Advanced Control Systems for Exo-Skeletons

by

Alexander Boehler

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2021 by the
Graduate Supervisory Committee:

Thomas Sugar, Chair
Sangram Redkar
Kevin Hollander

ARIZONA STATE UNIVERSITY

December 2021

Copyright © 2021 Alexander Boehler All Rights Reserved

ABSTRACT

With the extensive technological progress made in the areas of drives, sensors and processing, exoskeletons and other wearable devices have become more feasible. However, the stringent requirements in regards to size and weight continue to exert a strong influence on the system-wide design of these devices and present many obstacles to a successful solution. On the other hand, while the area of controls has seen a significant amount of progress, there also remains a large potential for improvements.

This dissertation approaches the design and control of wearable devices from a systems perspective and provides a framework to successfully overcome the often-encountered obstacles with optimal solutions. The electronics, drive and control system design for the HeSA hip exoskeleton project and APEx hip exoskeleton project are presented as examples of how this framework is used to design wearable devices.

In the area of control algorithms, a real-time implementation of the Fast Fourier Transform (FFT) is presented as an alternative approach to extracting amplitude and frequency information of a time varying signal. In comparison to the peak search method (PSM), the FFT allows extracting basic gait signal information at a faster rate because time windows can be chosen to be less than the fundamental gait frequency. The FFT is implemented on a 16-bit processor and the results show the real-time detection of amplitude and frequency coefficients at an update rate of 50Hz.

Finally, a novel neural networks based approach to detecting human gait activities is presented. Existing neural networks often require vast amounts of data along with significant computer resources. Using Neural Ordinary Differential Equations (Neural ODEs) it is possible to distinguish between seven different daily activities using a significantly smaller data set, lower system resources and a time window of only 0.1 seconds.

DEDICATION

To my wife Helen and my son Christopher.

They have supported and inspired me during the entirety of this journey.

I love them with all my heart and I could not have done this without them.

ACKNOWLEDGMENTS

First and foremost I would like to express my sincerest gratitude and appreciation to Dr. Thomas Sugar. His support, insight and guidance have been invaluable and without them this work would not have been possible. I would also like to thank my committee, Dr. Sangram Redkar and Dr. Kevin Hollander for their support and feedback throughout this work. Dr. Redkar has given me valuable ideas and direction especially in regards to the Neural Network based controls approach. Further, Dr. Hollander's efforts to organize and collect additional data for the development of our algorithm will not be forgotten. Finally, without Dr. Redkar I may never have decided to start this big undertaking.

Finally, I would also like to acknowledge the U.S. Air Force for providing the funding to support the APEX exo-skeleton project.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

With the advancement of the underlying technology, for instance microprocessors, motors and motor control, robotic assistive devices have become more and more common. The increased interest in these devices is due to mainly three reasons, an aging population, people with lifelong disabilities and the use for performance augmentation in industrial and military applications. As people age they walk less. Further, the elderly are more likely to develop gait disorders, lower limb impairment and medical conditions such as cardiovascular disease, which lead to stroke and lower limb amputation. According to a survey from United Nations currently about 11.5% of people are aged 60 and older and by 2050 this number is estimated to increase to nearly 22%. [66] For people with lifelong disabilities such as spinal cord injuries (SCI) robotic assistive devices can greatly improve quality of life. According to the Christopher and Dana Reeve foundation, in 2008 about 1.9% of the US population lived with some form of paralysis making it difficult or impossible to move their arms or legs [13]. 23% of those people are paralyzed due to SCI and approximately 12,000 new SCI cases occur every year. [45] Finally, for military and industrial applications these devices can help soldiers walk longer, carry more load or help warehouse workers with posture, joint and back-pain. [12]

Over the past ten years or so, big steps in technological advancement have been made in the areas of computer processing, sensing as well as motors and motor control. These advancements in turn have made wearable robotic devices much more feasible. The most important hardware tools are motors, motor drives (controllers), processing and sensing. Small, lightweight and powerful motors to drive these robotic devices have been around for quite some time, however, the drives to control them were lagging behind. 10 years ago motor controllers that were equally small, powerful and had good dynamic performance were impossible to find. Motion sensors, which are used abundantly today not only in many robotics applications but also almost every consumer electronic device such as cell phones and laptops, were in a state of infancy compared to today's systems. For instance, a one-axis gyroscope was about the size of an average person's thumb and costed several hundred dollars. Today, a full 9-axis inertial measurement unit (IMU) with 3 gyro, 3 accelerometer and 3 magnetometer axes with larger measurement range and better sensitivity is barely bigger than a rice corn and costs a few dollars. There are also powerful motor controllers available that are sufficiently small to fit onto these devices. In regards to processing there exist of course the Raspberry Pi, Beagleboard and Arduino off-the-shelf micro-controller PCBs [23; 22; 70]; however, the combination of hardware architecture, processing power and peripheral support makes these more targeted towards the hobby and maker community rather than for research or commercial use. Due to the unique requirements of wearable robotics, a project specific PCB is often still necessary. Because there are a multitude of solutions available today, the difficulty lies in the overarching design, which solves a particular problem in an optimal way.

Therefore, with the existing and future demand for wearable devices and technology readiness it is expected that these devices will become more and more common in our daily lives. Their design, however, is far from trivial because they encompass the combined disciplines of Mechanics, Electronics, Controls and Computer Programming. Furthermore, a key gap still exists: reliable, real-time capable control algorithms, which can successfully detect activities. In this dissertation, I present a systems approach to the problems encountered as well as a novel control algorithm for hip exoskeletons.

1.1 Organization of the Dissertation

Chapter 2 describes a literature review of the current state of the art of controls for exoskeletons.

In Chapter 3, I describe electronic systems that are used to control wearable devices and examine them from a systems perspective. Challenges faced when designing or selecting electronics and hardware for these devices are presented as well as possible approaches and solutions.

Chapter 4 presents basic structures for the control of wearable devices. In contrast to Chapter 5 and Chapter 6, the work described here evolves around higher level control structures rather than a specific algorithm. Again, the goal is to present requirements and approaches from a systems perspective to ensure a high chance of success for the correct operation of any wearable device.

In Chapter 5, I present the real-time implementation of the Fast Fourier Transform (FFT) as a method of extracting basic information like frequencies and amplitude from gait signals.

In Chapter 6, I present a novel activity recognition algorithm for a hip-exoskeleton using Neural Ordinary Differential Equations (Neural ODEs), a fusion between Neural Networks and ODEs.

Chapter 7 summarizes work completed and contributions to the field of robotics as well as a list of completed and expected publications in conferences and journals. Possibilities for future work are listed and briefly discussed.

Chapter 2

LITERATURE REVIEW

Many researchers have worked on controls for lower limb exoskeletons as well as human activity recognition. In addition, the state of the art of existing devices and control strategies has been reviewed extensively [30; 76]. Huo et. al. [30], for instance, conducted an extensive review of control strategies while categorizing them into one of three groups: (1) control strategies that are based on signals measured from the human body, (2) control strategies based on interaction force measurement between the robot and the human body and (3) control strategies based on signals measured from the robot. In this review, however, we categorize control strategies by the type of control signal used.

2.1 Control Using Electrical Signals From the Human Body

Two areas, which in the grand scheme of things are still in their infancy but particularly interesting because they would enable true user intent control, are Brain Computer Interfaces (BCI) and control using the electrical signals of skeletal muscles (EMG). Brain Computer Interfaces (BCI) can be split roughly into two classes, invasive and non-invasive, whereas the former can be broken down further into subclasses. In their analysis of the state-of-the-art of brain computer interfaces, Lebedev et. al. [37] provide a helpful graphic showing the classifications, see Figure 2.1. The electroencephalogram (EEG) is the most widely used interface of the non-invasive type [30] and records electrical signals on the scalp. Due to its limited bandwidth, typically around 5-25bits/s, this interface offers simple communication with the external world. [37; 74; 4] However, it also spares the patient the risks that come with

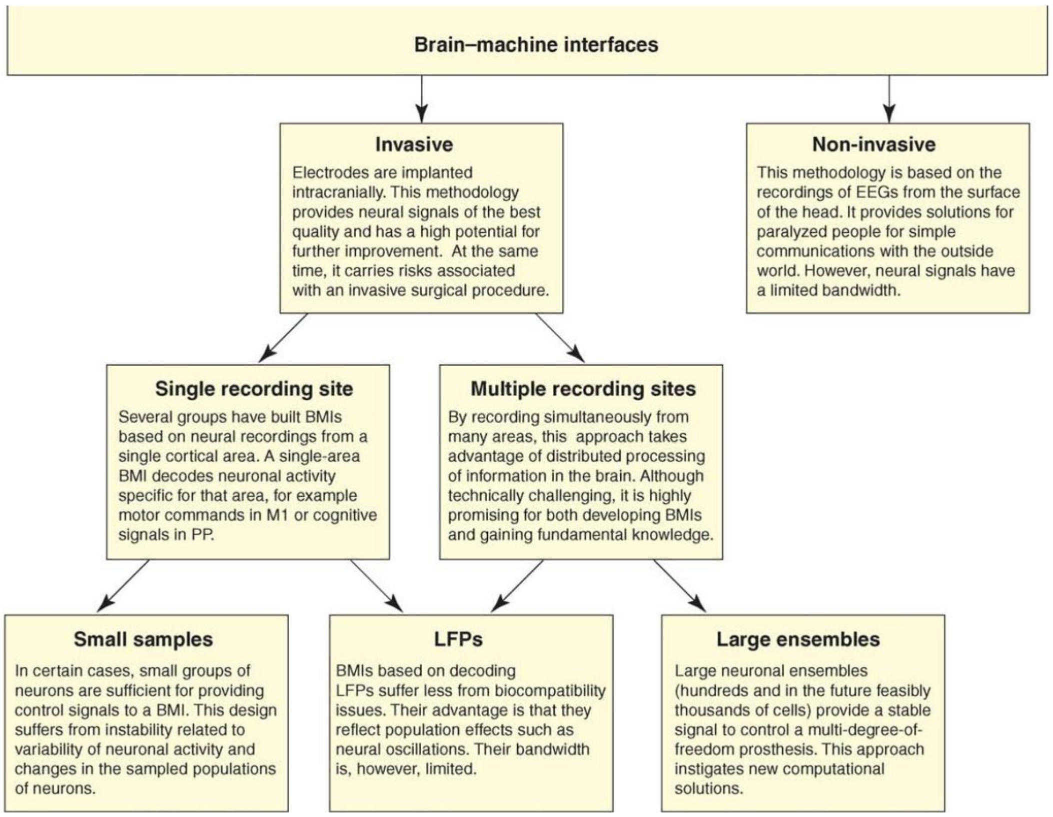

Brain–machine interfaces
Invasive
Electrodes are implanted intracranially. This methodology provides neural signals of the best quality and has a high potential for further improvement. At the same time, it carries risks associated with an invasive surgical procedure.
Non-invasive
This methodology is based on the recordings of EEGs from the surface of the head. It provides solutions for paralyzed people for simple communications with the outside world. However, neural signals have a limited bandwidth.
Single recording site
Several groups have built BMIs based on neural recordings from a single cortical area. A single-area BMI decodes neuronal activity specific for that area, for example motor commands in M1 or cognitive signals in PP.
Multiple recording sites
By recording simultaneously from many areas, this approach takes advantage of distributed processing of information in the brain. Although technically challenging, it is highly promising for both developing BMIs and gaining fundamental knowledge.
Small samples
In certain cases, small groups of neurons are sufficient for providing control signals to a BMI. This design suffers from instability related to variability of neuronal activity and changes in the sampled populations of neurons.
LFPs
BMIs based on decoding LFPs suffer less from biocompatibility issues. Their advantage is that they reflect population effects such as neural oscillations. Their bandwidth is, however, limited.
Large ensembles
Large neuronal ensembles (hundreds and in the future feasibly thousands of cells) provide a stable signal to control a multi-degree-of-freedom prosthesis. This approach instigates new computational solutions.

Figure 2.1: Classification of Brain-Computer Interfaces [37]

brain surgery. Existing research shows that despite its transfer speed limitations it has proved useful for paralyzed patients to regain some communication with outside world. [27; 35; 36; 46; 47] Simply put, EEG based BCIs try to decode a person's intentions and decisions by measuring the combined electrical signals of a huge amount of neurons. As result, resolution becomes limited because overlapping of different cortical areas occurs. Nevertheless, this technology can detect signals in the brain that are associated with visual stimuli, gaze angles, voluntary intentions as well as cognitive states. For example, researchers at the University of Aukland [43] have developed a robotic neurorehabilitation system, which uses visual stimuli. The interface helps the subject to control the speed of a lower limb orthosis. The researchers implemented Steady State Visual Evoked Potential (SSVEP) in order to control the robotic device.

SSVEP relies on detecting the stable oscillations generated by the brain when a visual stimulus is applied rapidly and repetitively. Such a visual stimulus can be generated by flashing LEDs, a strobe light or reversing patterns. In this case, researchers used a panel of twelve LEDs flashing at different frequencies whereas only four LEDs had a command associated with them. The commands the researchers used are “raise”, “lower”, “cycle” and “stop”. By concentrating on one of the four LEDs (commands) the subjects were able to move the lower limb orthosis. [43]

Researchers at the University of California have used an EEG based BCI to control a commercial robotic gait orthosis. Both, able bodied and SCI subjects were able to successfully operate the orthosis. To achieve this the researchers first recorded EEG while the subjects were engaged in alternating epochs of idling and walking kinesthetic motor imagery (KMI). Then, the data was analyzed offline to generate an EEG prediction model for real-time operation of the BCI. The prediction model consisted of a combination of classwise principal component analysis (CPCA) and approximate information discriminant analysis (AIDA). Testing was done in five 5-minute long sessions per subject. The subjects were asked to ambulate with the system while following computerized cues. For the offline analysis, researchers showed 86.3% prediction model accuracy. For the real-time test, the results showed cross-correlation of 0.812 ± 0.048 with a $p - value < 10^{-4}$ between instructional cues and walking epochs. [18]

Additional research in this area exists. The BETTER project (Brain-Neural Computer Interaction for Evaluation and Testing of Physical Therapies in Stroke Rehabilitation of Gait Disorders) aims to use BMI based assistive exo skeletons to improve physical rehabilitation of gait disorders in patients suffering from stroke. A particularly innovative aspect of BETTER is the approach of combining EEG with EMG in order to control the lower limb exoskeleton [53].

Researchers at the University of Houston in collaboration with The Neurological Institute at Methodist Hospital are working on augmenting a lower body and trunk exoskeleton (NeuroRex) with a BMI. They propose to use a kind of shared control where both neural signals collected through scalp EEG and manual control (via a handheld controller) is used to control their lower limb exoskeleton [14].

A second major area of research is the use of EMG (electromyography) signals, i.e. electrical activity produced by the skeletal muscles. The HAL-3 and HAL-5 systems both use EMG signals to measure the level of human-robot interaction. First, electrodes are placed on antagonistic muscle pairs, the biceps femoris and medial vastus as well as the gluteus maximus and rectus femoris. Second, the antagonistic EMG signals are used to estimate joint torque, which is then used to determine the assistive torque required by HAL-3. [32] The HAL-5 system added an new control component. When a user wears HAL-5 for the first time, the device records the user's walking profiles. The use of EMG signals is then simplified to determine when the user wants to walk, which in turn triggers the actuation of the previously recorded gait profiles. [25]

Fleischer et. al. from the Berlin University of Technology have researched a method to calculate the intended motion of joints by combining EMG signals with a biomechanical human body model. First, the current pose of the subject is determined using joint angles and floor contact information, which is fed into the biomechanical model. EMG signals are acquired and converted to muscle forces, which are also supplied to the biomechanical model. However, the EMG-to-force functions contain parameters that require calibration. Second, the model now calculates knee torque and acceleration of the knee joint, considering all joint torques and ground reaction forces. Finally, acceleration, velocity or position (through integration of acceleration) is used as desired motion and commanded to a motion controller. The researchers model the biomechanics using two legs with feet, shank, thigh and torso. Inverse dynamics is used to calculate, among others, joint torques. Forward dynamics use both EMG data and inverse dynamics outputs to solve for acceleration. Experiments have been performed with healthy subjects only and with the actuator detached. However, results show that the torques calculated using EMG data and torques calculated using the inverse dynamics during common daily activities, e.g. walking, climbing stairs, etc., match very well. [20]

One novel method of control worth mentioning in addition to EEG and EMG is the use of an FSR sensor to measure muscle hardness, which in turn is used to infer motion intention. The nurse assisting exoskeleton designed by researchers at Kanagawa Institute of Technology uses this approach. In principle, a contact projection transfers muscle hardness to pressure on the FSR, then pressure is used to estimate motion intention. [75]

In our lab at Arizona State University, we used FSR sensors to measure user intention based on FSR sensors mounted in a Wheatstone Bridge Configuration. [59]

Lastly, there is new research starting in detecting signals from peripheral nerve interfaces. This area shows promise because the signals are measured from residual nerves in the arm or leg. [62]

2.2 Control Strategies Using Kinematic and Kinetic Data

Some control strategies do not require measuring direct signals from the human body or interaction forces between the human and the robot. One such example is the BLEEX (Berkeley Lower Extremity Exoskeleton). It only uses information from the robotic device in order to estimate the human intention. The key to this is to increase the sensitivity of the device to the wearer's forces and torques. Sensitivity determines how much the user's forces and torques influence the exoskeleton velocity.

Therefore, if sensitivity is low, for instance if a strong velocity controller is applied, the user's forces and torques are attenuated, the controller compensates for those external inputs. However, if sensitivity is high the user's forces and torques do contribute to the exoskeleton velocity and allow the wearer to move the device easily. Hence, the goal is to increase exoskeleton sensitivity to the wearer's forces and torques and without actually measuring them. The researchers installed force and torque sensors at the hip, knee and ankle joints to measure the forces and torques introduced by both, wearer and actuators. Further, they developed two closed loops, one that represents how the wearer affects the exoskeleton and one that represents how the actuators affect the exoskeleton. Particularly interesting is that classical control theory usually tries to minimize sensitivity, thus minimizing disturbances, while their control algorithm tries to maximize the sensitivity. In principle, this simple solution can be achieved by positive feedback but at the cost of requiring precise knowledge of the system model. If the model is not accurate performance varies greatly from predicted performance or results in instability. Therefore, this control strategy has little robustness to parameter variations. [33]

Commercially available systems, like The ReWAlk or Ekso NR, often measure mechanical features of the human exoskeleton to control the device. For instance, the ReWalk, an exoskeleton with motors at the hips and knees intended for use at home and in the community, measures changes in the user's center of gravity and uses this information to control the device. If the system detects that the upper body is tilted forward, it initiates a walking step. Through repeated shifting in this way the device creates steps in sequence. According to the company this kind of control results in a functional natural walking gait. ReWalk [55]; Bionics [3]

Several research groups have worked on control strategies based on Adaptive Oscillators (AOs). AOs are mathematical tools that can adapt their parameters in order to extract the basic features of a periodic signal, i.e. frequency, amplitude and offset. These methods are particularly interesting and promising because they offer three important benefits: They do not require to specify a reference trajectory ahead of time, they do not require sensing complex signals (e.g. EMG signals) and they do not require user specific calibration [57].

Chinimilli et. al. [12; 11] from Arizona State University have developed an $A\omega$ adaptive oscillator ($A\omega AO$) coupled with a real-time human locomotion algorithm. The locomotion algorithm contains both low level and high level classifiers to detect activities and activity transitions respectively. A single IMU on each thigh is used to measure thigh angle, which then undergoes peak detection in order to compute amplitudes and frequencies (A, ω). These two features are passed to a support vector machine (SVM) to determine low level classifiers. A Discrete Hidden Markov model (DHMM) is used to obtain high level classifiers, i.e. it predicts the next probable activity based on a past sequence of activities detected by the SVM. A pool of three AOs together with a non-linear filter is used to compute future thigh angle, which then allows to generate the torque to apply at the hip. SVM output (activity information) is used to scale the torque signal while DHMM results are used to reset the AO when a transition is detected. Testing consisted of two sessions with three healthy subjects. In the first session two trials were performed to train the algorithm and one trial was performed to test the algorithm. In the second session the subjects wore a hip exoskeleton (HESA, Hip Exoskeleton for Superior Assistance) [61] and performed four activities, walking, going upstairs, going downstairs and jogging. Results show that the different activities clearly separate in the $A\omega$ - space. Further, the outputs of the non-linear filter and torque estimator produced a viable estimate of future hip angle and torque reference trajectory respectively. Overall, the tests showed high classification and prediction accuracy. [12; 11]

A research group from Switzerland and Italy have had similar success with controls using AOs for an upper body exoskeleton. They used adaptive oscillators as state estimators to smooth and anticipate the motion of the corresponding joint *in real-time*, in this case an elbow joint. First, an adaptive oscillator is used to measure amplitude and frequency of the elbow motion. The result is then fed into a state estimator to obtain elbow kinematics (joint angle, velocity and acceleration). Finally, a torque estimator uses the kinematic information to estimate torque, which is then scaled with a gain and applied at the elbow joint, adding it to the torque applied by the human. During the experiments each subject took part in three types of conditions, a “no-exo” condition, “constant frequency” condition and “variable frequency condition”. In the no-exo condition, subjects were wearing a simple 1-DOF goniometer to record movement kinematics while performing a repetitive movement task at a frequency of 1Hz. During the constant frequency condition participants performed several trials of a repetitive task at 1Hz with 0%, 33% and 50% assistance from the exoskeleton. During the variable frequency condition participants again performed a repetitive movement task but this time the frequencies were varied between 0.6Hz and 1.4Hz throughout the trials. During the trials, EMG activity of the biceps and triceps were measured. Results show that at the 50% assistance level EMG activity was decreased by 26% for the biceps and 59% for the triceps as compared to the no-exo condition. This indicates that the subjects had to produce less effort when the exoskeleton assisted them. [57]

Finally, researchers at Arizona State University worked on a state-based control for a powered ankle foot orthosis (AFO). The stance phase of the gait cycle is split into five zones and a different control method, velocity control, stiffness control or position control (hold position) is used during each zone that fits the system of human and robot. Transitions between states are detected using specific gait events. The advantage of this approach is that it removes the need to scale the amplitude of the reference signal as well as the need to adjust the timing of the signal. The different zones scale themselves based on where the user is in the gait cycle. The disadvantage of this method is that state based systems in general are prone to lock-up if one of the transitions is missing for some reason. However, in their testing the authors report that the initial tests look encouraging and that the controller did not get locked into one state during the tests. The resulting control trajectory looks similar in shape to the trajectory of the pre-programmed pattern and achieved similar amounts of output power. [7; 28]

2.3 Conclusion

A significant amount of research has been done in the area of controls for exoskeletons. The research spans many different methods ranging from the use of electrical signals measured directly from the human, e.g. EEG/EMG, to methods that measure human kinematics/kinetics, e.g. adaptive oscillators, or methods that do not require any direct measurements of human signals, e.g. increasing sensitivity to human torques and forces. Some of the methods allow the use of classical control theory while others require more recent developments in artificial intelligence and deep learning algorithms.

2.4 Critique

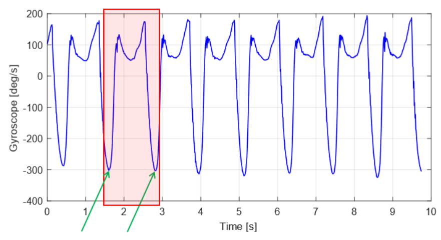
This critique focuses on control algorithms using adaptive oscillators and feature-space approaches. These approaches currently provide some significant advantages over other approaches discussed in the literature review, such as brain and nervous interface based methods. Two important advantages of AOs and feature space approaches are that they are non-invasive as well as less complex and more reliable on the hardware side than the various brain and nervous system interfaces, which require an actual connection to the human body. In the first section, we will closely investigate the previously presented papers from Chinimilli et. al. [12; 11] and Ronsse et. al. [57] about these topics. In the second section, we will examine possible extensions and future improvements of their work.

