the noise that is present in the data.

In some cases, the MATLAB solver would generate a solution that did not fit the data set. This occurred because of the estimator's structure and the nature of the optimization problem. The least squares method simply generates a set of parameters that have the least deviation from the data set. If the data set is noisy, then the least squares estimator might end up fitting the data poorly (as shown in figure 7). Some possible modifications that can be made in order to avoid this phenomenon are listed below:

- Tweaking the solver's options. More specifically, increasing the number of iterations and increasing the number of function evaluations leads to optimal solutions, regardless of the initial value of the ϕ-parameters. Increasing the tolerance for the first-order optimality measure might also be a solution, however, caution must be exercised, since the estimator can yield suboptimal parameters if the tolerance is increased.
- Changing the initial value of the ϕ-parameters. By setting the ϕ-parameters to reasonable values, an optimal solution can be reached without the need for tweaking the solver. However, this would require a certain degree of knowledge regarding the system. A solution used in this project was setting the initial value to 0.5 for all the scripts. This ensured that the solver

would find an optimal parameter combination.

- Using a better estimator. Given the noise in our data, a valid option could be the usage of a different estimator. The least squares estimator yields accurate results when the signal-to-noise ratio is high. However, this was not the case in this project, and the deviations in the estimator are quite clear. A better and more statistically rigorous estimation algorithm (such as the expectation-maximization algorithm) might be able to produce better results. However, this algorithm introduces concepts such as Kalman smoothers and a challenging mathematical framework. Given the short time frame of the project, as well as the complexity of the algorithm, the group did not implement this algorithm. However, it is recommended to analyze this algorithm further in order to improve the parameter estimation. In that regard, the work of Shumway and Ombao [3] is recommended as a starting point, with additional numerical optimization, as shown by Mader et al. [5].

In some cases, the filter manages to fit the data to the model accurately. This is especially true if only one of the sensors is connected to the model. The group was unable to test this phenomenon with the PCB and has therefore concluded that multiple tests need to be made before an accurate explanation can be given. At this point, the tests that were conducted have used the breadboard setup described in “Section 7.2.6”, which adds significant noise to the measurements. The group, therefore, recommends more tests of the parameter estimation pipeline with a PCB-based measurement circuit. This would in theory improve the estimates greatly.

Additionally, the estimation pipeline yields different sets of parameters for different fingers. In all test cases, the parameters have been more reliable for the index and middle finger, and more unpredictable for the ring finger and the little finger. This can be explained by the role of the different fingers in grip strength, and how the index and middle finger can output higher grip strength than the ring and little finger [2].

7.5 Kalman filter

The results from the implementation of the Kalman filter shown in “Section 4.4” show good noise reduction. Additionally, the residuals from the filter, as shown in “Section 6” are within the $[-3\sigma_{m_a}; 3\sigma_{m_a}]$ interval, showing good theoretical performance for the filter. However, the second state (the fatigued muscle mass) that is estimated by the Kalman filter deviates from the expected values. This is because the switching between the hand contraction ($u = 1$) and the hand relaxation ($u = 0$) is done through a measurement check. If the measurement is larger than a certain value, then we assume $u = 1$ and the trainer is getting gripped. Otherwise, the trainer is at rest. This form of switching introduces a delay which causes the fatigued muscle mass to increase drastically before switching the models. A solution that was tried during the project work was a derivative evaluation. If the difference between the two last measurements, divided by the sample time, is larger than a certain value, it is assumed that the models have switched. However, this solution is not recommended, due to its large inaccuracy caused by the noise in the measurement circuit. A possible solution could be embedded in hardware by adding a push button to the measurement circuit. This push button can send forth a boolean value which, when changed, switches the model from resting to active or vice versa. However, this would add a layer of complexity to the design. It would also require the users to activate the push button at the same time as they grip the trainer. If this sequence of operations is not executed properly, the filter will not perform well. It is important to note that this switching problem can only be seen in a real-time implementation. Simulating the Kalman filter in a dataset will yield different results without requiring any modifications to the trainer.

Additionally, it is important to mention that the ϕ-parameters are not necessarily constant. This is a simplification of the model which is done in order to express it as a linear time-invariant (LTI) model, which can then be inserted in the prediction block of the Kalman filter.

As mentioned in “Section 3.3.1”, the Joseph equation has been used for the computation of the state covariance matrix. This choice has contributed to the numerical stability of the filter by reducing the chance of obtaining an asymmetric state covariance matrix. If this asymmetry were to happen, the Kalman filter would quickly diverge. This becomes clear if we see the definition of the state covariance matrix for our system:

$$\hat{P} = \begin{bmatrix} \sigma_{m_a}^2 & Cov(m_a, m_f) \\ Cov(m_f, m_a) & \sigma_{m_f}^2 \end{bmatrix}$$

From an intuitive point of view, $Cov(m_f, m_a)$ must be equal to $Cov(m_a, m_f)$. If this is not the case, the state covariance matrix does not make physical sense because the covariance between m_a and m_f cannot have 2 different numerical values in the same time step. However, due to floating-point approximation in the embedded architecture used for this project, such divergences can occur. The Joseph equation simply corrects this approximation error.

The implementation of the Kalman filter on the ESP32 TTGO used a linear algebra library in order to overload the different operators. This library was wrapped in a structure in order to add a further layer of abstraction over the underlying math of the Kalman filter. This design choice was made in order to facilitate further work with the filter by "hiding" away the already established mathematical framework. Alternatively, an implementation described by Kettner and Paolone is also valid in this case[6]. This implementation, also called as a Sequential Discrete Kalman Filter, optimizes the load on the architecture as well as the time complexity of the algorithm. However, this algorithm is recommended for architectures which support parallel programming (such as FPGAs). Even though the design is possible for an inferior architecture such as the ESP32 TTGO, it was decided that this was not a key learning objective for this project.