2.5 Amplitude-Frequency Feature Space for Activity Recognition

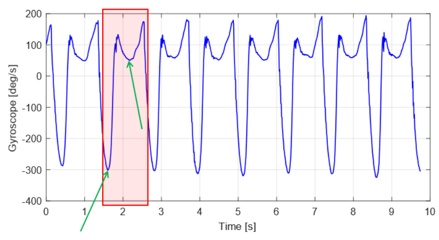
Because of the improvements in wearable motion sensing researchers have been developing activity recognition algorithms for some time. This usually involves several steps such as time window selection, feature extraction, dimension reduction and applying a classification algorithm [11]. Chinimilli et. al. present a two-dimensional feature space based approach for human activity recognition [12; 11]. The two features chosen by the authors are amplitude (A) and frequency (ω). This is sensible given that gait signals in general are periodic in nature. In [11] the authors present their main algorithm, which uses peak detection in order to find A and ω. Because the features are limited by design, feature reduction is not necessary. The machine learning algorithms Support Vector Machine (SVM) and k-Nearest Neighbors (k-NN) are employed for activity recognition. In [12] the authors expand their approach to account for transitions between the different locomotion activities using the Discrete Hidden Markov Model (DHMM). They also add a pool of AOs to increase robustness of the algorithm to encountered activities.

One additional critical characteristic of any *useful* activity recognition algorithm is that it must be *real-time* compatible. Running very complex and powerful algorithms on huge data sets during post-processing is acceptable in theory or for the evaluation of different approaches, however, if the algorithm is to be used to control a real-world device, the algorithm simply must be able to run in real-time.

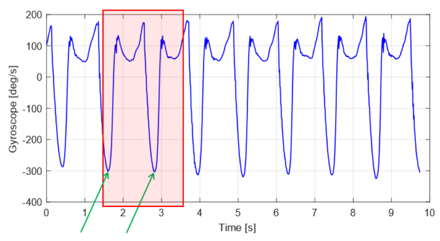
There are several ways to accomplish the first step of obtaining amplitude and frequency information about the motion signals. All of them require the use of a time window to look back at the data. To obtain frequency the authors chose to use the peaks search method (PSM), which relies on computing the distance between two adjacent peaks in the signal. Note that this search is done in the time domain. A time window is defined and the peak search is performed within that time window. In this method, the selection of the time window along with the peak search method are key parameters. The authors use a pre-written Python function to obtain the peaks and do not give much detail about the time window selection. Figure 2.2 shows the three different possibilities for selecting a time window. The blue signal is the angular velocity of the tibia in world coordinates of a person walking with their self selected pace (SSP).



(a) Time Window Selection: Most Efficient Window Size



(b) Time Window Selection: Window Too Short



(c) Time Window Selection: Window Too Long.

Figure 2.2: Influence of Time Window Sizing

Figure 2.2a shows an almost perfectly chosen time window. The window is just slightly larger than the distance between the peaks. It is large enough to detect the two peaks that determine the main frequency in the signal but not too large such that the peak search algorithm has to comb through more data than necessary adding to the computational load. Figure 2.2b illustrates what happens when the time window was chosen too narrow. Because the next consecutive peak determining the main frequency is outside of the window a peak detection algorithm will either detect a second false peak (the much smaller peak shown by the second green arrow) or only one peak which is insufficient for calculating ω. In either case, the result for ω, if there is one, is useless. Finally, Figure 2.2c shows a time window that was chosen too wide. This is preferable to the previous case for obvious reasons as the only down side is additional computing time, peaks are still detected correctly. The peaks search method is easily susceptible to noise in the signal. The gyro signal presented in the paper appears mostly clean and the Python peak search algorithm performs well. Once the frequency is known, the signal for a given activity can be approximated using a known number of terms determined using offline spectrum analysis. For example, an offline FFT of the gyro walking signal showed that three terms are sufficient to approximate the signal and hence the gyro signal can be approximated as follows

$$g(t) = A_1 e^{j\omega t} + A_2 e^{j2\omega t} + A_3 e^{j3\omega t} \tag{2.1}$$

Calculating gyro velocity and acceleration allows us to obtain the following system of equations in matrix form, which can now be used to calculate the amplitudes for the three terms.

$$\begin{bmatrix} g(t) \\ \dot{g}(t) \\ \ddot{g}(t) \end{bmatrix} = \begin{bmatrix} e^{j\omega t} & e^{j2\omega t} & e^{j3\omega t} \\ j\omega e^{j\omega t} & j2\omega e^{j2\omega t} & j3\omega e^{j3\omega t} \\ -\omega^2 e^{j\omega t} & -4\omega e^{j2\omega t} & -9\omega e^{j3\omega t} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} \tag{2.2}$$

Note, the above system of equations must be solved for each sample in the time window to compute a mean of each of the three coefficients. Finally, a resultant amplitude is calculated by taking the magnitude of the three means.

$$A = \sqrt{|A_{1_m}|^2 + |A_{2_m}|^2 + |A_{3_m}|^2} \tag{2.3}$$

There are other methods to determine the frequency, which are briefly mentioned. The Fast Fourier Transform (FFT) that has been developed for discrete signal processing, allows obtaining the frequency and associated amplitudes from a given sequence of time values of the signal. Applying the FFT also requires the use of a sliding time window though it would be interesting to research how the length of the required time window compares to other approaches. According to the authors, the FFT is the computationally most expensive of the methods. However, the computational load of solving a 3x3 matrix (by computing the inverse or some other algorithm) *for each data point in a time window* in order to obtain the amplitude A is also not to be underestimated. The FFT on the other hand provides both A and ω in the same calculation. The FFT is also real-time compatible, see [16]. Another method used in signal processing to obtain ω information about a signal is the auto-correlation function (ACF). ACF correlates a signal $f(t)$ with a delayed version of itself $f(t-\tau)$ as shown below in the discrete form.

$$\varphi_{xx}[\kappa] = \frac{1}{N} \sum_{k=0}^{N-1} x[k] \cdot x[k-\kappa] \tag{2.4}$$

The ACF works particularly well for noisy signals but requires the use of a lagged version of itself. This in turn results in additional time lag and does not appear to be a good way of computing frequency for this application in real-time. Any lag in detecting a changed ω and A will result in detecting changes in a person's gait later and therefore increases the chance that the device interferes with the person's gait. Finally, even though not discussed by the authors, wavelet transforms may also be used to find frequency and amplitude of a signal. This transform, shown in Equation 2.5 in its continuous form for simplicity, is obtained by multiplying a function $x(t)$ with a "mother wavelet" $\psi(t)$ and integrating it, similarly to the Fourier transform. There are several commonly used mother wavelets such as the "Haar" wavelet. The resulting function $W(a, b)$ is now a function of scale and time. However, the wavelet transform, is a fairly recent development in signal theory and its real-time application is not well understood yet. There are, however, several attempts, particularly in the neurology area for implementing wavelet transforms on digital signal processors in real-time. [50; 54; 51]

$$W(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \psi \left(\frac{t - b}{a} \right) x(t) \, dt \qquad (2.5)$$

After obtaining A and ω the next step now is to use the extracted features to train the machine learning algorithms offline in order to build the classification models. The authors use two well known machine learning algorithms, the Support Vector Machine (SVM) and k-Nearest Neighbors k-NN. The SVM starts with the assumption that the data is linearly separable, shown in Figure 2.3.

It then searches for a plane (a hyperplane for n-features) that maximizes the separation between the different classes. The SVM classifier is kernel based. This means it uses a mapping procedure (i.e. the kernel function) in order to make the dataset look more like a linearly separable dataset. Some common kernels are polynomial

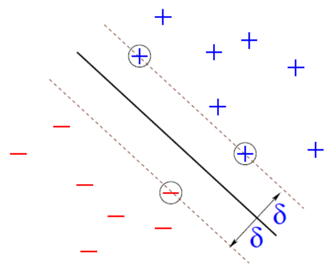


Figure 2.3: Idea of Linearly Separable Dataset

or Gaussian. Due to real-time requirements the authors use a linear kernel, which works well with linearly separable data. A feature space plot of A versus ω, shown in Figure 2.4 reveals that the data used is indeed linearly separable.

k-NN is an instance based machine learning algorithm. During testing it classifies a new point in the feature space based on the majority vote of its nearest neighbors from the training phase. For instance, if among the k-nearest training points class A (e.g. level ground walking) is most common, then the new point is classified as such. In order to obtain the k-nearest points the authors used the Euclidean distance.

$$d(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{2.6}$$

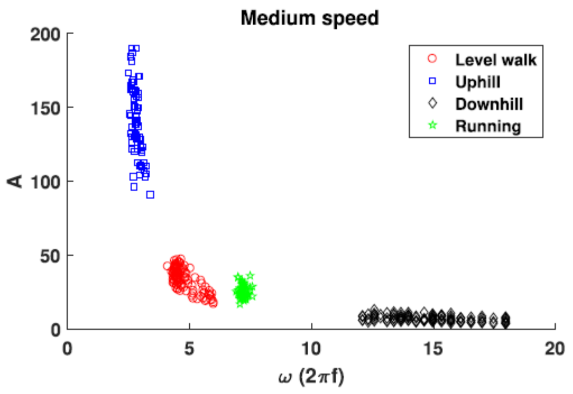


Figure 2.4: $A - \omega$ Feature Space for Medium Speed Walking [11]

Neural Ordinary Differential Equations (Neural ODEs) is a fairly recent approach in the area of Neural Networks to analyze time series data in particular. As the name implies it combines a Neural Network with Ordinary Differential Equations, which are used almost everywhere by scientists and engineers to analyze physical processes in nature. Rather than trying to solve an ODE, a Neural ODE seeks to estimate the ODE itself, specifically by estimating its coefficients. Therefore, it is ultimately a system identification tool that allows modelling the dynamics of a natural system. As such it appears to be a very powerful tool to analyze a multitude of physical processes, which are too complex for classical system identification tools. For example, Neural ODEs were used to forecast weather patterns and urban traffic flows [9; 77]. Generally, a Neural ODE can be defined as

$$\dot{y} = f\left(t, y, \theta\right) \tag{2.7}$$

whereas θ are learnable parameters. Because the result of a Neural ODE are the ODE coefficients, or parameters, learned by the Neural Network, we can then use the ODE to make predictions. Further, ODEs are often time-dependent and as such this approach has built-in support for time series data as is the case with gait data.

2.6 Experiments and Results

The algorithm was tested in controlled and uncontrolled environments. Tests were performed using a treadmill for controlled environments. For uncontrolled environments the subjects performed activities outdoors at their own self selected speed. Subjects were asked to perform a total of three trials of each activity. The first two trials were used to train the machine learning algorithms and the third trial was used to test the algorithms.

The data from controlled environments shows clearly distinguishable classes (activities) in the $A-\omega$ feature space, for example as shown in Figure 2.4 for medium speed walking on the treadmill. The data also shows that the variability is higher for going up (amplitude) and down (frequency) slopes. Additionally, the algorithm is able to distinguish different speeds within an activity with higher variability such as going up hill, see Figure 2.5. Overall for controlled environments the classification accuracy is very high at greater than 97%. While this number feels very high there is room for improvement due to the application. Every time an activity is misclassified the result is an incorrect control signal to the motor, which in turn is felt by the subject. An error rate of 3% therefore corresponds to three "weird feeling" or missed steps for every 100 steps taken. Considering that, on average, people easily take about 5000 steps a day, which corresponds to 150 missed steps, improving the accuracy does provide a clear benefit.

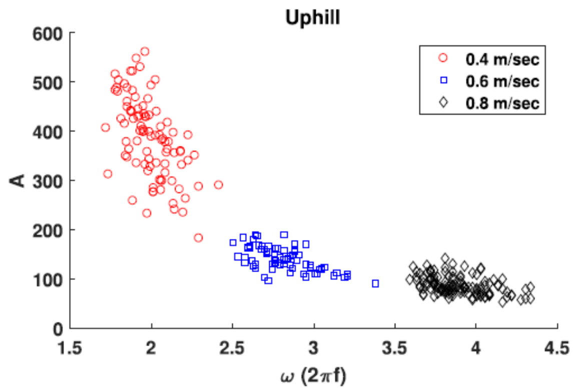


Figure 2.5: $A-\omega$ Feature Space for Different Uphill Speeds [11]

Data from uncontrolled environments, see Figure 2.6, show an increase in variability and therefore a reduction in accuracy to an overall greater than 93%. It can be seen that there is some slight crossover of some points of one class into the other and the separation between the classes is overall less clear than in controlled environments.

The authors also report the lengths of the adaptive time window as well as algorithm recognition speed. As expected, the time window becomes shorter with faster activities. The worst case time window of 218 samples and best time window of 66 samples was recorded at slow uphill walking and jogging in an uncontrolled environment respectively. The average recognition time of the algorithm is 80ms, consisting of a 50ms buffer rate for the peak detection algorithm plus an additional 30ms to compute feature and run activity recognition.

2.7 Summary of Limitations and Improvements

A detailed analysis of Chinimilli's paper has shown several possible improvements of their work. One crucial area of improvement is the time window that is used for peak detection. As mentioned, all of the methods require a time window and unfortunately it is the same time window that causes the largest delay between the time the person changes the activity and when the algorithm detects it. Note, that the "activity recognition times" quoted in the article appear to be the time for the algorithm to react to a change in the *measured* gait parameters. Let us make the following definitions, namely that t_s is the time at which the subject changes activity

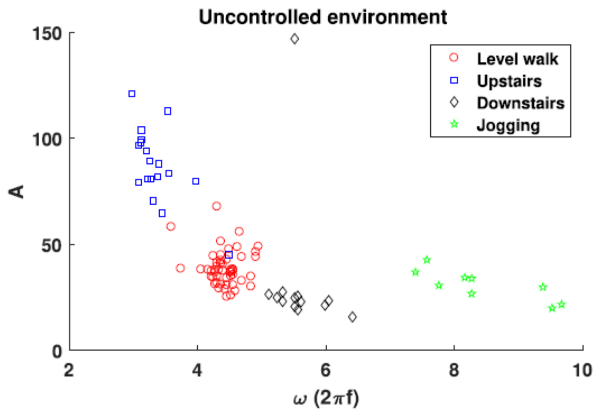


Figure 2.6: $A - \omega$ Feature Space for Uncontrolled Environments [11]

or speed, t_p is the time at which the peak detection algorithm detects the change and t_a is the time at which the entire activity recognition algorithm detects the change, and adjusts the actuation command such that an updated action can be sent to the actuator. The activity recognition time of 80ms specified in the previous section appears to be $T_{pa} = t_a - t_p$ and does not include the time $T_{sp} = t_p - t_s$, i.e. the time it takes for the changed peaks to be detectable. In other words, we are asking for the time it takes for a change in gait, initiated by the person, to show up in the time window via a measurable quantity. When using PSM, the only measurable quantity is a peak, hence the only time a change in gait becomes detectable is once a new peak is detected. And because peaks only occur at the comparably slow frequencies of human walking, running, etc. the delay T_{sp} is very large in comparison to T_{pa}. The worst case occurs when the gait is changed just after a peak has been detected, for instance the person changes from running to walking just after a running peak is detected. In this case, and assuming the person is changing to an average walking speed, the delay $T_{sp} = 1150ms$. Unfortunately, the limitation is simply a characteristic of using the peak search method and cannot be further improved. However, using a different method may yield a different result, whereas Neural ODEs and the FFT appear to be the most promising methods. We shall also point out that other authors appear to leave out the time T_{sp} as well as can be seen from Table VII in Chinimili's paper, so this limitation in itself may be an opportunity for a more detailed study.

A second area of possible improvement is the classification accuracy. In uncontrolled environments the accuracy of 93% means that seven out of every 100 steps taken or on average about 350 steps a day are missed. This may be a difficult problem to solve as the reason appears to be that in the A-ω feature space, there is some crossover between the different activities. One might try to apply filters or other signal processing operations in order to improve separation between the classes.

However, it is conceivable that crossovers are just part of human gait and that it will prove difficult to remove all of them reliably. A second approach then is to identify additional features that result in a wider separation between classes.

Finally, while the authors have covered the most important daily activities, there are activities that could be added to improve performance. Such activities could be sitting down and standing up or assisting when lifting an object or pushing an object. From our recent research at an air force base, we learned that sitting, standing, crawling, and kneeling are important as as well.

Chapter 3

ELECTRONIC SYSTEMS FOR WEARABLE ROBOTIC DEVICES

Every powered robotic device requires electronics in order to function. Similarly to humans and put in the most basic terms, a robotic device must be able to sense, process and affect. To move around their environment, humans mainly use vision, hearing and touch to sense, their brain to process and control, and their muscles to affect. In much the same way, robotic devices use sensors (e.g. position, force, etc.) to sense, microprocessors to process and control, and actuators (e.g. motors, pneumatics, etc.) to affect. This hardware provides *the basis* for controls.

3.1 Requirements for Electronic Systems

Surprisingly, humans are quite efficient and effective at performing the above mentioned functions compared to electronic hardware. Only in the last 10 years or so has the overall electronic hardware made enough technological progress that wearable robotic devices have finally become feasible. However, careful design is still required. What makes wearable robotic devices so much harder to equip with electronics than non-wearable robots? The main reason for this high technological bar is a combination of requirements that are all at odds with each other. These requirements are outlined below.

1. Controls often require sensors capable of measuring motion (gyroscopes, accelerometers)

2. Controls require powerful processors (speed and memory)

3. Actuation requires devices like motors and motor controllers that are able to provide meaningful power (in the order of 50-100W) while being small and lightweight

4. Electronics and actuation must be small and lightweight such that they do not become a hindrance to the user

5. Batteries must be able to store enough energy to avoid frequent charging

6. Software must be realtime capable for safety reasons

7. Wiring and connectors must be robust because they often connect to moving parts

8. Components must be able to communicate reliably in a network because they may be mounted on different parts of the body

9. Components must be able to function in extended temperature environments if a device is to ever make it out of the lab

10. Some wireless capability is important for non-critical functions such as data logging

11. High level controls are greatly aided by high level programming languages (e.g. Matlab, Labview) such that the developer does not need to spend a lot of time worrying about getting the basics to run

The item with the most drastic effect is the size requirement (number 4). For instance, motor size and weight increase with power. At the same time, the current and voltage required for the motor increase, which in turn leads to increases in the size and number of electronic components and PCB area. Additionally, satisfying extended temperature environments has the exact same effect on electronics size. More powerful processors also require more power, resulting in additional heat being generated. Wireless devices like Bluetooth or Wifi require power-hungry antennas to transmit data, demanding more current. Connectors and cabling that are robust to environmental influences are larger and heavier compared to those that do not need to fulfill these requirements. As can be seen, most of these requirements are in conflict with each other. Therefore, approaching this problem with anything other than a *Systems* perspective is likely to produce a sub-optimal outcome or failure.

Over the past five to ten years, I have harnessed the technological progress made in the field of electronics to develop electronic systems that meet the requirements outlined above. In Section 3.2 the design of wearable electronics is shown from a systems perspective. In Section 3.3 and Section 3.4 two such systems developed for exoskeletons are presented.

3.2 A Systems Perspective

This section analyzes the design of electronics systems for wearable robotics from a systems perspective. While this section is presented in a sequential form, the intention is not to imply that this should always be adhered to at all cost. Rather, the goal is to show the most important pieces of the puzzle and how they interact. Each design problem is unique and it is expected that the concepts presented in this section will need to be adapted and iterated upon.

3.2.1 Important Electrical Relationships

All electrical and probably many mechanical engineers are taught the basic electrical relationships mentioned in this section. However, their application when designing small wearable devices are not always obvious. For reference, the symbols used in this section are “U” for voltage, “I” for current, “R” for resistance and “Z” for impedance. Bars are used to denote complex quantities for AC analysis.

Firstly, and most importantly there is of course Ohm’s Law, which says that the product of resistance and current of a 2-pole electronic component (a “2-pole”) is equal to the voltage across it, see below.

$$U = R \cdot I \tag{3.1a}$$

$$\bar{U} = \bar{Z} \cdot \bar{I} \tag{3.1b}$$

Secondly, the power law should be mentioned, which says that the power across a 2-pole is equal to the product of voltage and current. We can also use Ohm’s law to express the power law in different ways only using one of the two quantities voltage or current, the most common one being current.

$$P = U \cdot I \tag{3.2a}$$

$$P = I^2 \cdot R \tag{3.2b}$$

Note that with the exception of equation Equation 3.2b, all of the above equations hold for DC as well as AC signals. Let us examine a purely resistive 2-pole with value $1k\Omega$ through which flows a current of 1mA. It is obvious that, (1) this resistor produces a negligible amount of heat (about 1mW) and (2) a voltage drop occurs across this resistor in the amount of 1V. Now, lets consider a wire of diameter AWG16 ($1.31mm^2$) and length of 2m to supply a motor controller and motor with a current of $10A_{RMS}$ and 40A peak. It is obvious we must consider that the wire is sized correctly for the amount of current going through it. Established literature [58; 64] shows that the wire is most likely appropriately sized for the current it is carrying. What is less obvious is that there is a considerable voltage drop due to the long length, 4m round trip, and large currents. At 10A we will see a voltage drop of 0.5V and 2V at 40A respectively. These voltage drops cannot be automatically neglected because they reduce the available voltage and, hence, speed of the motor under various conditions. And because wearable robotic devices have to be designed close to the limits to be feasible, we must take the drops into account. Otherwise, we may discover later that the motor cannot reach the required speed under certain conditions.

Finally, there are the laws for inductive and capacitive 2-poles. The former states that the voltage across an inductor is equal to the inductance times the change in current over time. The latter states that the current through a capacitor is equal to the product of capacitance and change in voltage over time.

$$u(t) = L \cdot \frac{di(t)}{dt} \tag{3.3a}$$

$$i(t) = C \cdot \frac{du(t)}{dt} \tag{3.3b}$$

Applying these laws is obvious if one is confronted, for instance, with a simple resistor capacitor (RC) low pass filter, see Figure 3.1. Using Ohm's law, Kirchhoff's law and

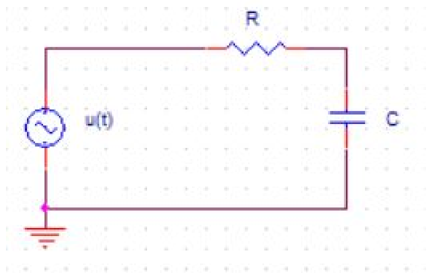


Figure 3.1: RC Low Pass Filter Circuit

the capacitor differential equation above, one can derive the differential equation governing this simple RC circuit.

$$\frac{di(t)}{dt} + \frac{1}{RC}\, i(t) = \frac{1}{R}\, u(t) \tag{3.4}$$

It is less obvious that the same RC filter could apply to a longer wire route on a PCB or a cable. As a cable or even a stretch of conductor on a PCB gets longer, it picks up parasitic capacitance and inductance. This capacitance is the main reason for the length limitations of I2C, which will be discussed later.

Sometimes it is easier to think about 2-poles in the AC domain. Equation 3.3a and Equation 3.3b can be used to derive the impedance for the inductor and capacitor by taking the Fourier Transform of the equations.

$$\bar{Z}_L = \frac{\bar{U}}{\bar{I}} = j\omega L \tag{3.5a}$$

$$\bar{Z}_C = \frac{\bar{U}}{\bar{I}} = \frac{1}{j\omega C} \tag{3.5b}$$

The impedance for an inductor and capacitor reveal new ways to think about them. One can think about an inductor as having essentially zero impedance to DC currents and high impedance to high frequency currents. On the other hand, capacitors have essentially zero impedance to high frequency currents but high impedance to low frequency currents. In other words, inductors can, roughly, be thought of as being a short for DC currents while capacitors can be thought of as being a short for AC currents.