8 Conclusion

As described in the results "Section 6", the thesis successfully demonstrates a proof-of-concept handgrip trainer with a measurement system. The device incorporates a Kalman filter to enhance the stability and accuracy of the measurement signal, making it suitable for various applications. However, it is important to note that despite the progress made through iterations and improvements, the device is still in its developmental yet ready for production.

The device's electronics are simple and functional, but not ideal. The OP-amps used are four-in-one packages, but only one OP-amp in the package can be used at a time. The 9V batteries are also physically quite big for the amount of power needed to run the sensors. These artefacts are the results of the proof-of-concept approach.

The electronics of the device, while simple and functional, are not considered ideal due to certain limitations. One such limitation is the four-in-one packages for the OP-amps, where only one OP-amp can be utilized at a time. This inefficiency is a consequence of the proof-of-concept approach taken during the development of the device.

Another aspect that can be improved is the choice of 9V batteries as the power source. These batteries, although effective in powering the sensors, may be physically larger than necessary for the power requirements of the device. This discrepancy between the battery size and power needs can be addressed in future iterations of the design.

As explained in "Section 1", the existing handgrip trainers are available in the current market, but they do not have a measurement system. The device has therefore undergone many iterations to incorporate the sensors inside. These iterations also placed a strong emphasis on achieving a good enough ergonomic design that ensures usability. The current design achieves this ergonomic usability point, but nothing more. Further development should emphasize this area.

To facilitate the measurement process, the device incorporates pogo pins in conjunction with rubber foam. This combination serves the purpose of exerting pressure on the sensors. Moreover, the utilization of rubber foam helps address the challenge of uneven surfaces inherent to 3D printing. By distributing the pressure evenly and minimizing interference caused by surface imperfections, the device's sensors are able to function effectively.

The parameter estimation process in the project has demonstrated suboptimal performance. While it is capable of producing working parameters, it heavily relies on high signal-to-noise ratio data. When accurate parameters are obtained, they can be utilized within the Kalman filter implemented in the ESP32 TTGO device. The Kalman filter effectively filters out a significant amount of noise; however, it encounters difficulties in estimating the hidden parameter known as the ”fatigued muscle mass.” These challenges primarily stem from the switching between the F-matrices and the estimation of the parameter u.

The estimation of u in the project has proven to be inadequate based on the employed methodology. Therefore, it is crucial to allocate dedicated attention and focus on addressing this issue in future iterations of the project. As discussed in “Section 7.5”, this problem primarily manifests in the real-time application of the Kalman filter, where accurate estimation of the fatigued muscle mass becomes challenging.

Appendix

A git repository was used for the code included in this project:

https://github.com/OriginalMetaTrademarkPending/IELET2920

The structure of the repository has been reworked multiple times, and given the visibility of the repository, the organization structure described below might not be consistent with the repository's content.

At this point in the project, there are 3 branches. Two of these branches have been used previously as development branches. The *parameter estimation* branch has been used to write and test the least squares estimator. The *master* branch was used to integrate the Kalman filter into the pipeline, making the branch complete. In the end, the master branch was synchronized with the *main* branch. The git log contains complete information on the project progression.

The group suggests using the master branch for further work.

Bibliography

[1]E. Innes, 'Handgrip strength testing: a review of the literature', Australian Occupational Therapy Journal **46**, 120–140 (2002).

[2]J. Macdermid, A. Lee, R. Richards and J. Roth, 'Individual finger strength: are the ulnar digits "powerful"?', Journal of hand therapy: official journal of the American Society of Hand Therapists **17**, 364–367 (2004).

[3]M. Ho, R. Shumway and H. Ombao, *The state space approach to modelling dynamic processes: applications in neuroscience and social sciences*, 2006.

[4]M. S. Grewal and A. P. Andrews, *Kalman filtering: theory and practice using matlab* (John Wiley & Sons, Inc., 2008).

[5]W. Mader, Y. Linke, M. Mader, L. Sommerlade, J. Timmer and B. Schelter, 'A numerically efficient implementation of the expectation maximization algorithm for state space models', Applied Mathematics and Computation **241**, 222–232 (2014).

[6]A. M. Kettner and M. Paolone, 'Sequential discrete kalman filter for real-time state estimation in power distribution systems: theory and implementation', IEEE (2017).

[7]S. A. Ross, N. Nigam and J. M. Wakeling, 'A modelling approach for exploring muscle dynamics during cyclic contractions', PLOS Computational Biology **14**, 1–18 (2018).

[8]P. Bobos, G. Nazari, Z. Lu and J. C. MacDermid, 'Measurement properties of the hand grip strength assessment: a systematic review with meta-analysis', Archives of Physical Medicine and Rehabilitation **101**, 553–565 (2020).

[9]H. Mielke and L. Wöhrle, 'Ame projekt', 6–16 (2021).

[10]J. Depp, *Why a strong grip is important, and how to strengthen those muscles*, (Apr. 2022) https://health.osu.edu/wellness/exercise-and-nutrition/why-a-strong-grip-is-important#:~:text=In%5C%20athletes%5C%2C%5C%20it's%5C%20important%5C%20to,improve%5C%20your%5C%20quality%5C%20of%5C%20life. (visited on 4th May 2023).

[11]R. R. Labbe Jr., *Kalman and bayesian filters in python*, https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python (visited on 10th May 2023).

[12]UNEO *Inc.*, *Ghf10*, https://www.uneotech.com/web/product/product_in.jsp?pdid=PD1607914673653&dmid=DM1598323840134 (visited on 28th Apr. 2023).