Let us now consider a continuation of the practical case above, where in addition to the motor and motor controller, a micro-controller PCB is providing control logic for a wearable device. The supply voltage should be able to range from 12V - 36V and the components on the PCB draw a combined current of $0.5A_{nom}$ and $1A_{peak}$ at 5V. The PCB is supplied 12-36V using an cable of 2m length. In this case a designer should recognize that the large span of the external supply voltage range provides for a number of inefficiencies. The most obvious one is that the required DC-DC converter has different efficiencies depending on input voltage, output current and step-down size. Less obvious is that the inductance of the cable together with the span in supply voltage necessitates a “worst case” design for the input capacitors of the DC-DC converter. Calculating the inductance even of simple shapes like straight wires is involved, which is why inductance is often confirmed through testing. However, we can get an idea of the inductance of the supply wire using an established formula for the simple assumption of a straight wire [72]. Using a cable length of 2m and cross-section of $1.31mm^2$ we arrive at a wire inductance of $L_w = 3.2uH$[1]. Knowing the inductance and assuming a conversion efficiency of 100% (unrealistic but sufficient in our case to illustrate the point) and switching frequency of 700kHz (common value) we can apply Equation 3.3a in a discrete way to calculate the input voltage ripple at the two extremes of the input voltage span.

$$U_{pp}@12V_{in} \approx L_w \frac{\Delta I}{\Delta t} \approx 0.94V \tag{3.6a}$$

$$U_{pp}@36V_{in} \approx L_w \frac{\Delta I}{\Delta t} \approx 0.31V \tag{3.6b}$$

DC-DC converters can usually tolerate about 2% input voltage ripple, which results in $U_{pp,max} \approx 0.24V$ for 12V input voltage and $U_{pp,max} \approx 0.72V$ for 36V respectively. Note that in the case of 12V input voltage, the ripple is *highest*, while the tolerance for ripple is *lowest*. For robustness, the design will have to be based on the worst case input voltage (12V) and result in the addition of significant bulk capacitance at the input to counteract the effects of the “hidden” wire inductance. These capacitors will add size, both square area and volume, as well as complexity to the PCB. The capacitors will be mostly superfluous when an input voltage of 36V is applied but it is impractical to simply “leave them off” for that case and even if they were taken off, the footprints would remain.

This view is, of course, a greatly simplified view of the complete design and there

[1]In the following equation l denotes the length of the wire pair, r denotes the radius of the conductor and μ_0 denotes the magnetic permeability of free space ($\approx 400\pi\, nH/m$). $L = \frac{\mu_0}{2\pi}\, l \left[\ln \frac{2l}{r} - \frac{3}{4}\right]$

are a myriad of other factors to consider. However, it illustrates nicely the "hidden" cost of flexibility.

As can be seen from this section, the difficulty when designing electronic systems for Wearables lies in the details. Designers must know when quantities are negligible and when they are not. Moreover, one must always pay attention to "hidden" quantities. Unfortunately, we either know from experience that these quantities are usually too small to matter, or we have been told too many times that they usually are. As a result, we tend to unconsciously ignore they exist. However, knowing about these nuances is not enough. In most cases, compromise is necessary to find a suitable design and model based design (MBD) is key to finding the optimal approach (compromise) for a given problem.

3.2.2 The Energy Source

In this work, we limit ourselves to the DC or EC motor as the choice of actuation technology for the reasons presented in Section 3.2.3 below. Therefore, the energy source is most likely a battery or, in rare cases, a tethered power supply.

The most important limitation from a design and safety perspective is the maximum voltage. In general, voltages below 50VDC are accepted as safe because even under unfavorable conditions, e.g. wet hands, the resulting current is less than the current, which results in irreversible muscle contractions and heart fibrillation, see IEC 60364-4-41 and 60479-1.

When batteries are used as the power source, it is important to consider that, when newly charged, the battery voltage can exceed the nominal voltage by many volts. Therefore, the maximum voltage when charged must be used to check against the allowable voltage of 50VDC to ensure the system is safe in all conditions. In most cases, Li-Ion batteries will be the best solution due to their wide availability, reliability and energy capacity. At the time of writing, it is still advisable to stay away from Li-Polymer type batteries because of their instability and safety concerns.

3.2.3 The Drive

In the current state of technology the DC or EC motor is arguably the best option for actuation of wearable robotic systems. This is mainly because of the way electrical energy can be stored as well as their ease of control. The main disadvantage of pneumatics and hydraulics as alternative actuation methods is that they require large, high pressure containers of air or oil. While the power-to-weight ratio of pneumatic or hydraulic actuators (excluding the tank) often exceed DC/EC motors, this advantage vanishes once the energy source (tank, compressor, etc.) is taken into account. Hydraulics especially, often also have safety concerns due to the high pressures involved. On the other hand, pneumatics pose control problems as both slow and fast precise movements are difficult. Other types of electrical motors such as linear or stepper motors cannot reach the power-to-weight ratios of DC/EC motors.

In contrast, DC and EC motors are available in a wide range of voltage and current configurations and all they need is an electrical battery. The combination of technological progress made in motor, motor control and battery technology over the last 10 years has made DC and EC drives a suitable choice for wearable robotics. For instance, in the HeSA project, see Section 3.3, we have used commercial drill batteries to power the device. As mentioned previously, Li-Ion batteries now have energy storage and current capacities large enough such that they can power a device for multiple hours or even a whole day. Therefore, Li-Ion batteries, improved motors and motor controllers provide a feasible drive solution. The challenge, however, becomes to design a suitable transmission because DC and EC motors provide power in the form of high speed/low torque whereas the human body requires the opposite. Hence, transmission design and motor selection becomes critical to overall design success and it is often sensible to start there.

When designing the drive the end goal is to choose a suitable motor, motor controller and transmission that fit within the voltage and current envelope of the battery. While this seems straight forward it almost always results in an iterative process where different combinations of solutions have to be analyzed. For example, design A may consist of a motor, which has reached its maximum speed. However, both battery voltage and transmission would allow for further increases in speed. Choosing a new motor will immediately result in redesign of the transmission and recalculation of the entire design.

Design Limits

Design limits are often misunderstood and poorly taught. In the most general sense a DC or EC motor has *only two* hard design limits: the maximum winding temperature and the maximum permissible speed, therefore

$$T_w \leq T_{w,max} \tag{3.7}$$

$$\omega \leq \omega_{max} \tag{3.8}$$

The maximum permissible speed simply stems from the bearings used to hold the shaft. On the other hand, the maximum winding temperature stems from the insulation used on the winding conductors and ensures that the insulation does not break down abruptly or over time. Unless the design has known thermal characteristics or extensive thermal modelling is possible, it is possible to reduce the temperature requirement to a continuous (RMS) current requirement such that the two hard limits become

$$I_{RMS} \leq I_{max,cont} \tag{3.9}$$

$$\omega \leq \omega_{max} \tag{3.10}$$

Note that voltage is not a hard limit. This is because voltage does not cause any immediate adverse effects. How voltage is to be interpreted depends also on the manufacturer. Often it is a way of distinguishing between different windings, i.e. combinations of speed and torque constants. The voltage can also be used as a nominal specification, i.e. the motor will output the specified nominal power at the nominal voltage and current. While it is best to consult with the manufacturer, for DC motors especially, it is advisable to stay below two times the given nominal voltage.

$$U \leq 2\, U_{nom} \tag{3.11}$$

This is because DC motors use brushes and as the voltage gets higher, the risk for arcing increases, leading to reduced brush life.

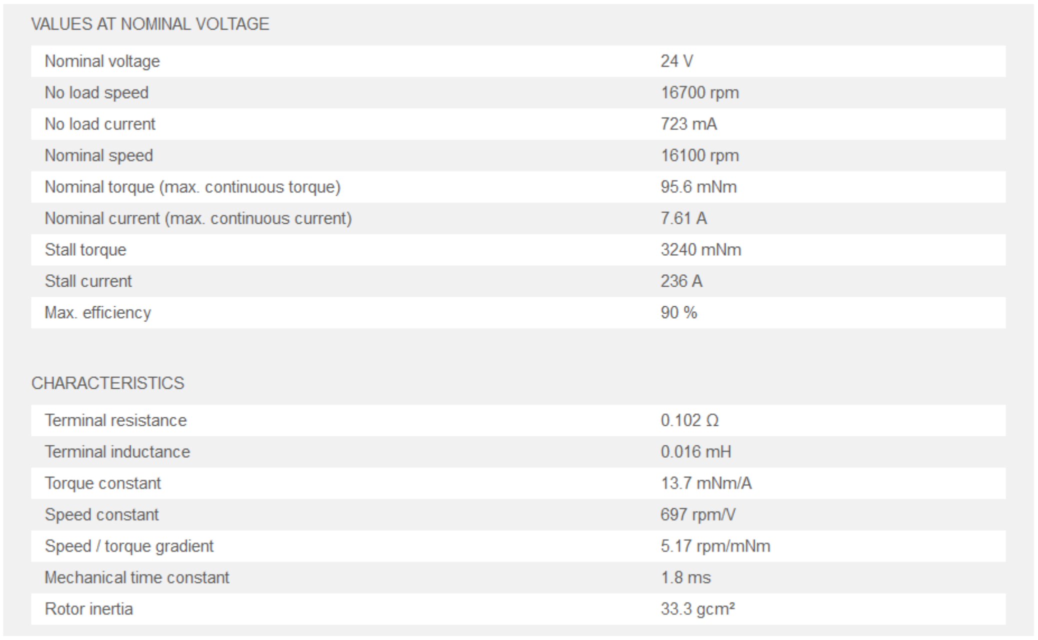
Brushed (DC) Versus Brushless (EC)

Generally, brushless motors have higher power-to-weight ratios than brushed motors. They also have lower rotor inertias because mass is kept closer to the center of rotation and better heat dissipation because the windings are close to the housing of the motor. This will lead many designers to believe that brushless motors are the best for most applications. Unfortunately, this is not true.

Firstly, the majority of brushless motors provide this larger power-to-weight ratio at larger speeds, making the dilemma of finding a workable transmission worse. Secondly, brushless motors often have low values of winding inductance. This can cause issues with heating and controls due to excessive current ripple. Thirdly, brushless motors require hall effect sensors for commutation and result in greater wiring effort and complexity.

For instance, compare a 24V Maxon EC 4-pole 30 brushless motor to a Maxon RE-40 brushed motor. The basic data for both motors is shown in Figure 3.2. With a power of 200W and a weight of 300g, the brushless motor has a nominal power-to-weight ratio of 667W/kg. The brushed motor, on the other hand, with 150W and 480g has a nominal power-to-weight ratio of 313W/kg. The difference is astonishing, in fact, due to this characteristic the EC 4-pole 30 could be called Maxon's flagship of brushless motors. However, closer examination reveals some significant disadvantages of the brushless motor. Most importantly, the brushless motor will run at more than twice the speed of the brushed motor at the same voltage. As a result, a factor of two larger transmission is required, which is difficult to achieve without the use of a gearbox and adding significant weight back in. Secondly, due to the extraordinarily low resistance and inductance, this motor requires large currents in order to achieve its torque. This can be seen in the starting current specification of 236A and continuous current of close to 8A. Large currents in particular will wreak havoc throughout the entire design due to the most basic electrical relationships between resistance, capacitance, inductance and voltage as shown earlier in Section 3.2.1. Thirdly, and, in comparison, perhaps least consequential, this motor requires five additional wires to connect. Nevertheless, one must take care not to underestimate the addition of wires to a design. More wires in turn lead to larger (or a larger quantity of) connectors in addition to the necessary cabling and therefore adds size to the design. Most importantly though connectors and connections in general are potential points of failure.

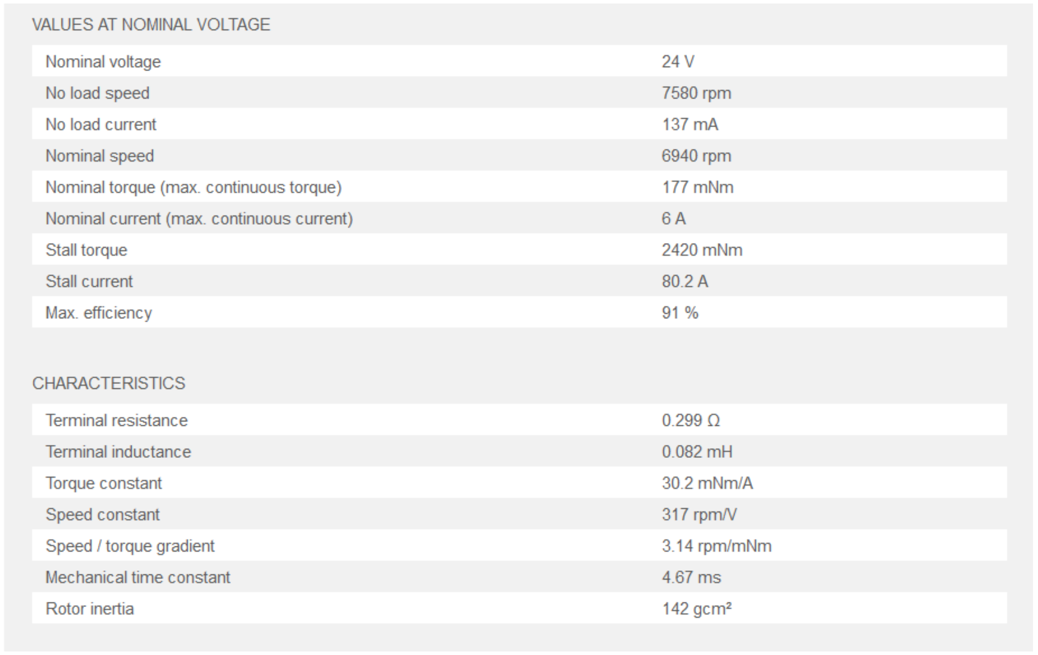
The goal of this comparison is not to disqualify this brushless motor or brushless motors in general, but rather to show that any motor design must be carefully consid-



VALUES AT NOMINAL VOLTAGE	
Nominal voltage	24 V
No load speed	16700 rpm
No load current	723 mA
Nominal speed	16100 rpm
Nominal torque (max. continuous torque)	95.6 mNm
Nominal current (max. continuous current)	7.61 A
Stall torque	3240 mNm
Stall current	236 A
Max. efficiency	90 %

CHARACTERISTICS	
Terminal resistance	0.102 Ω
Terminal inductance	0.016 mH
Torque constant	13.7 mNm/A
Speed constant	697 rpm/V
Speed / torque gradient	5.17 rpm/mNm
Mechanical time constant	1.8 ms
Rotor inertia	33.3 gcm²

(a) Basic Specifications for Maxon EC 4-Pole 30 Brushless Motor



VALUES AT NOMINAL VOLTAGE	
Nominal voltage	24 V
No load speed	7580 rpm
No load current	137 mA
Nominal speed	6940 rpm
Nominal torque (max. continuous torque)	177 mNm
Nominal current (max. continuous current)	6 A
Stall torque	2420 mNm
Stall current	80.2 A
Max. efficiency	91 %

CHARACTERISTICS	
Terminal resistance	0.299 Ω
Terminal inductance	0.082 mH
Torque constant	30.2 mNm/A
Speed constant	317 rpm/V
Speed / torque gradient	3.14 rpm/mNm
Mechanical time constant	4.67 ms
Rotor inertia	142 gcm²

(b) Basic Specifications for Maxon RE40 Brushed Motor

Figure 3.2: Comparison of a Brushed and Brushless Motor With Similar Voltage and Power Specifications.

ered. For some applications, the extraordinary power-to-weight ratio may be worth dealing with the extensive (and expensive) list of downsides. For other applications, the right choice may be to find a way to make the RE40 or a different brushed motor work.

3.2.4 Component Communication

The communication between different parts of a wearable device is another important factor of consideration. Because Wearables often span more than one limb, for instance a hip exoskeleton providing power for both left and right hips, communication across common distances on the human body are unavoidable.

At this moment, of course, wireless communication is on everyone's mind. In order to further examine wireless versus wired communication a differentiation between the purpose of the transmitted signal is made. Two groups are distinguished, signals, which are essential for control, i.e. without them the device cannot perform its basic functions, and signals, which are auxiliary, i.e. signals that augment the basic control algorithms but the device does not depend on them for the most basic functionality. For example, let us assume a hip exoskeleton uses IMU information from the torso in order to provide basic walking patterns to two motors at the left and right hip. This IMU information would be considered *essential.* In contrast, let us now assume that the same device contains a "base algorithm", which ensures a safe gait pattern based on internal (or wired) sensors, while the IMU information is used to provide additional functionality, e.g. the ability to recognize more activities. In this case, the IMU signal would be considered *augmenting.*

Essential Signals - Wired

Unfortunately, none of the currently available *wireless* technologies are suitable to transmit control signals, which are necessary for the device to perform its most basic functions. Wireless, as we all know, is not reliable enough for these types of signals. Disconnections can occur at any time for one of a multitude of possible reasons, for instance, because the person passed by a metal object such as a rail. It would be unwise to transmit these important control signals using a wireless interface. If the connection is lost, the device will abruptly cease to function correctly and potentially put the user at risk if the motors make an unexpected move due to the loss of signal.

For these signals it is highly advisable to use a wired connection for safety and reliability and there are a number of choices. Some of the most common interfaces are listed in this section, which is not an exhaustive list. Table 3.1 shows some of the most important types of communication as well as their basic characteristics.

	Type	**Direction**	**Speed**	**# Wires**	**Distance**	**Bus**
UART	Asynchronous	Full Duplex	10MBit/s	2	Short	No
RS485	Asynchronous	Full Duplex	10MBit/s	4	Long	Yes
		Half Duplex	10Mbit/s	2		
I2C	Synchronous	Half Duplex	5MBit/s[2]	2	Very Short	Yes
SPI	Synchronous	Full Duplex	80MBit/s	4+	Medium	Yes[3]
CANbus	Asynchronous	Half Duplex	1MBit/s	2	Long	Yes
EtherCAT	Asynchronous	Full Duplex	100MBit/s	8	Long	Yes

Table 3.1: Important Types of Wired Communication for Wearables

[2]Depending on the mode of communication 0.1, 0.4, 1, 3.4 and 5MBit/s are supported.

[3]Technically, SPI is bus capable, however connecting and communicating with several devices in a network is complex and very inefficient.

Note that the speeds listed specify the upper end of the range and both, speeds and lengths are only approximate numbers and are of course very interdependent as well as environment dependent. For instance, at tens of kBit/s UART may well be able to transmit over many meters. However, such a low speed is not useful for most wearable device applications. We also limit ourselves to serial communication because it is very widely used while parallel communication is less common due to the large (usually ≥ 8) number of required wires.

UART The UART (Universal Asynchronous Receiver Transmitter) interface is one of the oldest forms of serial communication. Despite its age, however, it is still widely used due to its flexibility and simplicity. We should clarify that "UART" is a hardware device and does not by itself specify a voltage level (physical layer) or protocol. In fact, RS232, RS422 and RS485, which specify the physical layer, use the same UART hardware module. When the term UART is used what is usually meant is TTL (transistor-transistor-logic) levels. However, many micro-controllers, even if they are based on a lower voltage such as 3.3V, provide the UART Rx and Tx lines on 5V tolerant pins for ease of integration. UART provides full duplex communication at speeds up to about 5-10 MBit/s depending on length. When using speeds in the high kBit/s and in the MBit/s range lengths are of course limited and in any case only two devices can communicate over one pair of UART lines. However, a big advantage of UART is that it uses only two wires while the designer can use whatever protocol they find best for the application. UART is also ubiquitous and can be found on most micro-controllers. Due to the short lengths it is best used for on-board communication and communication between components that are located close (within a few 10 cm) to each other.

RS485 As mentioned above, RS485 is a physical layer (signal level) specification for UART. It adds differential signaling along with stronger drivers and therefore can reach much longer cable lengths and is more immune to noise. However, it also requires four wires instead of two for full duplex operation due to differential signaling. RS485 also adds the capability of supporting at least 32 devices [73] on the same line. As an approximate guideline one can use the rule that the length in meters times the speed in Bit/s should not exceed 10^8 [73]. As such, RS485 easily supports a length of several meters at a speed of 10 MBit/s and would be well suited to transmit essential data between different places on the human body.

I2C The Inter-Integrated-Circuit (I2c, I^2C, IIC) is a standard developed by NXP Semiconductors. I2C, a master-slave bus, provides half duplex, bus capable serial communication using two wires at speeds up to 5 MBit/s. However, the speed depends on the mode of communication supported by the slave devices and the slave with the slowest mode of communication determines the maximum speed for the bus. In reality, most devices support speeds of 1 MBit/s and almost all devices support 400kBit/s. As mentioned, this interface requires one device on the bus to act as the master, which configures the slaves and controls all communication. A particularly strong point of I2C is that a master can configure a bus with multiple devices using only two wires because addressing is built into the I2C protocol. Further, because I2C is a standard all available devices follow the same general protocol. Therefore, talking to a sensor is not all that different from talking to a memory chip. On the other hand, this also reduces flexibility and adds some inefficiency. Physically, I2C uses simple bi-directional open-collector or open-drain lines and as such is a inexpensive interface available on most micro-controllers with a multitude of available slave devices from memory to sensors or even ADCs. However, as the name implies, lengths are limited due to the simple driving circuit. Although booster chips are available to extend the range, this interface is best suited for on-board communication and communication with devices close to the main PCB.

SPI The Serial-Peripheral-Interface (SPI) developed by Motorola is in many characteristics the complement to I2C and also a Master-Slave bus. Unlike I2C, SPI uses push-pull drivers and therefore is capable of larger speeds and longer wire lengths. In fact, SPI does not have a maximum clock limit other than the limit imposed by the design and environment. In practice, this means that speeds of 80 MBit/s are not uncommon [2] and SPI provides full duplex communication at these speeds. However, this high bandwidth comes at the price of complexity and bulk. SPI requires a minimum of four wires for a bus with two devices, one master and one slave. Each subsequent device (slave) then requires an additional wire or exorbitantly complex communication. Therefore, it is my opinion that SPI is not really a bus and is best left for communication between two devices where speed is a determining factor. Interfacing an ADC at high data rates (in the tens of kSamples/s or more) or communication between two micro-controllers are just two examples where SPI is the perfect fit. Further, SPI is not an official but a de-facto standard and hence offers great flexibility in communication protocol.

CANbus The Controller-Area-Network Bus (CANbus) was developed by Bosch GmbH, has been in use for decades and is still a "go-to" interface in automation and control. Originally developed for vehicles, it is both robust and minimalist in size and its only limitation is speed [52]. CANbus is a multi-master bus and supports half duplex communication at 1 Mbit/s using only two wires. Together with the CANopen software layer it is used for many automation tasks but first and foremost for motor control. Due to a robust physical layer using differential signaling, CAN-bus is capable of a maximum speed of 1 Mbit/s over lengths of tens of meters. The CANopen communication protocol provides device communication profiles optimized for embedded systems and in a real-time environment. NMT messages and SDOs allow device configuration from the Master, PDOs facilitate real-time communication and, Emergency Objects and heartbeats provide safety mechanisms. SYNC objects are particularly important for robotics because they allow for precise synchronized motion of multiple axes. Finally, the use of CRC error detection codes is part of its robustness. However, as mentioned above these advantages come with the disadvantage of speeds limited at 1 MBit/s, whereas RS485 and, in particular EtherCAT, allow an increase in speed by up to two orders of magnitude at full duplex communication; and at the cost of size and power as we will see. In summary, the main advantage of CANbus are robust, real-time communication at decent speeds using a minimal amount of wires and small amount of power compared to EtherCAT. These are the reasons why CANbus is still used so frequently today and almost all higher level motor controllers provide this interface. Therefore, unless high required data rates or a large amount of sensors reduce its feasibility, CANbus is well suited to provide communication for remote devices around the human body.

EtherCAT Ethernet for Control Automation Technology (EtherCAT) was developed by Beckhoff Automation, is standardized (IEC61158) and is also suitable for real-time automation control [24]. As the name implies, EtherCAT is the adaptation of Ethernet to the requirements of automation and control. Its main advantage over CAN is speed, EtherCAT currently supports 100 MBit/s and could be increased into the low GBit/s range though this is not currently necessary. Theoretically, at this high speed, 100 servo axes can be updated at rates of up to 10kHz, which is incredible. Moreover, EtherCAT supports CAN application protocol over EtherCAT (CoE) such as CANopen and others simplifying integration. Naturally, the large increase in speed comes with a lengthy list of costs. In this case the costs are size, complexity, both for hardware and software, and power consumption.

First, EtherCAT requires four times the amount of wires compared to CAN. This increase is significant because in the wearable device space we do not have the luxury of using the nicely sized RJ45 connector. The RJ45 connector is designed to plug into a stationary router or other network device, and then to be left alone. It was not designed to connect devices mounted to human limbs, which constantly move relative to each other, and to get pulled and abused. A significantly larger, more expensive connector must be chosen.

Second, EtherCAT requires both, more complex hardware and software as is expected for a high throughput, high flexibility interface. The required hardware is likely different for a Master versus a Slave device. Assuming that the Master micro-controller has a MAC interface (usually MII or RMII), which can be found on many 32-bit chips, the main other hardware required is a PHY (Ethernet Physical transceiver). The word "transceiver" maybe be misleading, however, as for embedded systems it usually implies a device of average size and power consumption. Let us consider the TJA1102 from NXP for example, which comes in a relatively high density QFN 56-pin package of 8x8mm. We have just added a second micro-controller to our wearable device, which will increase PCB size and complexity by far above 8x8mm due to required external components and routing. On the Slave side we will most likely need an EtherCAT Slave Controller, which comes in a TQFP 128-pin package of 16x16mm (e.g. Microchip LAN9255). We could also duplicate the setup on the Master side but it is likely that the single Slave chip despite its immense size will result in a more compact soluation. Software complexity will also increase greatly and require a 32-bit micro-controller running a large Ethernet stack. In comparison, a typical CAN transceiver is available in an 8-pin 4x4mm package requiring only a few external components.

Thirdly, the power consumption of the two named transceiver/controller devices is large. The TJA1102 has four supply pins drawing a combined typical current of about 105mA and maximum current of 1.5 times the typical current. The LAN9255 does not list power consumption in its datasheet but we can expect it to be equally power hungry. In comparison, a typical CAN transceiver requires only around 40mA nominal current.

In short, when deciding between CAN and EtherCAT for wearable devices the main deciding factor should be whether the extraordinarily high bit rate is really required. If the answer is yes, one should ask themselves if one or a combination of the following approaches might solve the problem: Are the requirements reasonable or could they be relaxed? When generating requirements one tends to take very large “safety factors” into account due to the unkowns in the system. Could a two CANbus system be fast enough? Can we use RS485? One should put models and numbers next to those answers even if they are crude because of uncertainty. In summary, for wearable devices EtherCAT should be avoided if possible due to the costs, which have a significant, direct negative influence on the general requirements mentioned at the beginning of this chapter Section 3.1. If all else fails, EtherCAT will of course swiftly dispose of a persistent bandwidth problem for most (if not all) projects.

Summary As a final note, we would like to mention that while this section may be seen as a very critical evaluation of EtherCAT, one should remember that we are evaluating options for *wearable devices* only. EtherCAT is a powerful communications interface and will only gain popularity in the coming years because of the continuing increase in sensors, data rates and overall system complexity. If we were building a quadruped robot with three independent actuated axes per leg and a number of sensors at remote locations on the device, the decision to use EtherCAT would be a much easier one. Updating alone of 12 axes and reading their feedback signals at 500Hz will require a bandwidth of above 1 MBit/s. Further, there is more available space and the power requirements by EtherCAT are probably dwarfed by the power requirements already in place from powerful processing and of course 12 motors. The goal of this chapter is to show a systems perspective of electronics for wearables and the foregoing analysis of EtherCAT illustrates the system wide effects and balances well.

Auxiliary Signals - Wireless

If the signal is of auxiliary function, wireless may be a suitable choice. In the second example at the beginning of this section, if the augmenting signal is lost, the basic device function does not get compromised because the base algorithm will continue to provide control signals. Again, this requires careful examination because there are also other points to be considered, the most important ones being power consumption, complexity and approvals. For the purpose of regularly streaming data in real-time, generally all wireless technologies will require more power than a wired connection. As mentioned in Section 3.2.1 the current consumption of a PCB has a real and direct influence on its size through power supplies and capacitors. Therefore, one should think about this when considering adding these modules. On the other hand, collecting data of a wearable device while tethered to a computer is equally unfeasible.

Figure 3.3 shows a comparison of the different technologies available today in terms of speed versus distance. At this point it should be obvious that cellular is out-of-the-question. Unless there is a large compelling reason for doing so, using Wifi

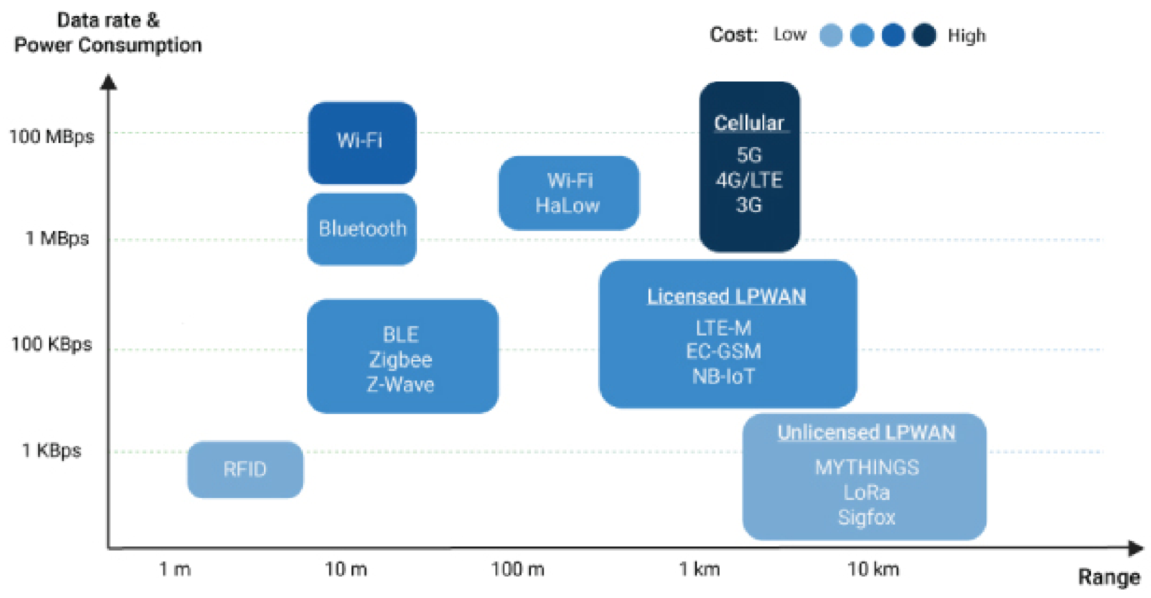


Figure 3.3: Wireless Technologies Speed vs. Distance [63]

is probably equally unfeasible and overpowered. For example, the “JODY-W2” Wifi module from U-Blox [65] requires a transmit current of 240mA to 340mA at a supply voltage of 3.3V, which is equal to one whole Watt of power. In comparison, a power hungry 16-bit dsPIC processor has a nominal consumption of 90mA at its maximum operating frequency.

Bluetooth and Zigbee are probably the best choices overall, providing data rates in the order of hundreds of kBit/s to the low MBit/s range. Their data rates are sufficient for the application and their power consumption much lower. Bluetooth Low Energy (BLE) has gained significant attraction over the past years due to its low energy consumption and is ubiquitous in our daily lives [5; 71]. Wireless headphones, heart rate monitors and other fitness devices, car speaker phones all heavily rely on Bluetooth. Their requirements have obvious commonalities with wearable device applications, for instance similar data rates and tolerance to loss of signal.

3.2.5 Sensors

Sensing is one of the three basic functions mentioned at the beginning of this chapter and hence, every wearable device requires sensors. Sensor technology has developed greatly over the last decade, especially in the area of inertial sensors. For instance, a two-axis gyroscope used to be the size and volume of a human thumb and cost several hundred dollars. It provided an analog signal that was prone to noise while travelling from the sensor to the ADC on a micro-controller. Today, a 9-axis inertial measurement unit (IMU) is almost the size of a rice corn, costs several dollars and provides a digital signal (e.g. I2C) because it also has an on-chip ADC. Figure 3.4 shows a the size of a typical IMU today. The availability of the most basic quantities

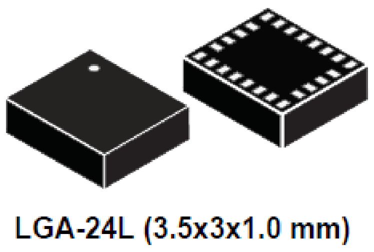


Figure 3.4: Typical Size of an IMU Today

of motion, namely acceleration and velocity, has truly catapulted the state-of-the-art of control algorithms.

While the progress is most obvious in the area of motion sensing, there are also a multitude of other sensors available today, which are smaller and easier to integrate. For example, current sensors are available in 8-pin SOIC packaging and provide resolutions in the 10mA range with on-board basic filtering. Rotary sensors with above 12-bit resolution can be found in below 20-pin TSSOP packaging, requiring only to be next to a magnet and therefore decoupling the cabling between two moving joints. In other words, the good news is that it should not be too difficult to find these basic sensors needed for wearable devices. Therefore, in this section we want to briefly talk about placement and how to interface them.

Interfaces Generally, digital interfaces are preferred over analog ones due to their robustness. Analog signals are susceptible to noise before they reach the ADC and therefore require a bit more attention. Nevertheless, some sensors do not yet provide an integrated digital solution such as force or torque sensors. Care must be taken to keep distances as short as possible and to follow established "good practice" rules. This includes using twisted pairs and shielded cables whenever possible. In some cases we can also choose to use an external or specialized ADC, for example with I2C or SPI interface, close to the sensor and the digital signal for the bulk of the distance. Again, the formulas outlined in Section 3.2.1 are of key importance. A voltage divider is obvious when we are designing one but much less obvious if it is inadvertently added, for instance when biasing an analog signal.

As we add length to cables, the cable can no longer be modelled as a pure resistive element and starts to pick up parasitic capacitance and inductance. This capacitance is the main limitation for I2C. Let us assume that we are using 1 Mbit/s to interface a sensor with an I2C interface using a pull-up resistor of 2.2k. A speed of 1 Mbit/s means that each clock cycle (low/high) of the SCL/SDA line should last about $1\mu s$. The pull-up resistor together with the line capacitance forms a low pass RC-filter with a time constant of $\tau = RC$. We can therefore calculate the maximum allowed line capacitance for the speed of 1 MBit/s.

$$C_{max} = \frac{\tau}{R} = \frac{1}{5Rf_{I2C}} \approx 100pF \tag{3.12}$$

We see that a very small amount of capacitance is enough to make our I2C line look like a low pass filter rather than a wire. This low pass filter will now start to degrade our bit stream from a pulse waveform to a sawtooth and eventually to a constant signal somewhere in between ground and supply voltage. Note, that a pull-up resistor of 2.2k is already fairly strong on the lower end of what the standard allows.

Placement Placement of the sensor is equally important to choosing the right sensor and interface. Generally, sensors should be placed as close as possible to the quantity, which they are required to measure. This rule holds true equally for sensors that measure electrical quantities and mechanical quantities. For example, let us assume we want to measure a joint angle and we have two options of measuring it. First, we can measure it using a linear sensor connected to the joint via a lever arm. Second, we can measure it using a rotary sensor at the joint. Mathematically, these two approaches seem identical. If joint play and flexing are considered they are not, however, and the direct approach using the angular sensor will yield a by far higher fidelity signal.

In electrical engineering, measurement is almost its own discipline for a reason because "how" and "where" a quantity is measured really matters. Consider the simple circuit shown in Figure 3.5, consisting of a voltage source U_0, resistor R_1 and measurement device A, an amp-meter. We want to measure the current flowing through R_1. We must recognize that if we are to expect a correct result for the

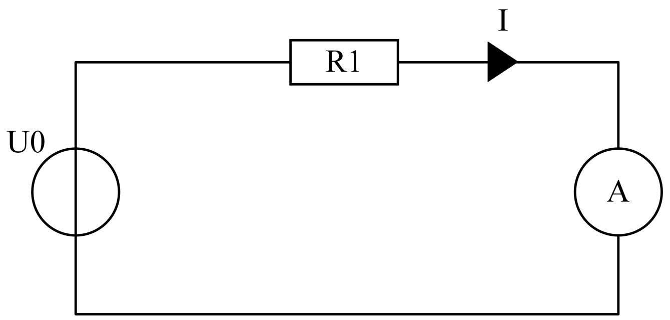


Figure 3.5: Simple Circuit Consisting of Source, Resistor and Amp-Meter

measured quantity I, we must know something about R_1. This is because every measurement device has an internal resistance itself, usually in the range of a few Ohms when measuring current. Therefore, if the value of R_1 is close to the internal resistance of the amp-meter, the measured current will be wrong because we will actually measure the current caused by $R_1 + R_i$, with R_i being the internal resistance of the amp-meter. For example if $R_1 = R_i$ the measurement error will be 50%, the measured current will be half of the actual current.

3.3 Electronics System Developed for HeSA

The Hip Exoskeleton with Superior Assistance (HeSA), shown in Figure 3.6, was designed to apply a significant amount of torque at the hips in order to safely support and augment a person's strength. The device allows people to feel less tired when walking with or without a load (e.g. backpack) for longer distances. Moreover, it allows to carry a heavier load for a shorter duration. One actuator package at each hip, circled in red in the figure, contains all the electronics with the exception of the battery, which is carried on the back. Each actuator package includes a DC motor, belt and screw transmission, motor controller board, sensor acquisition microprocessor board, sensors, as well as mechanical support structures and linkages. A central microprocessor board (Master Controller) is located in one of the actuator packages. Figure 3.7 shows a system level diagram of the chosen electronics system. The motor controller, shown in Figure 3.8 is a custom designed Maxon high power (600W continuous, 1600W peak), high performance DC/EC motor controller in a very small 80x35x12mm form factor. This motor controller can perform all main modes of control (current, velocity, position) for both DC and EC (brushless) motors. Moreover, it provides an array of other higher level control functions such as homing and profile motion. The sensor acquisition microprocessor PCB, shown in Figure 3.9, contains

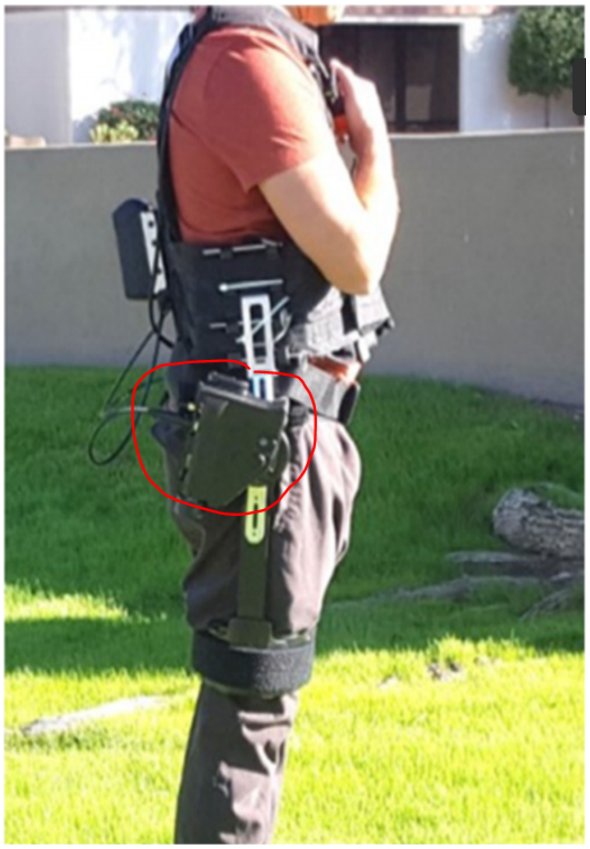


Figure 3.6: The HeSA Wearable Robotic Device [17]

a powerful digital signal processor from Microchip, 9-axis IMU, CAN transceiver, all power supplies as well as basic sensor interface circuitry, LED, push-buttons and connectors in a package of only 24x35x8mm. Finally, the central node microprocessor PCB, shown in Figure 3.10, contains a larger, powerful digital signal processor from Microchip, 9-axis IMU, CAN transceiver, all power supplies, LED, push-buttons and connectors. Additionally, it contains more flexible, expanded I/O circuitry as compared to the sensor acquisition PCB. For instance, it contains multiple conditioned analog channels, some of them capable of interfacing a force type sensor (e.g. load cell) as well as an array of different I/O interfaces (e.g. UART, I2C, SPI, CAN and digital I/O) and additional on-board persistent flash memory. The board is directly stackable with the Maxon EPOS motor controller, saving space and wiring effort. The PCB is slightly larger than the sensor acquisition PCB with main dimensions of 35x50x12mm. Both, sensor and central node PCBs can be programmed very quickly using Matlab/Simulink.

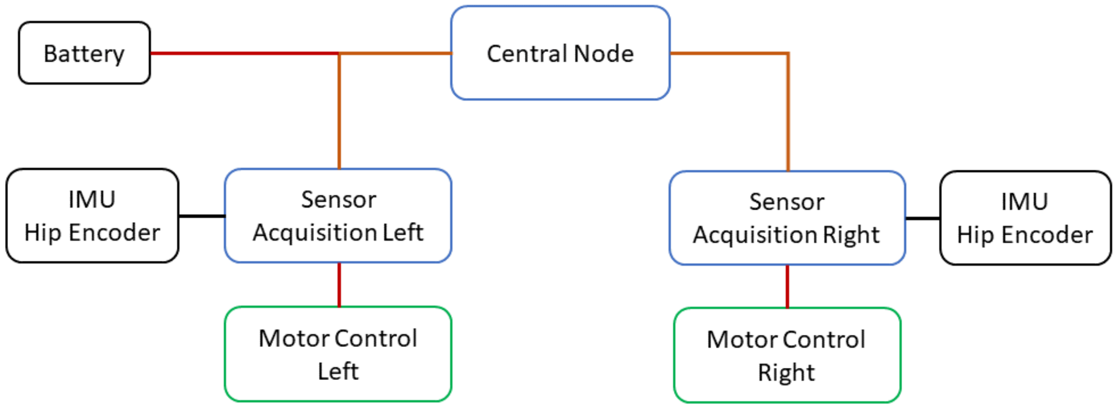


Figure 3.7: System Level Diagram of HeSA Electronics; Black: Sensors; Green: Motor Control; Blue: Processing; Brown: Power and CANbus Network

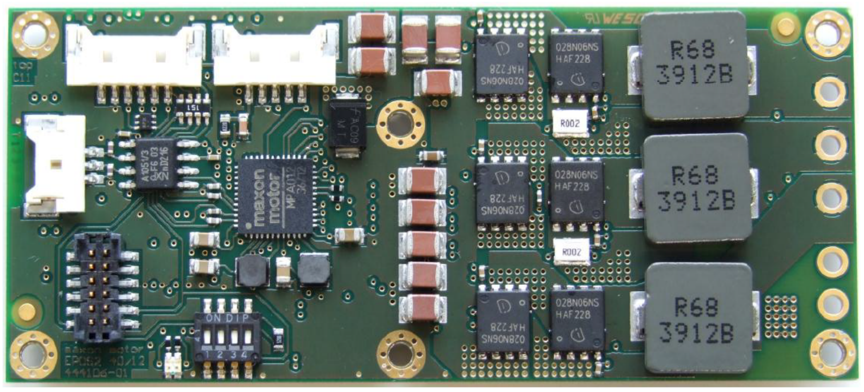


Figure 3.8: EPOS2 High Power Motor Controller

3.3.1 Theory of Operation

Consistent with the requirements outlined in Section 3.1 the HeSA electronics system implements the basic functions of sensing, processing and affecting.

Sensing is performed by an absolute angular encoder and a motor encoder at each hip. In some configurations, one additional sensor acquisition PCB is used on each thigh segment in order to also measure motion data such as thigh angular velocities.

Processing is performed in a distributed manner on the central node PCB, sensor acquisition PCB and motor controller. On the sensor acquisition PCB, the hip angular encoders as well as push buttons are read and basic sensor conditioning, e.g. zeroing,

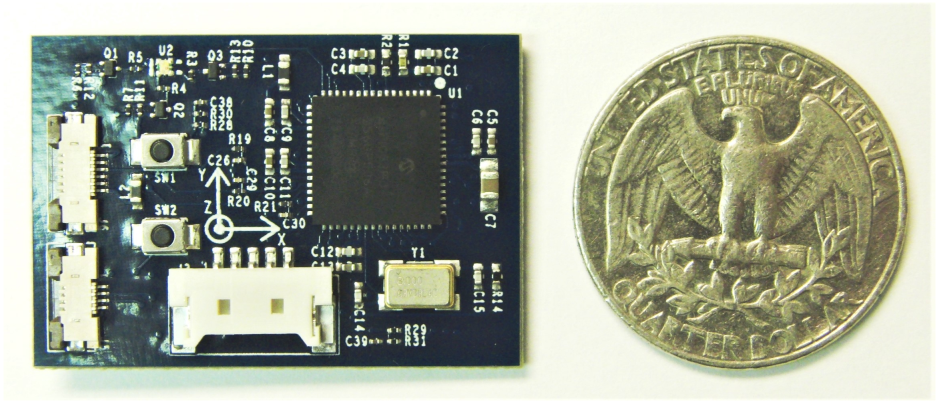


Figure 3.9: Sensor Acquisition PCB

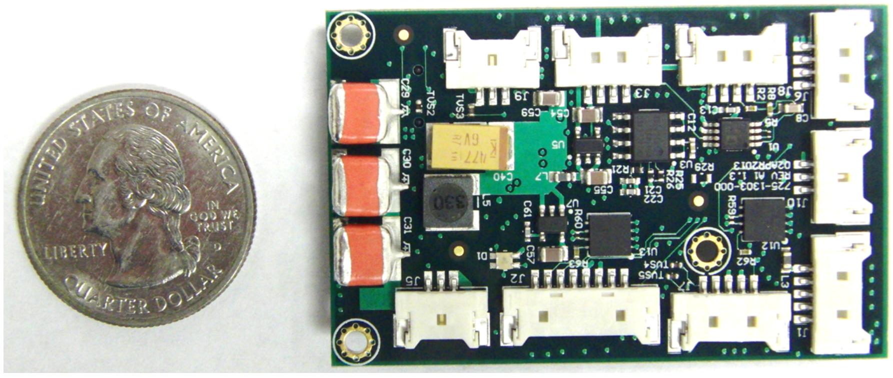


Figure 3.10: Central Node PCB

is performed. The data sensor data is then sent via CAN to the central node PCB. In the use case where two additional sensor acquisition PCBs are used on the thighs, these PCBs acquire nine axes of motion data. 3-D angular velocities, 3-D linear accelerations and 3-D magnetic field strengths are acquired though only the former two are sent via CAN to the central node for processing. Each motor controller is acquiring motor position and current data and sends this data along with status information to the central node via CAN. Moreover, the motor controller is performing all functions associated with motor control, i.e. receiving the command signals from central node, closing the chosen control loop as well as system and safety monitoring. When used in profile mode, e.g. profile position mode, the motor controller can produce smooth paths between a small number of position commands.

The central node is the main processing platform and responsible for the overall system response. First, it receives all remote sensor data from sensor acquisition PCBs and motor controllers and then performs any necessary further signal conditioning, e.g. digital filtering. Second, it runs all necessary higher and lower level algorithms such as activity recognition and path generation. Third, it checks the generated path for feasibility (e.g. limits) and sends the command signal to the motor controller via CAN. Finally, it runs its own high level state machine as well as system monitoring. The high level state machine determines system state in terms of "Initialization", "Homing", "Standby", "Enabled" or "Fault". System monitoring includes monitoring of the motor controller status, main supply voltage status, PCB temperature status as well as some sensor and communication status. If a fault is detected, e.g. low supply voltage due to drained battery, the system is able to alert the user and shut down safely.

Affecting is performed by the motor and motor controller. As mentioned above, the motor controller can receive current, velocity or position commands and ensures that the motor follows these commands as closely as possible.

3.4 Electronics System Developed for APEx

The Aerial Porter Exoskeleton (APEx), shown in Figure 3.11 provides assistance at the hips when lifting or pushing heavy objects.

The electronics system layout is similar to HeSA. A central node microprocessor, left and right hip node as well as a left and right motor controller are all networked in a CAN network. However, in this design an additional microprocessor node, the NextGen MPU, is used to perform high level activity recognition. The NextGen MPU and central node board communicate via UART as they can be located close to one another. Due to smaller power requirements a significantly smaller motor controller, shown in Figure 3.13, is used. This off-the-shelf EPOS2 motor controller from Maxon has the dimensions 25x55x10mm and can provide 4A peak and 2A continuous current up to an absolute maximum voltage of 42V. The motor controller can also be directly mounted onto a motherboard using a standard PCI-e connector.

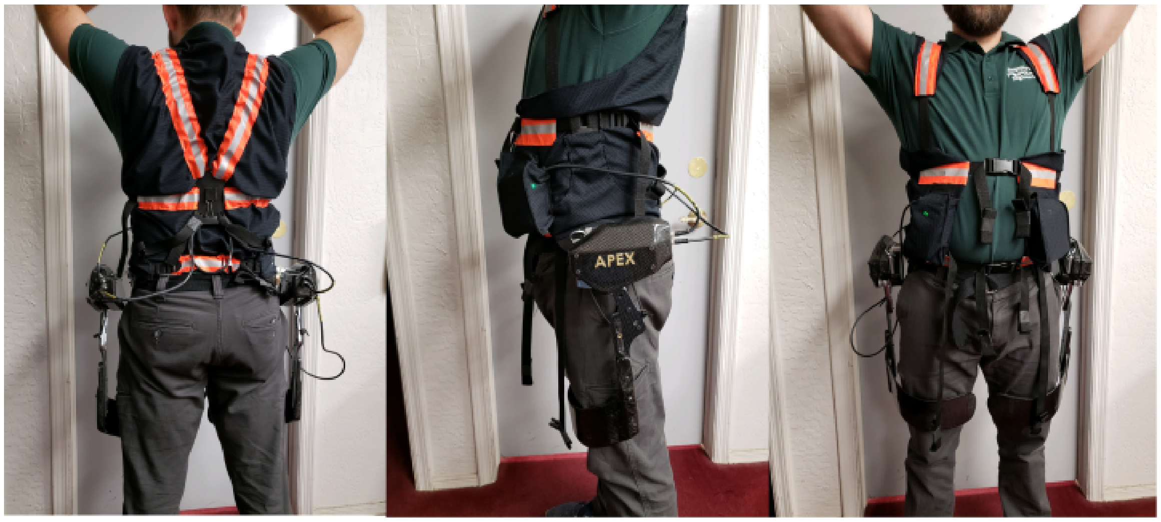


Figure 3.11: APEx Exoskeleton

A new hip node microprocessor board, which is based on the same Microchip digital signal controller used in HeSA, is used as a sensor acquisition PCB on the hips as well as a central microprocessor. It provides the required connector such that the EPOS2 can be mounted directly to it. The stack of hip node and EPOS2 is shown in Figure 3.14.

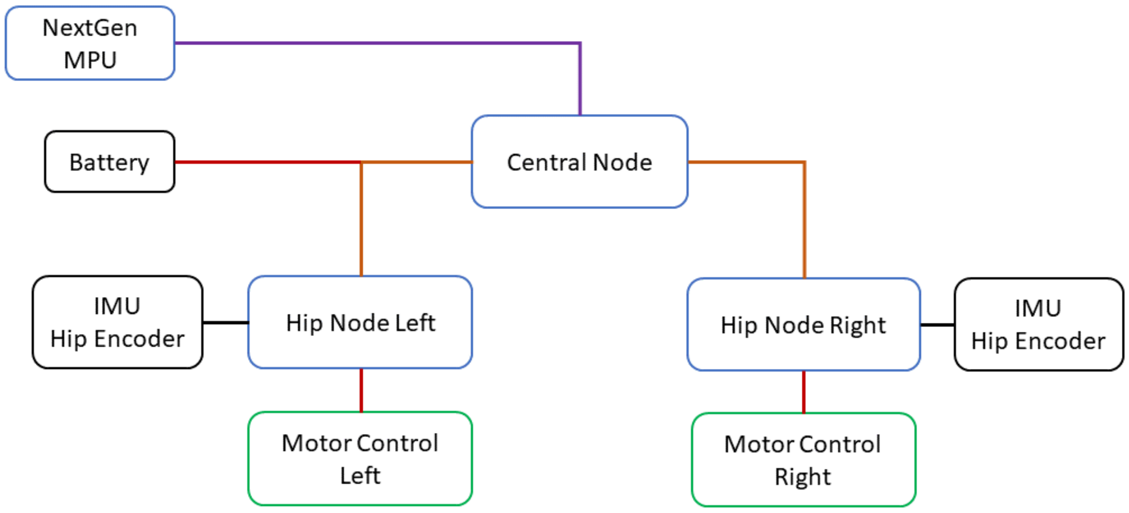


Figure 3.12: System Level Diagram of APEx Electronics; Black: Sensors; Green: Motor Control; Blue: Processing; Brown: Power and CANbus Network; Purple: Power and UART

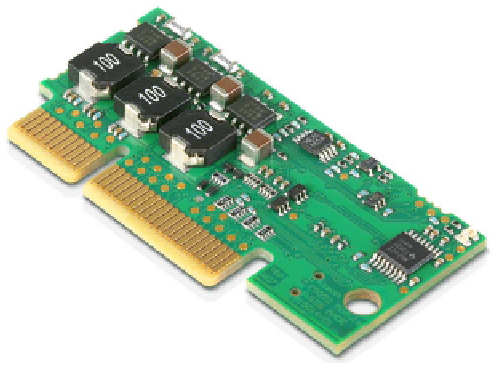


Figure 3.13: EPOS2 Miniature Motor Controller

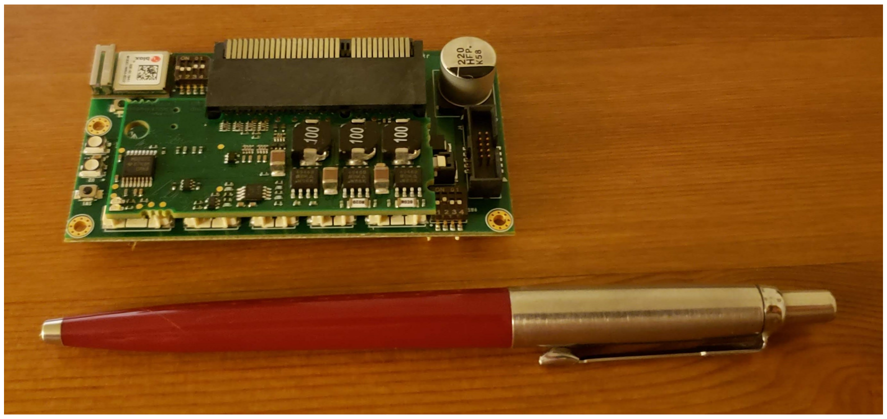


Figure 3.14: Central Node PCB Used on APEx

Chapter 4

CONTROL SYSTEMS FOR WEARABLE ROBOTIC DEVICES

Controls and software are a second set of major disciplines, which are required for a wearable robotic device to perform its functions. In our earlier categorization of the basic functions of a wearable robotic device into sensing, processing and affecting, Controls and Software can be found to perform all three of these functions. The boundary between the two is often fuzzy because controls is part of software and vice versa. For example, designing a PID compensator for a motor position loop is a classical controls problem. However, if this compensator is to be used on real hardware in real-time it helps to also understand how the code, which implements the compensator, is written and executed on the hardware. Even though a PID compensator could be implemented as an analog circuit, in practice today this is done rarely because analog circuits cannot be easily changed. Especially in robotics it is unlikely we will encounter "true" continuous systems in the form of an analog circuit. Micro-controllers and computers in general lack the concept of "continuous" time that human beings are so accustomed to but they have the neat advantage of being able to be reconfigured thousands of times. And, with correct design, the final system performance will be very close to a continuous system. The key to achieving this performance again lies in system wide design and knowing how different variables such as gains, sample times, memory access and even external processes like data transmission from a sensor or the bandwidth of a DC motor interact with each other. Therefore, in this chapter we shall describe control systems for wearable devices from a systems perspective and provide two examples of completed work.

4.1 Requirements for Control Systems

As in the preceding chapter we shall start by listing basic requirements for control systems tailored towards wearable devices. Some of these requirements are directly or indirectly related to the requirements listed in Chapter 3.

1. The code for wearable robotic systems must be able to execute in real-time.
2. For safety reasons, the control system must include basic system monitoring, in other words some implementation of Fault Detection, Isolation and Recovery (FDIR).
3. Control systems should be developed in graphics based, state-of-the-art model based design and auto-code-generation tools such as Matlab/Simulink and others.

4.2 A Systems Perspective

As discussed at the beginning of this chapter, the field of Controls naturally encompasses a wide area of disciplines. In order to design a compensator for a system, first we must understand the system for which a compensator is to be designed. The system could be anything from a high bandwidth voltage control loop for a DC-DC converter to a (relatively) slow bandwidth auto-pilot system keeping a plane on a programmed flight path. In this section we will discuss some important concepts that apply to a range of problems including the obvious stability requirement.

Sampling

In almost all of today's real-world systems we will be implementing controls using a micro-controller or similar device. As mentioned before, these devices lack the concept of continuous time, which is why we will be working in the discrete time domain. In controls, this domain is also commonly called z-domain because of the independent variable being "z" as opposed to "s" in the continuous domain. In consequence, *all* signals from the real world must be transformed into the discrete domain, generally through a process called sampling, see Figure 4.1. Sampling is

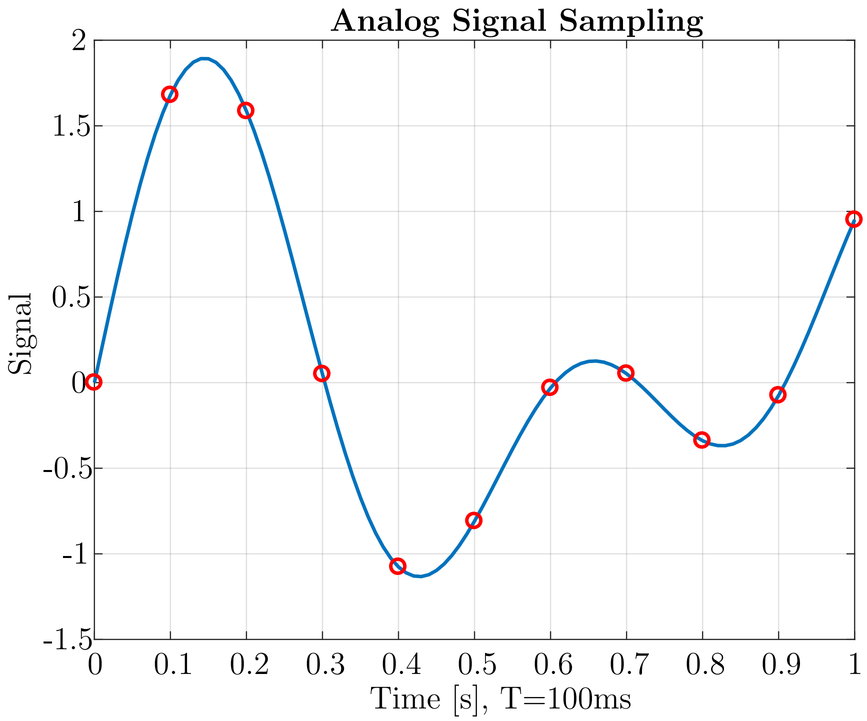


Figure 4.1: Sampling of a Sinusoidal Signal at Intervals $T_s = 100ms$

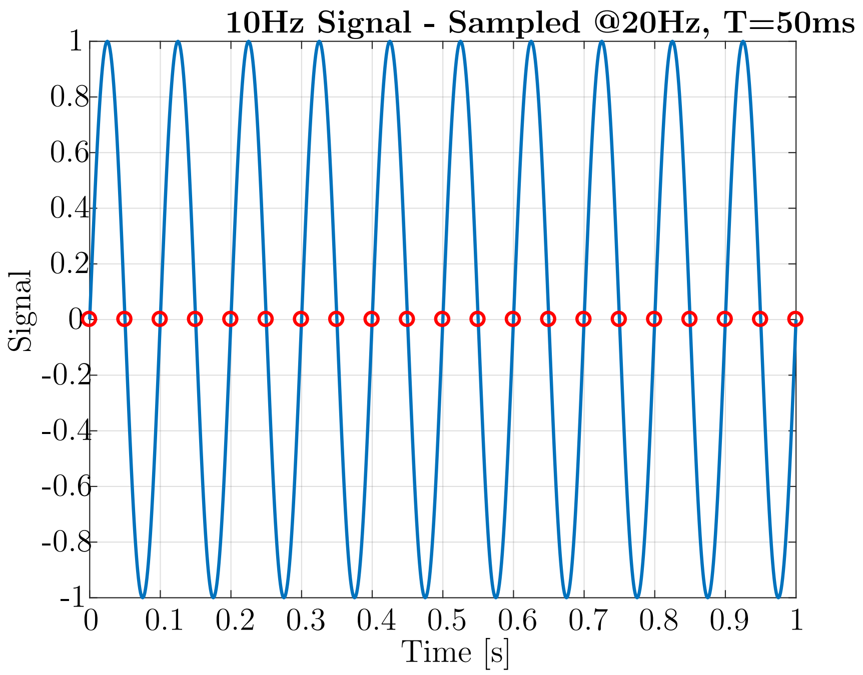
the digital recording of the value of a continuous signal (blue in Figure 4.1) at fixed time intervals (red). Choosing the correct time interval is critical to ensure that the sampled signal correctly represents the actual real-world signal. According to Shannon and Nyquist this sample rate must at a minimum be twice as fast as the fastest frequency we are trying to capture.

$$f_s \geq 2\, f_{max} \tag{4.1}$$

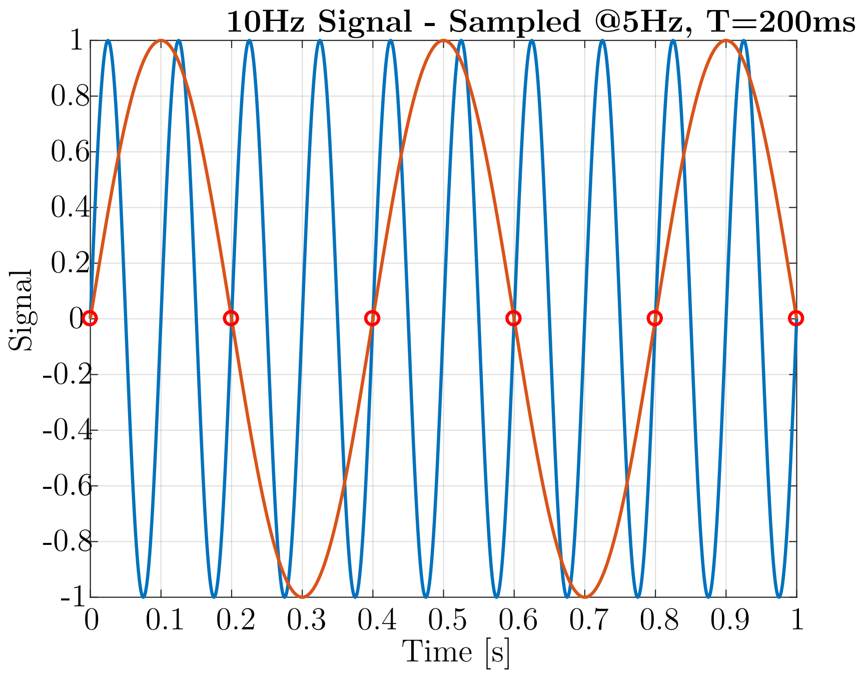
If we neglect to do this, our measured signal is subject to an issue called "aliasing" where a signal with a frequency different from the signal we are trying to measure occurs in the digital representation, shown in Figure 4.2. Figure 4.2a shows a simple sine wave of 10Hz being sampled correctly at 20Hz. If we take the Fourier Transform of the digitized data, we will obtain a correct result for the frequencies as well as be able to reconstruct the signal correctly. Figure 4.2b shows the same sine wave being sampled at 5Hz, below the Nyquist rate. The sine wave in brown is the aliased frequency being inadvertently measured because the sample frequency is too low. A Fourier Transform will yield the frequency of the brown signal rather than the blue and, hence, we are not be able to reconstruct the original signal correctly.

The above discussion from a Signal Theory standpoint is straight forward and most engineers probably encounter this at some point during their curriculum. However, practical implementation brings additional uncertainties.

First, consider that we should develop a control system for this signal. If were to only sample at the minimum required rate (the Nyquist rate) our control system will most likely perform poorly or potentially even be unstable. The reason for this is that our digitized signal does not "look like" the real signal or, in other words, we do not have enough resolution in time for the data to be useful for control. As Figure 4.2a shows, if we start sampling at just the right time, our digitized signal will be zero. In fact, in the case of the sine wave, no matter when we start sampling, we will always measure a signal with at most two different values. It becomes obvious that we must



(a) Sampling a Signal at the Nyquist Frequency



(b) Sampling Below the Nyquist Frequency

Figure 4.2: Illustration of Sampling at Different Intervals (Frequencies)

measure (much) more often than the minimum rate given by Nyquist if we are to use the signal for time domain processing. The much less known “Valvano Postulate” tells us that if we want the measured signal to “look like” the actual signal, we must sample at least ten times the maximum frequency we expect to see in the signal [67].

$$f_s \geq 10\, f_{max} \tag{4.2}$$

Indeed, in practice we usually stick to a value of 10 - 20 times the maximum frequency in the signal. Figure 4.3 shows the same sine wave sampled at twice the minimum frequency given by Valvano.

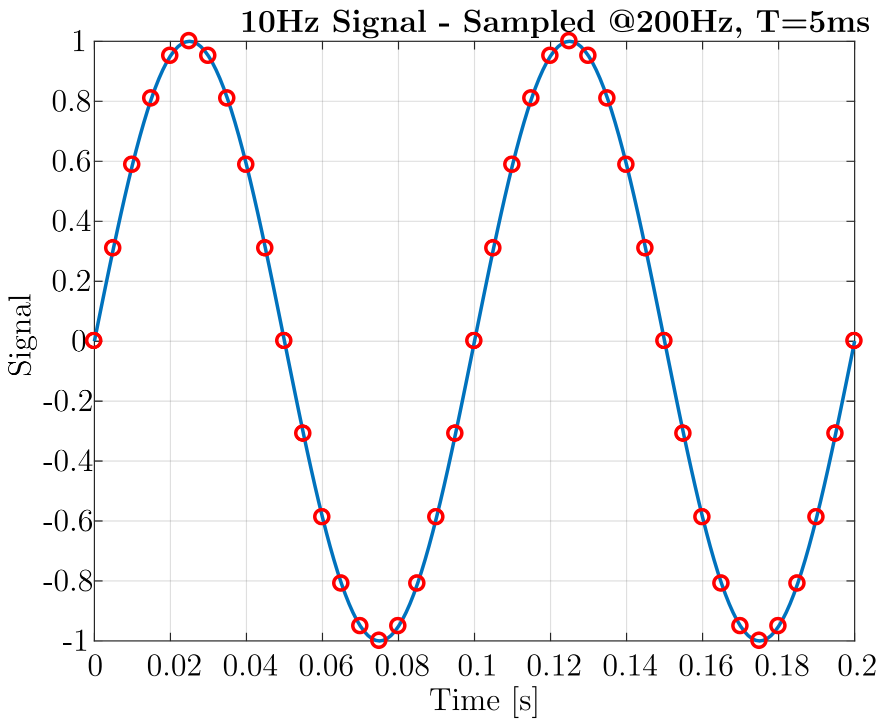


Figure 4.3: Sampling of a Sinusoidal Signal at $f_s = 20\, f_{max}$

Second, we must take precautions that aliased signals of higher frequencies do not again accidentally make it into our digitized signal despite having chosen the correct sample rate. This is usually done by including a low-pass filter before passing a signal to the ADC.

Stability

Stability, in the context of Control Theory, deals with the following question: Under which circumstances will a (linear) system exhibit a limited response to a limited input? In other words, under which circumstances will a system behave safely and provide a bounded output if given a bounded input. Because in practice linear systems are most often used we also limit ourselves to these systems here. In the context of Control Theory they are often called "Linear Time Invariant" (LTI) systems. LTI systems are mainly described by linear differential equations. Therefore we know the basic responses we can expect from these systems, some of them are shown in Figure 4.4. Again, many engineers will be familiar with the fact that stable signals

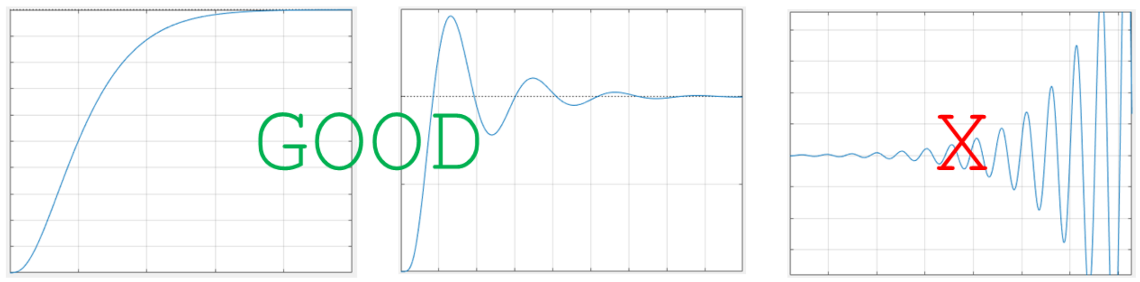


Figure 4.4: Common Responses of LTI Systems

generally look like $y(t) = e^{-t}$ and that in Control Theory language these systems have *all* poles located in the left half of the s-plane. Two such possible responses are shown in the left two graphs in Figure 4.4. In contrast, unstable systems look like $y(t) = e^t$ or $y(t) = t$ and have at least one pole located in the right half of the s-plane, shown in the right graph in Figure 4.4. We should add that, technically, a system with response $y(t) = t$ or $y(t) = sin(\omega t)$ is considered "marginally stable". However, due to the fact that even the slightest disturbance can cause such a system to become unstable we also try to avoid marginally stable systems at all cost and generally require all poles to be in the left half s-plane.

The foregoing discussion assumes design in the continuous domain as the usage of the s-plane indicates and control system design in the continuous domain is very common in practice. However, as we have seen earlier most systems end up running in the discrete domain; so how does this fit with our continuous domain design? It is generally accepted that, as long as sample rates are chosen correctly, i.e. much higher than the maximum frequency, the system designed in the continuous domain will correspond closely to the real-world system. Now, let us consider the transfer function below for a simple first order low pass filter with cut-off frequency $f_c = \frac{\omega_c}{2\pi}$.

$$G(s) = \frac{\omega_c}{s + \omega_c} \tag{4.3}$$

Note, that the structure of this filter is equivalent to the first order low-pass filter presented in Section 3.2.1 in the form of a resistor and capacitor. Clearly, this filter is a stable system in the continuous domain and based on our previous discussion we might fast assume that the following statement holds true about this filter.

> "As long as the cut-off frequency f_c is less than twice the sample frequency f_s the discrete implementation of the filter will be stable."

Note that we are only concerned about stability, not resolution. However, once we implement this filter using a sample rate of $f_s = 2f_c$ we will quickly find out that the opposite is true, see Figure 4.5. While the step response (top graph) is bounded, the response to a varying signal such as a sinusoidal signal is clearly unstable. We have made a wrong assumption and ignored that the stability test for a discrete system must be done in the discrete z-domain. Calculating the z-transfer function for the filter above using the zero-order-hold method yields the following discrete domain transfer function and single real pole z_1.

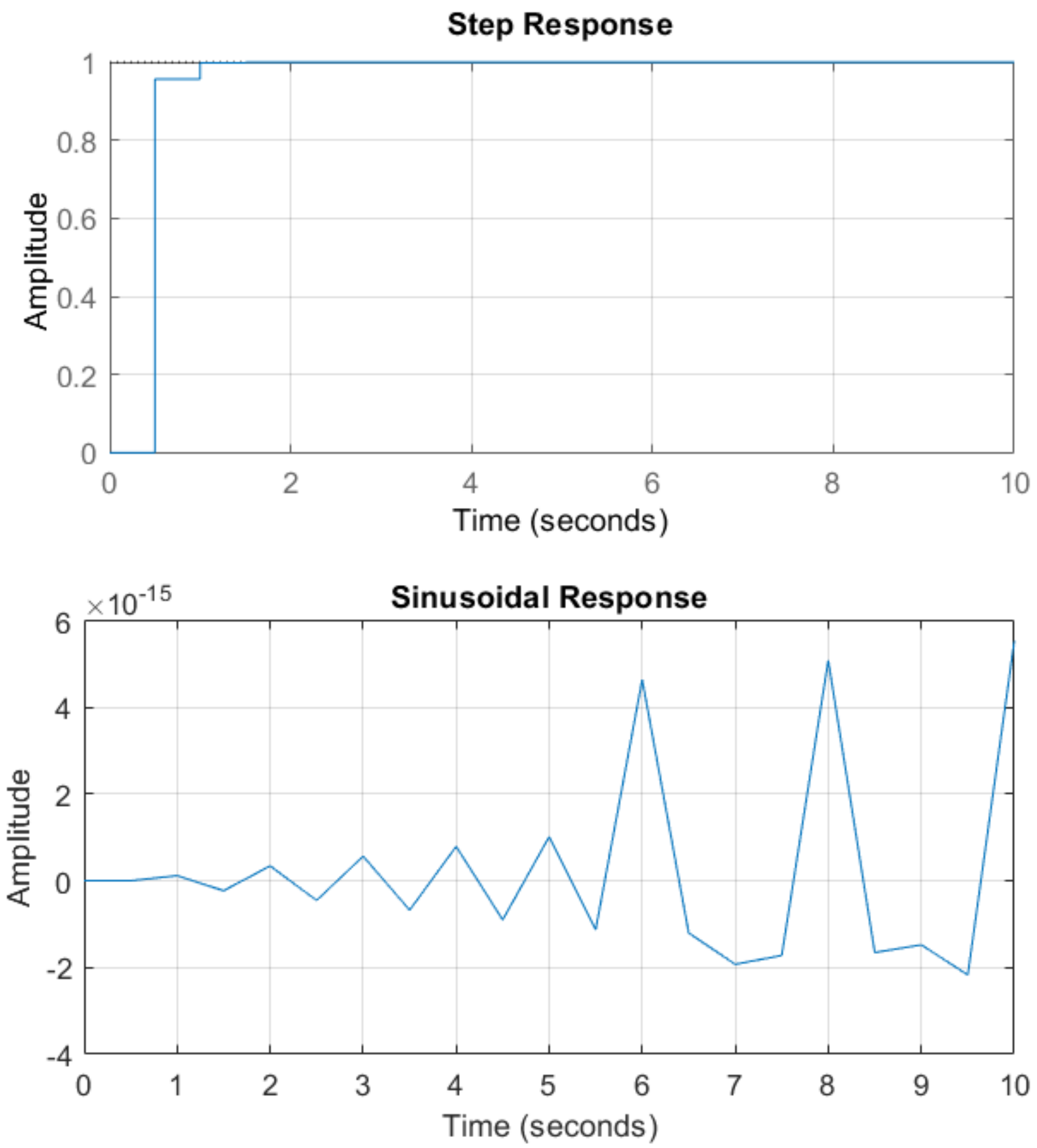


Figure 4.5: Step and Sinusoidal Response of a Simple First Order Discrete Filter

$$G_d(z) = \frac{\omega_c T_s z}{z - 1 + \omega_c T_s} \tag{4.4}$$

$$z_1 = 1 - \omega_c T_s \tag{4.5}$$

We must apply the z-domain stability criterion, which requires all poles to be located within the unit circle for stability, resulting in the following equation governing the stability of the discrete filter.

$$\omega_c < 2 f_s \iff f_c < \frac{1}{\pi} f_s \tag{4.6}$$

We see that it is actually the circular frequency ω_c, which is required to be less than two times the sample frequency. In terms of frequencies this means that the maximum cut-off frequency for our filter is less than a third of the sample frequency rather than half.

Remember that in a previous paragraph we mentioned that the discrete version of a continuous model will perform closely to the continuous model only if the sampling frequency is "much" higher than the maximum frequency and this example illustrates one such pitfall. The obvious question is, why would we even consider operating at a sample frequency that is coming close to these limits? Why not just choose a sample frequency that we are sure is high enough? The obvious answer to this question is we can only be *sure* if we do the math. Another answer is that we may be forced to push these limits due to real-time processing restraints. For instance, we may be trying to run a fast motor control loop and our processor simply cannot go any faster. Most importantly, wearable devices have a tendency to push us close to these limits as we have seen multiple times in this work.

4.2.2 Real-Time

Real-time is a concept of utmost importance and, unfortunately, in the dawn of the age of the Arduino robotics hobby community is often forgotten. However, real-time is absolutely essential to making wearable devices work in the way they are designed to and most importantly to make them safe.

In summary, in a real-time system the correctness of an operation is dependent not only on the correctness of its result, but also on meeting strict timing requirements. [48]

Let us revisit our discrete filter above to illustrate one example. Using the inverse z-transform of the transfer function $G_d(z)$ we can calculate the difference equation for the filter, which can then be implemented on a computer. Note that we use subscripts to indicate samples with u being the input signal and y being the output signal.

$$y_n = \omega_c T_s u_n - \left(\omega_c T_s - 1\right) y_{n-1} \tag{4.7}$$

We can see that the calculation of our filter directly depends on the sample interval T_s. If the actual sample interval is different from the one we used to calculate the filter our result will be wrong. Moreover, we remember from Equation 4.6 that the filter's stability also depends on the same sample interval. Therefore, we conclude the following.

> If signals are not sampled at precise times or
> if transfer functions are not calculated a precise intervals or
> if digital data is not sent/received at precise intervals
> then the theory and framework underlying all our assumptions including stability are not guaranteed.

Finally, as we will see again later, the failures will fall into a category, which is very difficult to troubleshoot. For example, a closed loop controller will work perfectly well and as designed but then unexpectedly becomes unstable without a discernible pattern. Eventually, one will find out that the periods of instability happen when a large transfer of data, unrelated to the control loop, to a peripheral is taking place. Therefore, as with all failures in this category it is best to avoid them by design. So how do we achieve real-time? We achieve real-time by using

1. oscillators, preferably crystals, to provide a precise time base,
2. timers to provide precise counts of time,
3. interrupts to ensure critical events get timely attention, and
4. real-time operating systems and schedulers to allow tasks to run at different rates.

4.2.3 Important Software Concepts

In this section we shall briefly talk about three important software pitfalls, which also fall into the category of issues that are very difficult to find once they have made their way into the design.

Preemption

Preemption is an occurrence in a multi-tasking system, where a task with higher priority is granted CPU access over (preempts) a task with lower priority currently running on the CPU. While this architecture allows for great reductions in processor load because a slow but high load task can run in the periods between a fast but low load task, it also creates a list of hazards. In fact, both race conditions and deadlocks can be caused by preemption, though preemption is not the only cause.

One particular problem caused by preemption is priority inversion. This happens in a multi-tasking system where there are also shared resources such as a buffer for a serial interface. Consider a multi-tasking system with a high priority task A and low priority task B, both of which need access to a buffer for a serial interface. If the low priority task B takes control of the buffer shortly before A is about to run, A has to wait for B to release control of the buffer, even though A has a higher priority. In other words, B has preempted A.

Race Conditions

Race conditions occur when the behavior of the software depends on the timing of processes and the fact that only few operations on a low level are atomic, i.e. they can be completed in one cycle of the CPU.

As an example consider a global variable X, which is accessed and written to by two processes A and B. Before the access the value of X is zero. First, process A accesses X and increments it by one. The value of X is now one. Second, process B accesses it and also increments it by one. Therefore, the expected end result for X is two.

Let us now consider the sequence of events a bit more realistically, listing out the steps involved in the operations. The value in parentheses shows the value of X at each step.

1. X(0)
2. Process A reads the value of X(0).
3. Process A increments X(0).
4. Process A writes (stores) the value of X(1).

5. Process B reads the value of X(1).

6. Process B increments X(1).

7. Process B writes (stores) the value of X(2).

8. X(2)

Finally, let us assume that these processes are running in a multi-tasking system where process A has priority 6 and process B has priority 4. Higher priority values indicate a higher priority for a given task.

1. X(0)

2. A regular scheduling event occurs and the scheduler assigns process B to run on the CPU.

3. Process B reads the value of X(0).

4. A regular scheduling event occurs and the scheduler determines that it is time for process A to run. Because process A has a higher priority, process B is paused and process A is given time on the CPU.

5. Process A reads the value of X(0).

6. Process A increments its copy of X(0).

7. Process A writes (stores) its copy of X(1).

8. Process A finishes, the scheduler assigns the waiting process B to run on the CPU.

9. Process B increments its copy of X(0)

10. Process B writes (stores) its copy of X(1)

11. Process B finishes, the scheduler assigns the next task to the CPU.

12. X(1)

As we can see, the variable X now does not contain the expected value of two, the final value of X is dependent on the *timing of events*. Had process B been assigned to run after process A stored his value for X, the problem would not have occurred. This issue produces undeterministic and undefined behavior, which is incredibly difficult to resolve once it has made its way into code. Therefore, as with all issues in this category, it is best avoided by design. One of the key mechanisms that enable the behavior we have seen are interrupts, however, because interrupts are necessary for real-time operation it is next to impossible to avoid them. For instance, one may choose a single tasking design where tasks cannot preempt each other. In fact, highly critical systems will often use this approach in order to eliminate one source of problems and make tasking more deterministic and predictable. Of course, this comes with the cost of high processor loads because now we are left with a worst case design again: All tasks have to be able to be completed within the smallest sample rate. Nevertheless, we are still left with regular peripheral interrupts. For example, process A may be a task and process B may be a UART reception interrupt. We must then make sure that processes are locked out of one another during the critical data access by means such as a semaphore or similar mechanism and make careful use of atomic instructions.

Deadlocks

A deadlock occurs when a process is waiting for a shared resource or other process but the resource or process never becomes available. Let us consider again the two processes A and B from earlier, which are trying to access a buffer (the shared resource) to send data over a serial interface. Process B also requires data provided by process A. The following situation is possible.

1. Process B starts to run on the CPU and requests access to the buffer.
2. The semaphore indicates that the buffer is currently free so process B locks the semaphore to start writing to the buffer.
3. Process A starts to run on the CPU and also requests access to the buffer.
4. The semaphore indicates that the buffer is currently in use so process A is put into a wait state.
5. Process B now requires the data from process A, which has not been provided yet.
6. Process B waits for process A to provide the data.

We see that we end up with a situation where process B waits for process A, because B requires data from A, and process A also waits for process B, because B is waiting for A to release the buffer. As a consequence, A and B will wait forever. The reason for this is that A and B do not know of each other's involvement except that B requires data from A. If A and B were people, the situation would be easy, we might say, "why can we not let A temporarily release control of the buffer such that B can continue?". This statement, however, infers that there must exist an entity, which has overarching knowledge of the situation. In practice, this "entity" is most often the operating system.

Summary

In summary, we should say that the issues encountered in this section are avoidable because there are solutions to each one of them. However, because they are best avoided by design, it is up to us to recognize an arising problem when we are developing the code rather than afterwards. We must always pay attention and review our architecture and code because, unlike the simple examples in this section, the real issue may not be always as obvious.

4.3 Controls for HeSA and APEx

In Section 3.3 the electronic system design for the HeSA and APEx projects were discussed. Because the high level design is similar for both projects we will discuss the high level control system design for both projects in this section.

MATLAB/Simulink and a separate toolbox from Microchip MPLAB Tools for Simulink was used to implement the control system design for these projects. As discussed earlier rapid code prototyping tools like Simulink greatly aid the development of controls for wearable devices. Simulink is a graphical programming language allowing to create block diagrams in an intuitive, controls focused way. One can quickly implement structures like filters, compensators, state machines and many more. An additional toolbox (Embedded Coder) then allows to generate C/C++ code from the block diagram for any type of processor implementation. However, usually one must develop the C-code for the basic micro-controller setup (e.g. oscillator setup) as well as its unique inputs, outputs and peripherals on their own, unless a toolbox specific to this processor exists. In this case, we used the toolbox developed by Microchip, which provides blocks for the configuration of many of their micro-controllers. Figure 4.6 shows a simple block diagram in the Simulink development environment and

Figure 4.7 shows some of the device blocks available from Microchip. One particular advantage of these tools is that they mostly handle the lower level details such as creating different tasks with different sample rates and handling the data transfer between these tasks. Further, discrete transfer functions can be designed and then tested both in simulation and in real-time by auto-generating C-code for them. If we encounter a specific task for which no support from Microchip exists we can integrate our own C-code using a C-function call block provided by Microchip, and in newer versions also by Simulink.

Naturally, generating C-code from a higher level programming language like Simulink will add some inefficiencies, for instance, in some cases the value of a variable may be moved from a to b to c, when it could have been moved to c directly. However, these inefficiencies are small and become less meaningful due to the increase in processor speeds and memory. Any small inefficiencies are also by far outweighed by the ability to quickly test multiple complex algorithm as well as the ability to perform model based design, hardware-in-the-loop execution and other advanced control system techniques. Finally, Simulink and others are industry standard and have been around for a long time, which is why big companies like Boeing are also using these tools for highly critical systems, of course using specially certified blocks.

4.3.1 Control System Architecture of HeSA

A high level view of the HeSA system controller design in Simulink is shown in Figure 4.8 and an overview of the functions and associated blocks can be found in Table 4.1. The sample rates given in the last column show the rates, which roughly correspond to the major functions within the blocks though some overlap exists.

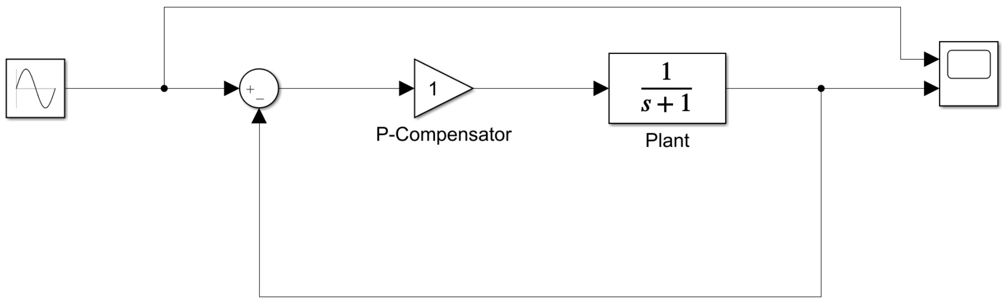


Figure 4.6: Block Diagram in the Simulink Development Environment, Showing a Simple Control Loop Consisting of, From Left to Right, a Sine Wave Reference, Summation Block, Proportional Gain, Plant Modelled in the Continuous Domain and a Scope for Display

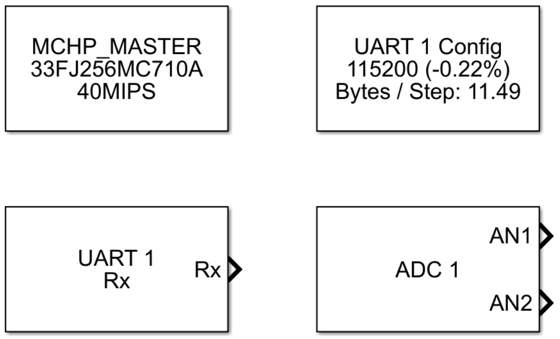


Figure 4.7: Example of the Blocks Available for Simulink from Microchip. Left Top: Basic Setup, e.g. Clock Frequency, Right Top: UART Configuration, Left Bottom: UART Reception Output

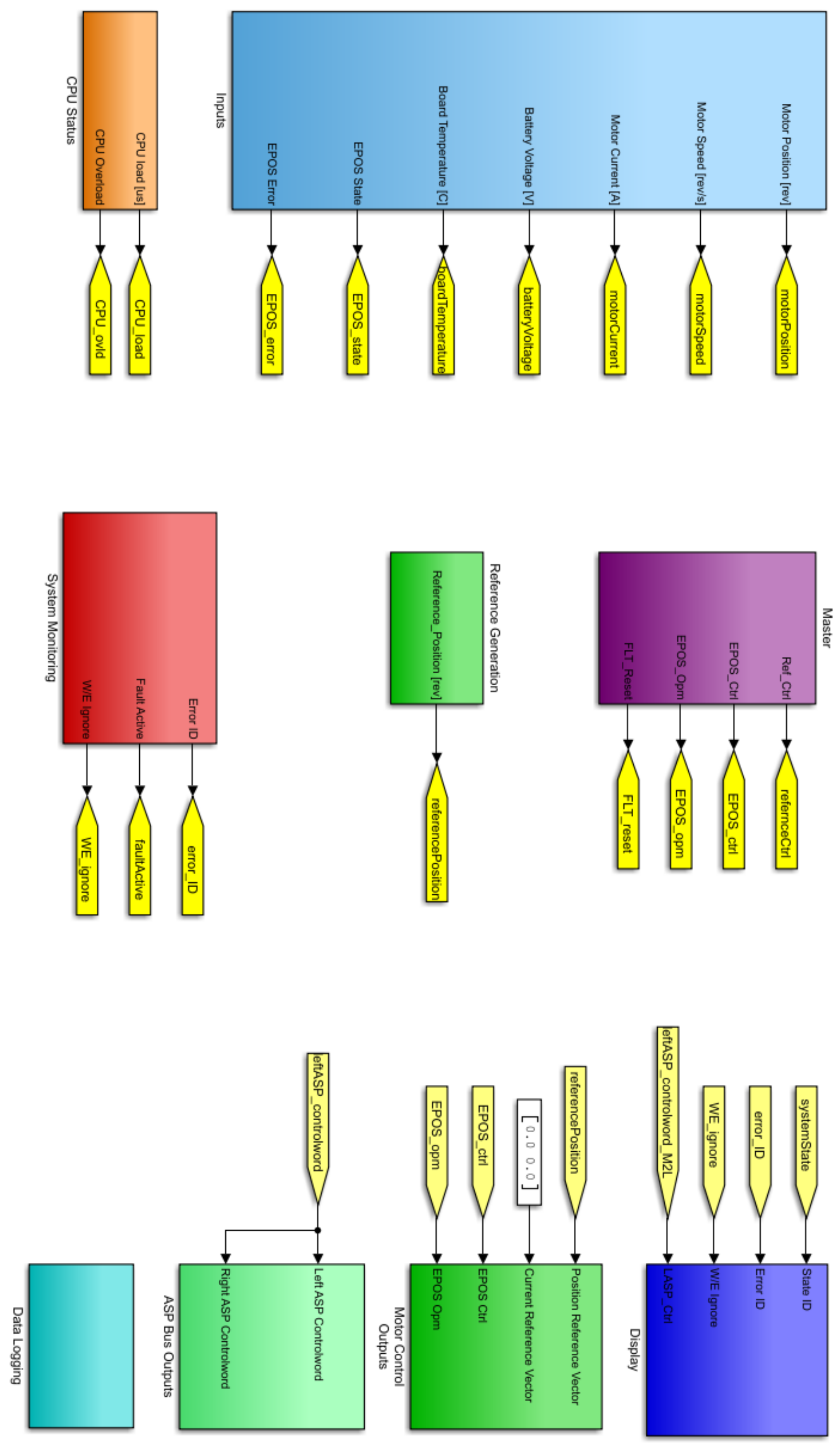


Figure 4.8: High Level Diagram of the HeSA System Controller

As can be seen from Table 4.1, the control system is split into the six basic functions listed below.

- Inputs & Outputs (I/O)
- System State Control (also part of FDIR)
- Controls
- System Monitoring (FDI/FDIR)
- User Interface
- Debugging, Design & Test

Inputs and outputs encompass the software, which interfaces the outside world with the control system and is part of the sensing and affecting functions of the wearable device. For example, within the "Inputs" block on-board sensors such as the

Block Name	**Function**	**Color**	**Rate**
Inputs	Inputs & Outputs	Light Blue	100-500Hz
CPU Status	System Monitoring (FDI)	Orange	100Hz
Master	System State Control	Purple	1000Hz
Reference Generation	Controls	Green	100-500Hz
System Monitoring	System Monitoring (FDIR)	Red	1000Hz
Display	User Interface	Dark Blue	10Hz
Motor Control Outputs	Inputs & Outputs	Green	500Hz
ASP Bus Outputs	Inputs & Outputs	Light Green	100Hz
Data Logging	Debugging, Design & Test	Cyan	100Hz

Table 4.1: Major Functions and Associated Blocks of the HeSA System

PCB supply voltage are read with an ADC and sensor data arriving from the Maxon EPOS2 motor controller and the left and right actuator sensor platforms are received. On the right hand side, the block "Motor Control Outputs" transmits the commands derived from controls to the motor controller. The block "ASP Bus Outputs" sends high level control commands to the actuator sensor platforms. Finally, the block "CPU Status" reads important status information from the processor, for instance, the processor load during the preceding time step and whether a task was not able to finish before its next execution (task overload). Most inputs are read at 100Hz with the exception of the CPU information, which is first read at 1000Hz (the fundamental rate) and then averaged and down-sampled to 100Hz.

System state control is concerned with control of the system on the highest level. The "Master" block is essentially a state machine, which decides overall system state based on information reported by the subsystems, for instance the status of the motor controller or the fault status of system monitoring. System State Control decides the main states of the system, i.e. Initialization, Homing Mode, Standby, Operation or Fault. Because System State Control also monitors faults and, depending on the fault, can decide to issue actions to recover from a fault, System State Control is also part of FDIR. Because System State Control must be able to act as fast as possible, for instance when attempting to recover from a fault, it runs at the fastest rate of 1000Hz.

Controls implements the control algorithms of the HeSA system. In the most simple terms, Controls generate a reference command for the actuator (the motors) based on sensor information. Controls also decides the type of command to send, for example motor current, velocity or position. The control algorithm runs at 100Hz and the output signal (motor position command) is then up-sampled to 500Hz and filtered for a higher resolution trajectory.

System monitoring (FDI/FDIR) encompasses all subsystems involved with observing critical system signals, detecting when these signals are outside normal operating conditions and isolating the subsystem responsible. Finally, a recovery response is determined, which can range from an attempt to reset the fault without the need to exit the operational state to disabling the system in a controlled way and latching this state until a hard reset occurs. In some cases, when a signal is outside the normal operating range, there may be a known remedy while in other cases the only known remedy is to safely disable operation. For example, if the motor controller reports a maximum current fault, this can be caused by an accidental, momentary current spike due to high acceleration. In this case, the System State Controller will attempt recovery by instructing the motor controller to reset the fault in order to recover from the fault safely and without interrupting the system operational state (recovery through fault reset). If the fault reoccurs more than a predetermined number of times within quick succession, then the fault gets latched permanently and the system is taken out of the operating state (recovery by acknowledging the fault and bringing the system into a safe state). Another example is the monitoring of the system supply voltage. Two voltage levels are monitored, corresponding to roughly 30% and 10% battery charge levels. If system monitoring detects that the supply voltage has fallen below the 30% level, the recovery attempt is to alert the user by changing the color of the user interface LED. If the voltage falls below the 10% level, the recovery response is to take the system out of the operational state in a controlled way. In contrast, if the battery was allowed to continue to discharge, the system would shut down abruptly once the voltage has fallen low enough to disable the components, or worse, the system might behave in an uncontrolled way when logic levels are undefined. System monitoring must be able to respond fast to any issues, therefore it runs at 1000Hz.

User Interface subsystems are those blocks, which control elements that allow the user to interact with the system. HeSA uses a simple interface consisting of two push-buttons and one three-color LED. The push-buttons allow the user to transition the system into different state, for instance from “Initialization” into “Homing” or from “Standby” into “Operational” and vice versa. The LED displays the current state of the device as well as any faults. For example, a blinking green light indicates the “Standby” state while a slowly blinking red light indicates that the battery must be recharged soon (voltage level below 30%). In regards to sample time, the user interface is the least critical system and runs at 10Hz to provide enough resolution for events in the order of 100ms or slower.

Debugging, Design and Test is the subsystem, which reports various data and information from within the device to a person running a matching interface on a PC or laptop. HeSA uses a relatively high bandwidth 1MBit/s serial-over-Bluetooth wireless interface to log 70 signals in single-precision floating point format at 100Hz for this purpose. Even though often unmentioned, this functionality is indispensable not only to troubleshoot problems but also to verify and analyze device performance. It is also possible to provide more detailed information about any faults over this device. In addition to reporting data from the device, it is also possible to change parameters on the device in real-time and wireless. This functionality allows to quickly try different actuation patterns for different users and quickly tune the device to each user.

4.3.2 Theory of Operation of HeSA

Upon power-on, the system initializes and then waits until the user pushes either button to go into the homing (or zeroing) state. During zeroing the motor controller drives the actuator at a constant velocity towards one end until a hard stop is detected by means of crossing a preset current threshold. Once the hard-stop has been detected, the absolute position of the actuator is known and the actuator is driven to the home position, which corresponds to zero degrees hip angle or a standing up position. Once the home position is reached, the system stops and enters “Standby” mode, where the motors are disabled.

The user now has two options to get into operational mode. First, using button one the user can enter a following mode, where the device simply follows the user’s motion without any support. Second, using button two, the user can enter support mode, where the device supplies torque to the user as they walk or run. To exit the operational mode, the user can use the same button they used to get into operational mode. If a fault occurs, the system reacts as outlined in “System Monitoring (FDI/FDIR)” above.

4.3.3 Zero Force Algorithm

A zero force algorithm was developed to enable the device to follow the user’s motion. This algorithm can be used to measure the user’s gait pattern as well as to get the user used to the device being active but without applying a torque to the user. In order to enable this functionality, the hip angle rotary sensor is used to derive a reference position for the motor.

The following control law was derived to implement the zero-force algorithm, with u denoting the output to the motor, x_r denoting the motor reference position,

x_a denoting the motor actual position measured by the motor incremental encoder and K_x being the P, I and D gains.

$$u(t) = (x_r(t) - x_a(t)) \cdot \left(K_p + K_i \int dt + K_d \frac{d}{dt} \right) \tag{4.8}$$

The reference command x_r is calculated as follows with $\theta(t)$ denoting the hip angle measured by the hip encoder and L denoting the lever arm of the mechanism.

$$x_r(t) = L\,\tan(\theta(t)) \tag{4.9}$$

While the control resulting from this algorithm allows the user to move the device with little resistance as opposed to when the system is unpowered and nearly locked, there exists some remaining resistance. This resistance is mainly a result of mechanical backlash but also the delays in the system, most notably the delay of θ, which is measured on the actuator sensor platforms and then sent via CANbus to the central node. To improve the performance a small amount of constant reference "lead" X_l was added to the reference, see below.

$$x_r(t) = L\,\tan(\theta(t)) + \mathrm{sgn}(\theta(t)) \cdot X_l \tag{4.10}$$

In practice, values of X_l from 0.5mm to 1.5mm produced an improvement over the basic form of this algorithm shown in Equation 4.9. Note that too high values for X_l can result in runaway conditions.

4.3.4 Reference Command Pre-Filter for HeSA

Part of control system design is the design of a reference command pre-filter or trajectory generator. The main purpose of the pre-filter is to condition the reference signal before it is sent to the control loop. This is done to prevent the closed loop system from reacting in a way that is not desired or not safe. For HeSA, a simple pre-filter consisting of a first order Butterworth filter with a passband frequency of 10Hz and a stopband frequency of 40Hz as well as a reference command limit (saturation) was chosen. The bode diagram for the filter is shown in Figure 4.9. The value for the reference command limit was set to slightly lower than the mechanical limit of the actuator. The secondary purpose of the pre-filter is to smooth the reference command

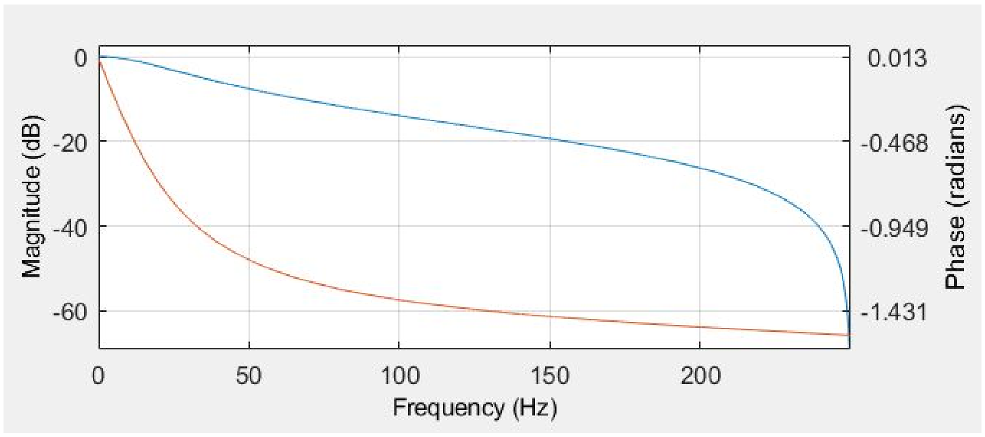


Figure 4.9: Pre-Filter for the HeSA Control System. The Amplitude Is Shown in Blue, the Phase in Brown

after it has been up-sampled from 100Hz to 500Hz. This part of the architecture allows the control algorithms, which are processor intensive, to run at a lower rate while still providing a higher resolution signal for smoother control at a higher rate.

4.3.5 Control System Architecture for APEx

Figure 4.10 shows the high level view of the APEx control system. The overall control system architecture for the APEx exoskeleton device is similar to HeSA in many aspects, hence we will only discuss the parts, which are different in more detail here. Table 4.2 shows again an overview of the functions and blocks different from the HeSA system.

As can been seen from Table 4.2 only some of the functions of APEx have been *architecturally* changed with respect to HeSA. These functions are outlined below.

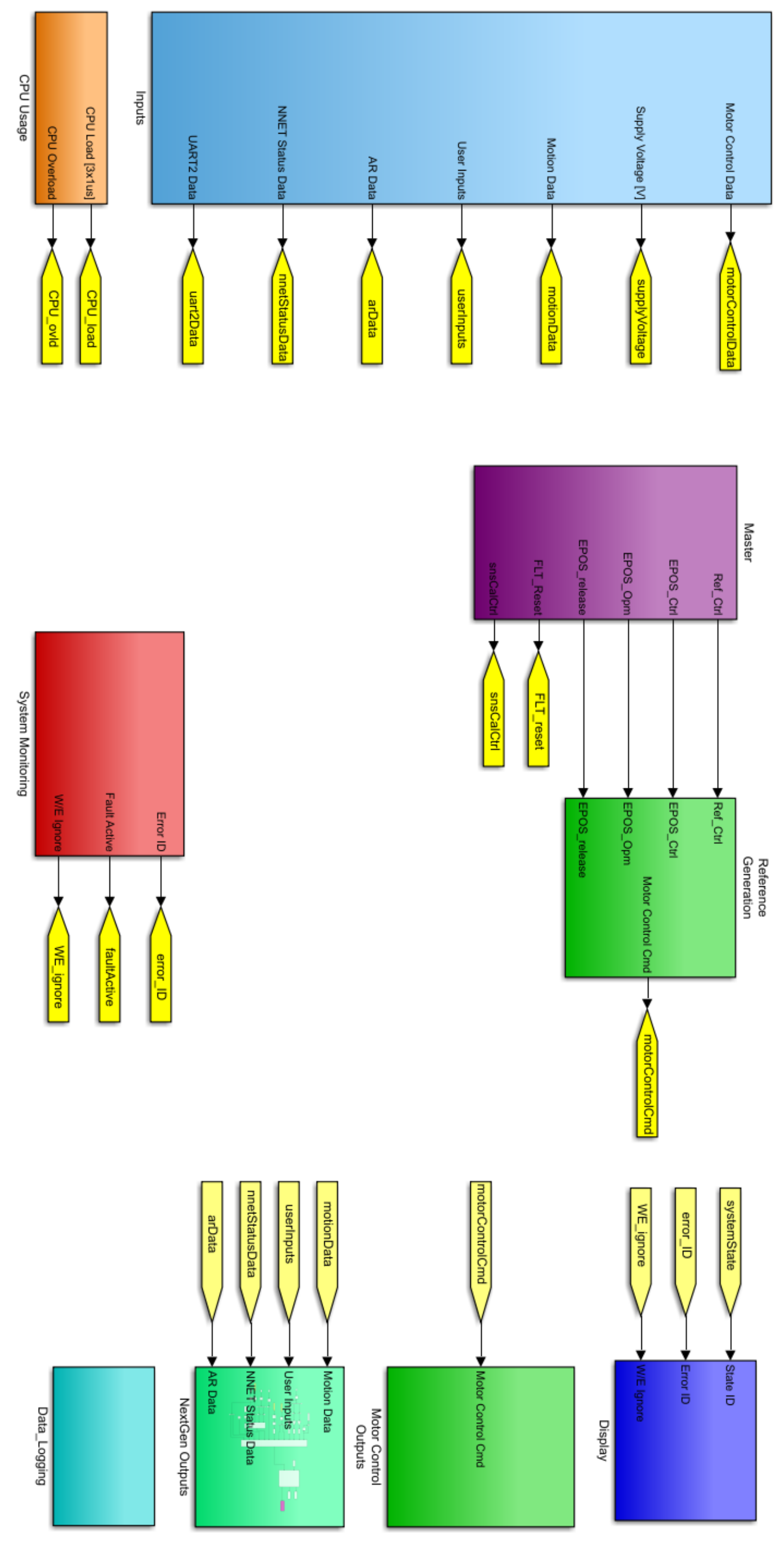


Figure 4.10: High Level Diagram of the APEx System Controller

Block Name	**Function**	**Color**	**Rate**
Inputs	Inputs & Outputs	Light Blue	200-500Hz
Reference Generation	Controls	Green	200-500Hz
NextGen Outputs	Inputs & Outputs	Light Green	200Hz

Table 4.2: Major Functions and Associated Blocks of the APEX System. Only Functions and Blocks Different From HeSA Are Shown

Note, that the code underlying each of the blocks has changed regardless because APEx is based on a newer hardware platform and functions differently. However, in this section we are only concerned with changes in system architecture.

Inputs and outputs were changed to be able to receive sensor data from both hip nodes now sent at 200Hz to the central node. This increase in speed was done to improve reaction times over the previous HeSA design.

Additionally, a new UART serial interface was added to enable communication with an external MPU board from NextGen. The MPU board runs a high level activity recognition algorithm based on sensor data from its own sensor network as well as sensor data transmitted from the central node. While the central node has its own activity recognition algorithm, the additional high level information from the MPU board is used as confirmation data. Transmission and reception between the central node and MPU board runs at a rate of 200Hz.

Controls of the APEx system are running at 200Hz, which aligns with the change in data rate of the sensor signals feeding the control algorithms.

4.3.6 Theory of Operation for APEx

Upon power-on the APEx system initializes and then after three seconds automatically enters two sequential startup procedures. First, the system enters the zeroing procedure. During zeroing, the actuator is driven at a constant velocity towards one end until a hard-stop is reached and detected by a preset current threshold. The actuator is then driven to the home position, which is located 1mm away from the detected hard limit. The home position corresponds to the actuator being disengaged from the rest of the mechanism, therefore allowing the user to move freely, without resistance and without power required from the actuator.

Second, after the home position has been reached, the system enters the automatic sensor calibration procedure. The calibration sequence is outlined further below and ensures that the accelerometer based sensor data has a common reference point, independent of slightly different mounting positions of the device on different people. During the calibration sequence the user simply stands normally. The automated calibration algorithm measures the offset of the sensor data, detects when it has obtained a reference point and advances to the Standby state. To this point, the device required no inputs from the user other than to turn on power to the device, all steps are performed automatically and usually within about 20 seconds or less.

In order to advance from the Standby state to the operational state the user turns a knob (rotary potentiometer) from left to right through a dead period of about one quarter turn. This dead period is fairly large and prevents the user from accidentally turning on the device. Once the dead period has been passed the device is in operational mode with the lowest level of support. As the user keeps turning the knob to the right, the support level increases in discrete steps from low to medium to high support.

In the operational state the device employs two modes of control depending on the internal activity recognition algorithm and the feedback from the MPU. During the support phase APEx uses current (torque) control to apply the level of torque selected by the user via the physical knob. Once a support phase is completed, the actuator is returned to a waiting position using position control. This ensures that the actuator stays within the bounds while allowing for a more natural feel of support using torque instead of position. By default, the internal activity recognition algorithm is used for decisions, implementing a safe, lower level layer. If the MPU board is present, the MPU board feedback is used to provide confirmation on a higher level of the decisions of the internal algorithms.

4.3.7 Automatic Sensor Calibration Algorithm

The internal activity recognition algorithm depends on sensing the tilt of the user's pelvis in order to determine the activity and timing of the support phase. The accelerometer sensors in both, the left and right hip nodes can be used to obtain a measurement of pelvis tilt. However, the orientation of the hip node boards depends to some degree of the person's size and how the exo-suit is mounted to them, which then leads to a different reference point (offset). For example, when a person is standing straight, the pelvis tilt measurement may be around two degrees for person A, while for person B it may be 12 degrees. Further, after donning and doffing the device, the pelvis tilt measurement may now be five degrees and eight degrees, respectively.

An automatic sensor calibration algorithm was developed to deal with these variations without requiring to create and maintain a user parameter table. The basic idea of the sensor calibration algorithm is to measure the mounting offset without picking up significant sway motion of the user or sensor noise as well as being able to operate without requiring an input, for instance "start", "stop", from the user. The algorithm, shown in Figure 4.12, consists of a very low frequency cut-off (0.05Hz) first order Butterworth filter, a signal variation acceptance logic with configurable sensitivity and a time threshold. The bode diagram of the filter can be seen in Figure 4.11.

The acceptance logic first calculates the "calibrated signal" while the calibration is in progress as a reference for the signal variation. Note, this is not the final calibrated signal because this value changes as the calibration progresses. The reference is then compared to a tolerance value. If the reference stays below the tolerance value for a predetermined amount of time (3 seconds) then the filtered value is held and subtracted from the raw sensor signal from now on. The result of this subtraction is the true calibrated signal. The tolerance is adjustable via a sensitivity value corresponding to a tolerance from one degrees to five degrees. Equation 4.11 below shows the threshold equation, which implements the signal acceptance logic with $u(t)$ being the unfiltered sensor signal, $u_f(t)$ being the filtered signal and S being a sensitivity value from one (least sensitive) through five (most sensitive).

$$|u(t) - u_f(t)| < |S - 6| \tag{4.11}$$

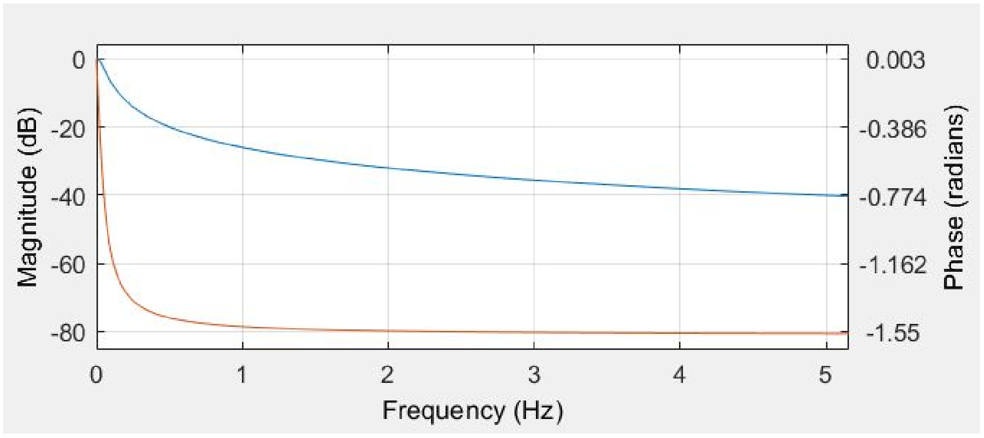
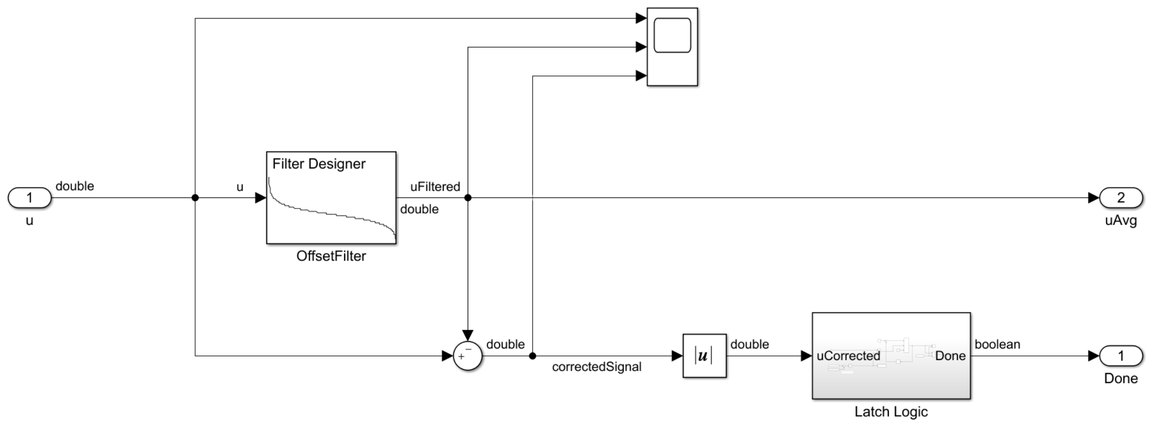
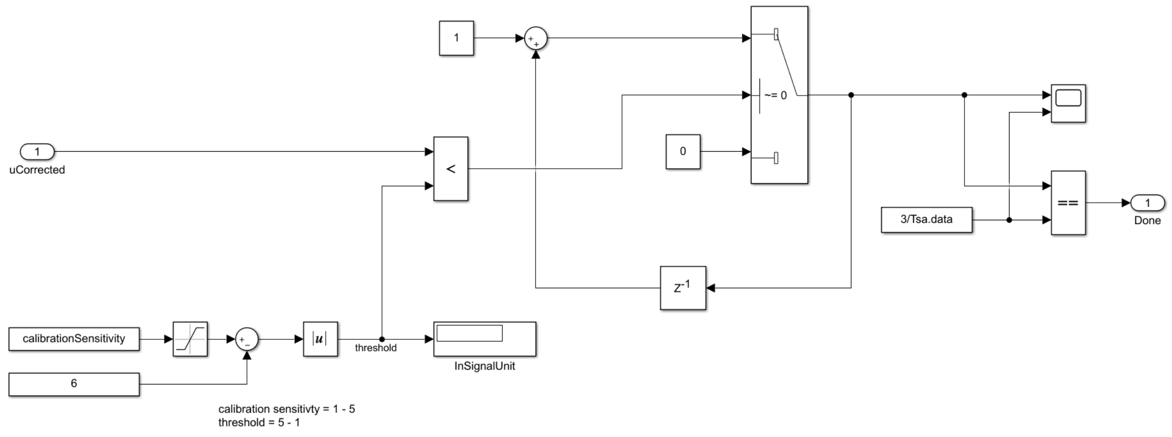


Figure 4.11: Sensor Calibration Algorithm Filter. The Amplitude Is Shown in Blue, the Phase in Brown.



(a) Main Level View of the Algorithm



(b) Signal Variation Acceptance Logic and Time Threshold of the Algorithm

Figure 4.12: Calibration Algorithm Implemented in Simulink

4.4 APEx Key Accomplishments

The APEx project was successfully demonstrated to the United States Air Force during a final demonstration in May at Travis Air Force Base in California. [21] Over the course of eight weeks of testing on over 12 individuals at the base [40], APEx was able to reliably and successfully show performance of the following tasks:

- Calibrate itself completely autonomously in a standing position
- Provide support for pushing
- Provide support for lifting
- Refrain from interrupting the user during tasks the exoskeleton should not support, i.e. walking, jogging or running
- Four hours of use with a battery of 36Wh, weighing only 396g

Chapter 5

REAL-TIME IMPLEMENTATION OF THE FAST FOURIER TRANSFORM FOR CONTROL OF WEARABLE DEVICES

5.1 Objective

A key step in the approaches discussed in Chapter 2 is to obtain the most basic characteristics about gait signals, i.e. frequency and amplitude. Once these are known, direct control strategies can be used to determine a command signal or algorithms such as machine learning can be used to extract further details about the gait. There are several ways to obtain amplitude and frequency information about a time varying signal. Many of them require the use of a time window to look back at the data. One such method is the peaks search method (PSM), which relies on computing the distance between two adjacent peaks in the signal. Note that this search is done in the time domain. A time window is defined and the peak search is performed within that time window. In this method, the selection of the time window along with the peak search algorithm are key parameters.

A disadvantage of the PSM method is that the peaks only update at the main gait frequency, which in general is fairly slow, in the order of 1Hz. Further, the time window must be kept larger than this frequency to ensure we do not miss a peak if the ambulation speed is reduced.

In this chapter we present the estimation of amplitude and frequency of gait signals using the Fast Fourier Transform (FFT) with the goal of reducing the time it takes to detect a change of these characteristics. Because the FFT looks at the entire data rather than just the peaks it may be able to detect changes in amplitude and frequency using shorter time windows. In order to be useful for the control of a robotic assistive device it is essential that the algorithm is able to run in real-time. Therefore, the FFT is tested on the dsPIC33FJ256MC710A processor in real-time to evaluate real-time compatibility of this approach. The development is performed using Matlab and Simulink as well as its capability to produce real-time executable code for the above processor with the Microchip Simulink Tools blockset.

5.2 Fourier Analysis and the FFT

5.2.1 Continuous Time Fourier Analysis

Fourier Analysis provides two main ways of obtaining amplitude and frequency of a given signal in the time domain, the Fourier Series and the Fourier Transform. The Fourier Series allows to calculate the coefficients (weights) of a series (sum) of sine and cosine terms, that are used to describe the continuous function $f(t)$ as follows

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n cos\left(\frac{2\pi nt}{P}\right) + b_n sin\left(\frac{2\pi nt}{P}\right)\right) \tag{5.1}$$

with P being the interval length or period. The approximation of the real continuous function $f(t)$ through the series $s(t)$ becomes more accurate the more terms a_n and b_n are used. However, the calculation of these terms is computationally intensive as two integrals must be solved and many sine and cosine terms must be calculated. The calculation of the a-terms is shown below, the b-terms are calculated analogously using sine instead of cosine.

$$a_n = \frac{2}{P} \int_P f(t) \cdot cos\left(2\pi t \frac{n}{P}\right) dt \tag{5.2}$$

The continuous Fourier Transform, shown in Equation 5.3 transforms a function from the time domain into the frequency domain. It is in concept similar to the Fourier Series but rather than resulting in discrete coefficients, it calculates a continuous frequency spectrum function $F(\omega)$.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi j\omega t} dt \tag{5.3}$$

As can be seen, this transformation in its continuous form also poses problems to be applied in the discrete domain due to the two sided infinite integral and is also computationally intensive. However, a discrete formulation, the Discrete Fourier Transform (DFT), exists, which is often used to perform Fourier Analysis in practical applications. Nevertheless, it is important to understand these basic mathematical principles in order to apply and understand the discrete version of the transform.

5.2.2 The Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT)

The DFT allows performing Fourier analysis on a finite amount of data. It can therefore be implemented on computers using numerical algorithms. The DFT transforms a time sequence of numbers x_n into the frequency domain representation F_k as follows

$$F_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{2\pi j}{N}kn} \tag{5.4}$$

One way to calculate the inverse transform is to use a Fourier Series, where the DFT samples F are used as coefficients at the corresponding DFT frequencies k. However, in its basic form, shown above even this discrete transformation is often too slow to be useful. Therefore, different algorithms, such as Cooley-Tukey, have been developed that allow computation of the DFT quickly. Often this is done by factorizing the DFT matrix, see below, into a product of sparse matrices and taking advantage of the fact that the terms $e^{-\frac{2\pi j}{N}}$ are primitive N-th roots of unity. These algorithms are often called Fast Fourier Transforms (FFT). [26; 68]

The DFT Matrix

In order to get a better understanding of how and why the DFT works we examine a 4-sample example, i.e. $N = 4$. A signal x with 4 samples shall be analyzed. We first write the DFT as follows and expand the terms.

$$\begin{aligned} F_k &= \sum_{n=0}^{3} x_n \, e^{-\frac{2\pi j}{4} kn} = \\ &= x_0 + x_1 e^{-\frac{2\pi j}{4} k} + x_2 e^{-\frac{4\pi j}{4} k} + x_3 e^{-\frac{6\pi j}{4} k} \end{aligned}$$

Per definition the DFT F_k of our signal x_n has as many "samples" as x_n and we can write the complete DFT as follows

$$F_0 = x_0 + x_1 + x_2 + x_3 \tag{5.5a}$$

$$F_1 = x_0 + e^{-\frac{2\pi j}{4}} x_1 + e^{-\frac{4\pi j}{4}} x_2 + e^{-\frac{6\pi j}{4}} x_3 \tag{5.5b}$$

$$F_2 = x_0 + e^{-\frac{4\pi j}{4}} x_1 + e^{-\frac{8\pi j}{4}} x_2 + e^{-\frac{12\pi j}{4}} x_3 \tag{5.5c}$$

$$F_3 = x_0 + e^{-\frac{6\pi j}{4}} x_1 + e^{-\frac{12\pi j}{4}} x_2 + e^{-\frac{18\pi j}{4}} x_3 \tag{5.5d}$$

We can now write these equations in matrix form with D_N being the DFT matrix

$$F = D_N \cdot x \tag{5.6}$$

whereas

$$D_N = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \quad ; \quad \omega = e^{-\frac{2\pi j}{4}} = -j \tag{5.7}$$

From this, we can now deduct the general form of the DFT matrix for a problem with N samples.

$$D_N = \begin{bmatrix} 1 & 1 & 1 & 1 & & \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ 1 & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix} \tag{5.8}$$

whereas $\omega = e^{-\frac{2\pi j}{N}}$. Solving the above matrix equation Equation 5.6 is now highly suitable for a numerical solver, given that the matrix calculation can be simplified, for instance by factorization or some other means. This is especially true for problems with a large amount of samples.

As an example, the Cooley-Tukey algorithms mentioned above works by breaking the DFT calculation down into many smaller DFT calculations. This is also often called decimation in time. For instance, as a first step the calculation of an N-point DFT is decomposed into calculating two $N/2$-point DFTs. During the next step it is further decomposed into four $N/4$-point DFTs and so on. These kinds of algorithms are also often called Radix algorithms. [15; 39; 49]

Finally, let us examine the DFT matrix in another way. To calculate a component in F, the components in one row of D_N get multiplied with the samples in x and then added together. The first row is all ones so one could look at this row as “measuring” the DC component of x. Keeping in mind Euler’s identity $e^{jx} = cos(x) + jsin(x)$, the second row contains four samples of a complex sinusoid with a base frequency of $\frac{2\pi}{4}$, the third row contains four samples of a complex sinusoid with a base frequency of $\frac{4\pi}{4}$ and so on. We can therefore look at each row and the resulting value in F as “measuring” how much a certain frequency is present in x.

5.3 Simulation in Matlab/Simulink

5.3.1 Post Processing Analysis

As a first step, the tibia angle measured with an IMU and sampled at 100Hz was analyzed in post-processing. Visual analysis using peaks yields a stride time of about 1.22s or 0.82Hz, see Figure 5.1. The data was then examined using the FFT with

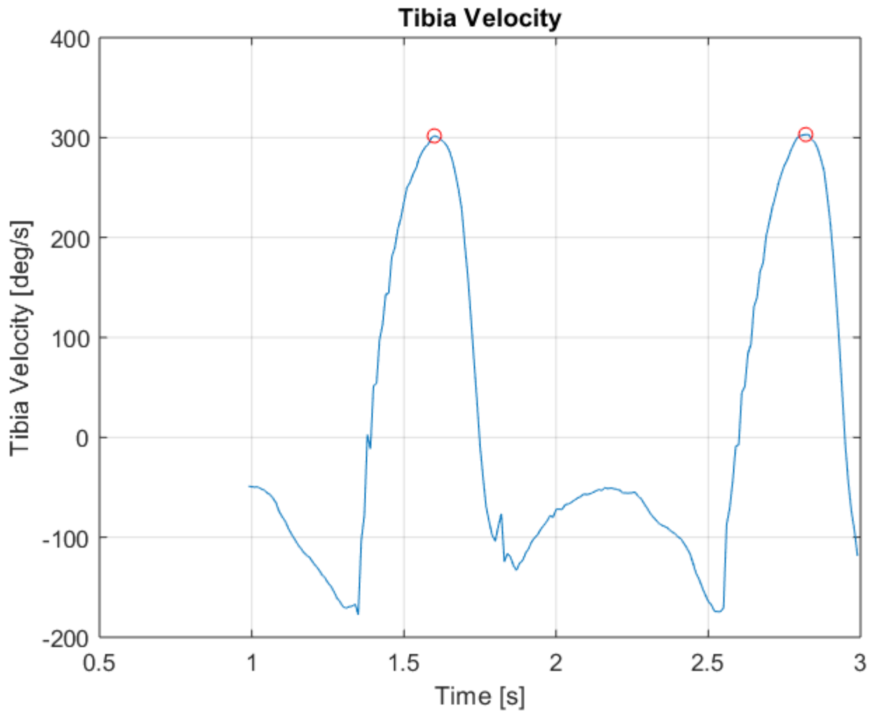


Figure 5.1: Walking Frequency of Data Sample, Measured With the PSM Method

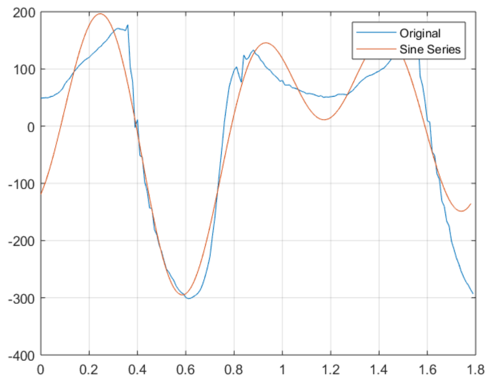
several different time window lengths from 1.1s to 4s. Table 5.1 shows the smallest frequency increment dF, the three most dominant frequencies f_1 f_2 and f_3 as well as the smallest deviation Δf in those frequencies compared to the "gait frequency" measured above with the peak search method. The calculation of Δf is shown in Equation 5.9 below, whereas $f_g = 0.82Hz$.

$$\Delta f = \min\left(\left|\frac{1}{f_g} \cdot [f_1 \quad f_2 \quad f_3] - 1\right|\right) \tag{5.9}$$

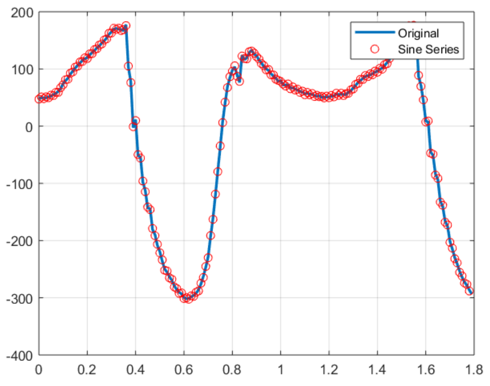
Table 5.1: Post Processing Analysis With Multiple Time Windows

W-Length	dF	f_1	f_2	f_3	Δf
0.99	1.00	1.00	2.00	4.00	21.95
1.09	0.91	0.91	1.82	2.73	10.86
1.19	0.83	0.83	1.67	2.50	1.63
1.29	0.77	1.54	0.77	2.31	6.19
1.39	0.71	1.43	0.71	2.14	12.89
1.49	0.67	0.67	1.33	2.00	18.70
1.59	0.63	0.63	1.25	1.88	23.78
1.69	0.59	1.76	0.59	1.18	28.26
1.79	0.56	1.67	1.11	0.56	32.25
1.89	0.53	1.58	1.05	0.53	28.37
1.99	0.50	1.50	1.00	2.50	21.95
2.49	0.40	0.80	1.60	2.40	2.44
2.99	0.33	1.67	1.00	0.67	18.70
3.49	0.29	0.86	1.71	2.57	4.53
3.99	0.25	1.75	0.75	1.00	8.54

There are two important points to note. First, note that the deviation in frequency does not mean that the FFT is not accurate. It is simply how the FFT works. Looking back at the previous section, we know that the FFT, operating in the discrete time domain, provides fractional frequencies based on the sample frequency and the number of samples in the examined signal x. To be precise, the frequency increment, as we have seen in the previous section, is calculated as $dF = F_s/N$ whereas F_s is the sample frequency and N is the number of samples. This leads to the second important finding. The longer the data set, i.e. the more samples, the smaller the increment between the discrete frequency points. This fact is also reflected in column dF, which lists the smallest frequency increment. It can be seen that the longer the time window, the finer the frequency increment becomes. If we were able to sample data over an infinite amount of time, the strength of any frequency up to half of the sample frequency could be obtained and we would have a semi-continuous Fourier Transform. If we were also able to make the sample frequency infinitely large, we could obtain the full continuous Fourier Transform. But because we only have a finite number of discrete data points we are left with discrete, fractional frequencies. Nevertheless, within the boundary of the Nyquist-Shannon Sampling Theorem, the result of this FFT is accurate as shown in Figure 5.2. Figure 5.2a shows the original



(a) Original and Reconstructed Signal Using 3 Terms in Sine Series



(b) Original and Reconstructed Signal Using All Terms in Sine Series

Figure 5.2: Validation of FFT Analysis

and reconstructed signal using a sine series with the three most dominant frequencies, i.e. three cosine and three sine terms. It can be seen that only using the three most dominant frequencies allows reconstruction of the original signal with a good degree of accuracy. Figure 5.2b shows the original and reconstructed signal using a sine series with all available frequencies. The two signals are now almost indistinguishable of each other.

This result does, however, raise an important question. Even though the frequencies are accurate, we are not easily able to obtain a “main” ambulation frequency. We can see this problem in the column that shows the frequency error of each time window. Whenever the actual ambulation frequency is not an integer multiple of dF, even the smallest error taken from the three most dominant frequencies can become significant. The question becomes how this impacts the application of machine learning algorithms that would most likely be used in a step subsequent to obtaining the basic amplitude and frequency characteristics of a gait signal.

A first potential issue is the update of the time window. The time window must be updated such that it can grow or contract with different gait speeds. One approach to define a time window is to use the obtained gait period and add a safety margin to it such that we can ensure that we can capture at least one full cycle if the speed is reduced. However, if the main gait frequency is not accurately known how can we achieve this? To approach this issue let us remember that the FFT works distinctly different from other approaches such as the peak search method. For instance, the PSM does not provide any updated information *until* a new peak is found. In other words, the frequency and amplitude output will remain at their previous value until a new peak is detected. On the other hand, the FFT output will keep changing every time step as the oldest value in the time window is discarded and a new one is added. Therefore, it is conceivable that the FFT is more robust to the selection of the time window. Even when the time window is selected smaller than the main gait frequency, the FFT still provides results that allow complete reconstruction of the signal in the window and, therefore, these results contain valid information about the underlying signal. They may not allow to directly obtain the main gait frequency but provide a close enough estimate to update the time window length accordingly.

A second potential issue is whether the results of the FFT will provide useful information for machine learning algorithms. For instance, will the main frequencies and amplitudes of different activities cluster nicely in a cluster analysis? The FFT may provide a possible approach to this as well. For instance, one could choose to use more than one dominant frequency and expand the machine learning algorithm. This would have the added benefit of increasing the amount of information that the machine learning algorithm uses to classify the activity. However, these possible approaches and their feasibility will only become clear once a more detailed machine learning analysis is performed using data obtained via the FFT.

5.3.2 Real-Time Simulation in Simulink

In order to prepare for real-time execution of the FFT algorithm on the embedded dsPIC processor, the FFT algorithm was first run in a real-time simulation in Simulink. "Real-time simulation" means that the Simulink model was configured to use a fixed-step discrete step solver. This means that the model will be run in fixed, discrete time steps just as it would be on the real hardware. The time window was preset to a constant value and not changed on the fly. The FFT calculation was run at the same sampling frequency as the raw data, i.e. 100Hz. Figure 5.3 shows the simulation data. The top graph is the gyro input signal. The middle and bottom graphs show the three most dominant frequencies and associated amplitudes, respectively, calculated during each time step. While the amplitudes change constantly, the frequencies change at a much slower rate. The middle graph also clearly shows the discretization of the calculated frequencies. It also reveals that using the results to update the time window length on-the-fly proves more difficult because the most

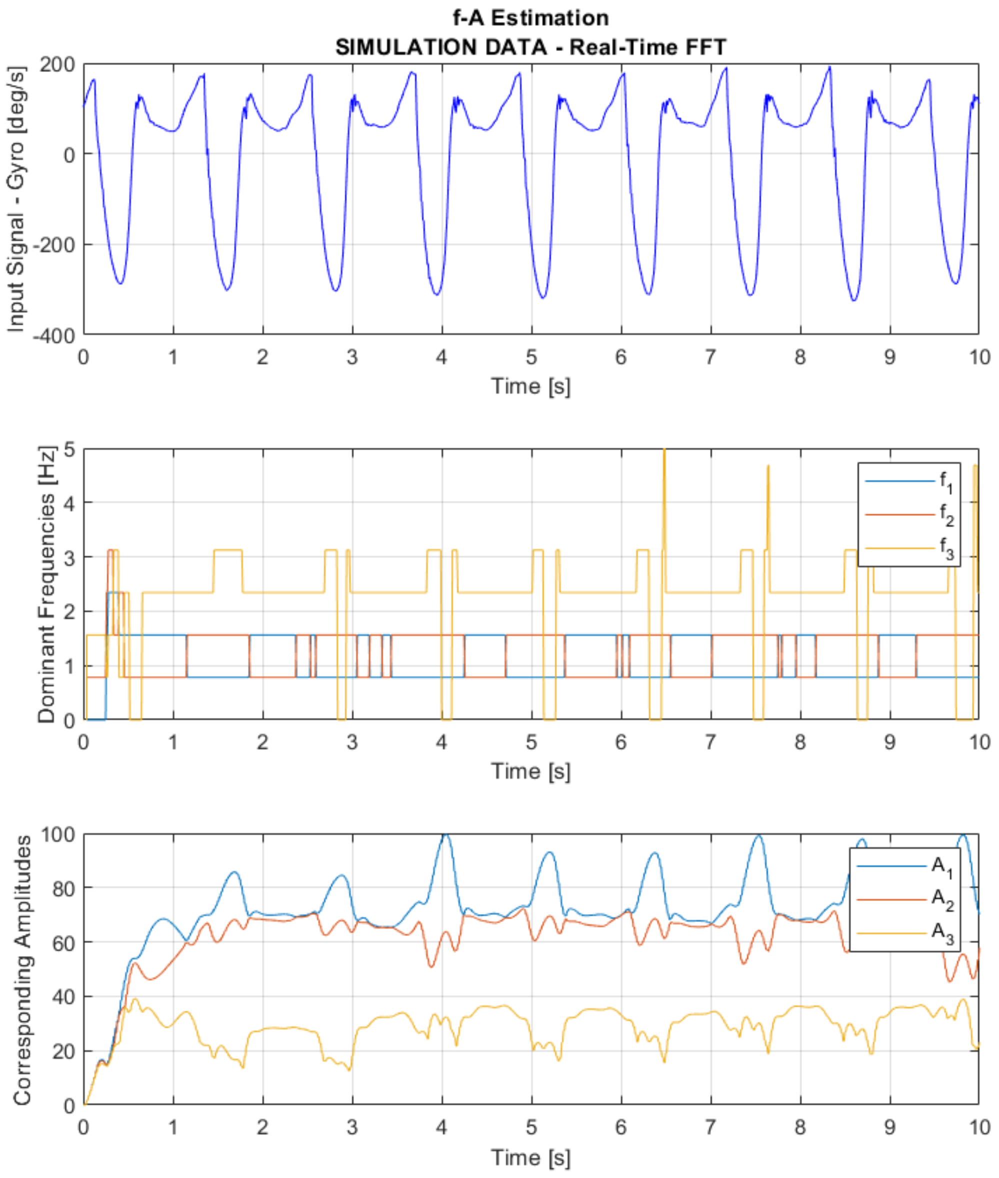


Figure 5.3: Simulation Results Showing Input Signal and FFT Outputs

dominant frequencies change over time.

Figure 5.4 compares the input data to the reconstructed signal using the results of the simulation. The two signals overall match well and validate the results. Note

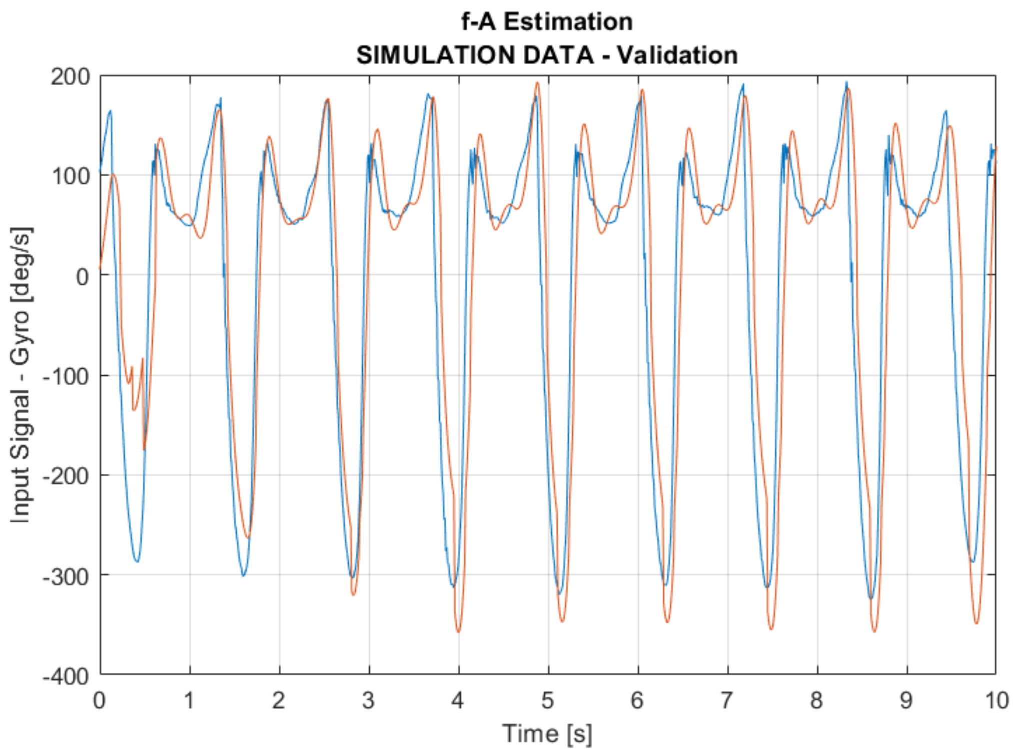


Figure 5.4: Validation of Simulation Results Through Reconstruction of the Signal

that the discrepancy in the beginning is due to the calculations being performed in real-time as opposed to being post-processed. When the simulation first starts, the time window is empty and must be populated with enough data in order for the FFT to yield meaningful results. This means that during the first execution of the FFT algorithm only one sample is available, during the second two samples and so on. Considering this fact, the fit is actually quite good as it still captures the overall shape and does not yield unstable behavior.

5.4 Real-Time Execution On the dsPIC33 Processor

As a last step, the FFT algorithm was implemented on a dsPIC33 processor. The same gyro data used in the previous section is now stored in memory on the dsPIC and then accessed time step by time step as if it was data coming from a real sensor at 100Hz.

For real-time execution, the sample frequency of the FFT was set to 50Hz. This means that frequencies and amplitudes are updated at half the rate of the incoming data. This is due to the processor CPU limitations. Execution at 100Hz causes some tasks on the CPU to overload, which means that real-time execution for those tasks cannot be guaranteed. At 50Hz no task overloads occurred and real-time execution is working properly. The average processor load at this rate was about 95%.

Data was collected by streaming the signals using a UART interface over Bluetooth to Windows and Matlab. The sample frequency of this data is also 100Hz. Figure 5.5 shows the results of the code running in real-time execution. The results are similar to the simulation results. Upon close inspection, the amplitudes appear slightly jagged as compared to the simulation, which reflects the slower sample rate (50Hz vs. 100Hz) used for FFT calculation. Note that the algorithm behaves in the same way as in the simulation, even though the presented graphs do not show the same initial start-up behavior as seen in the simulation. This is because it only happens in the beginning and due to the time it takes the Bluetooth connection to be established it is currently impossible to capture this behavior during real-time execution.

Figure 5.6 shows the validation of the calculated frequencies and amplitudes through reconstruction of the signal in post processing. This was done to not further increase processor load. We can see that the reconstructed signal again matches the actual signal quite well, similar to the simulation, validating correct execution

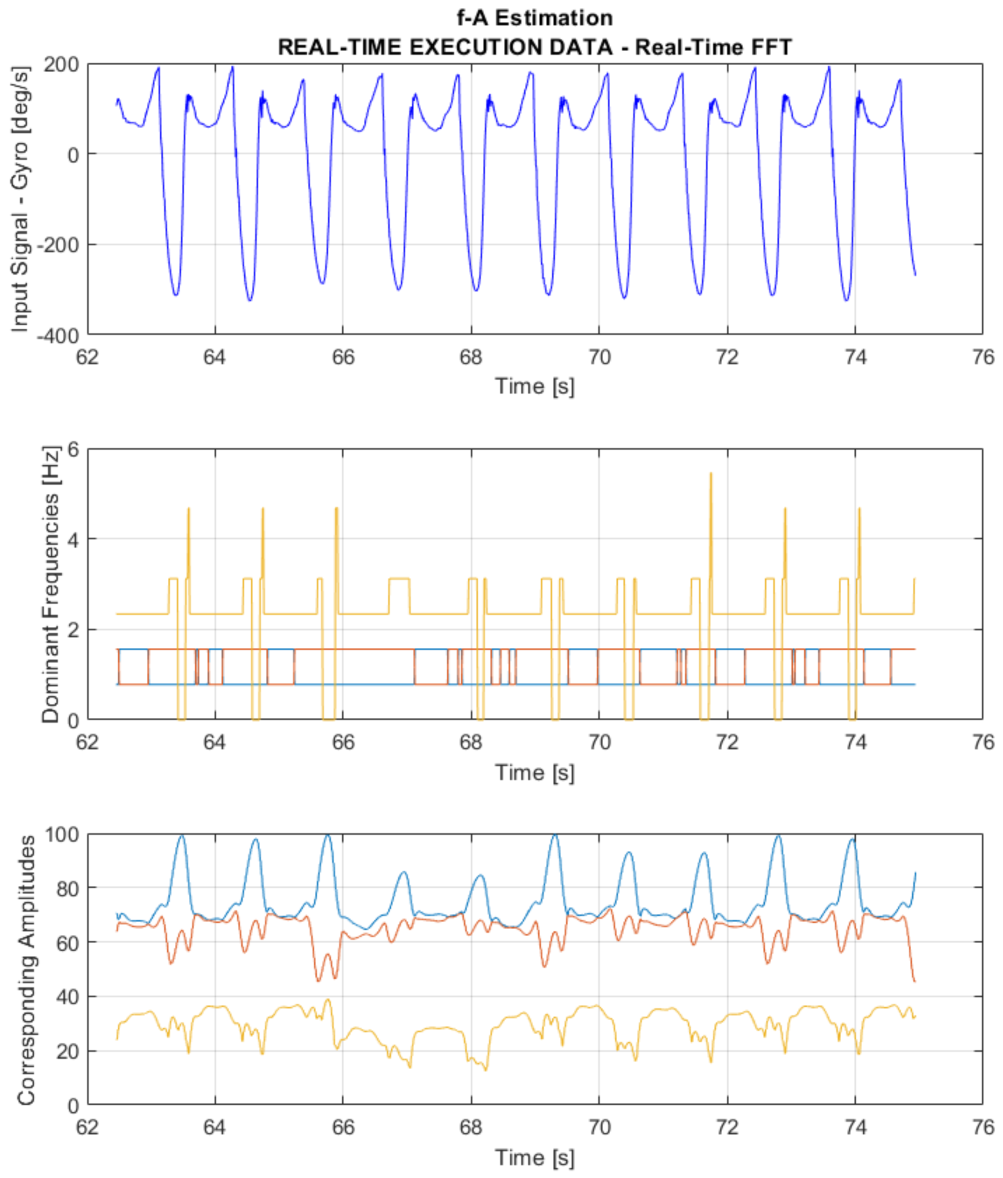


Figure 5.5: Real-Time Execution Results Showing Input Signal and FFT Outputs

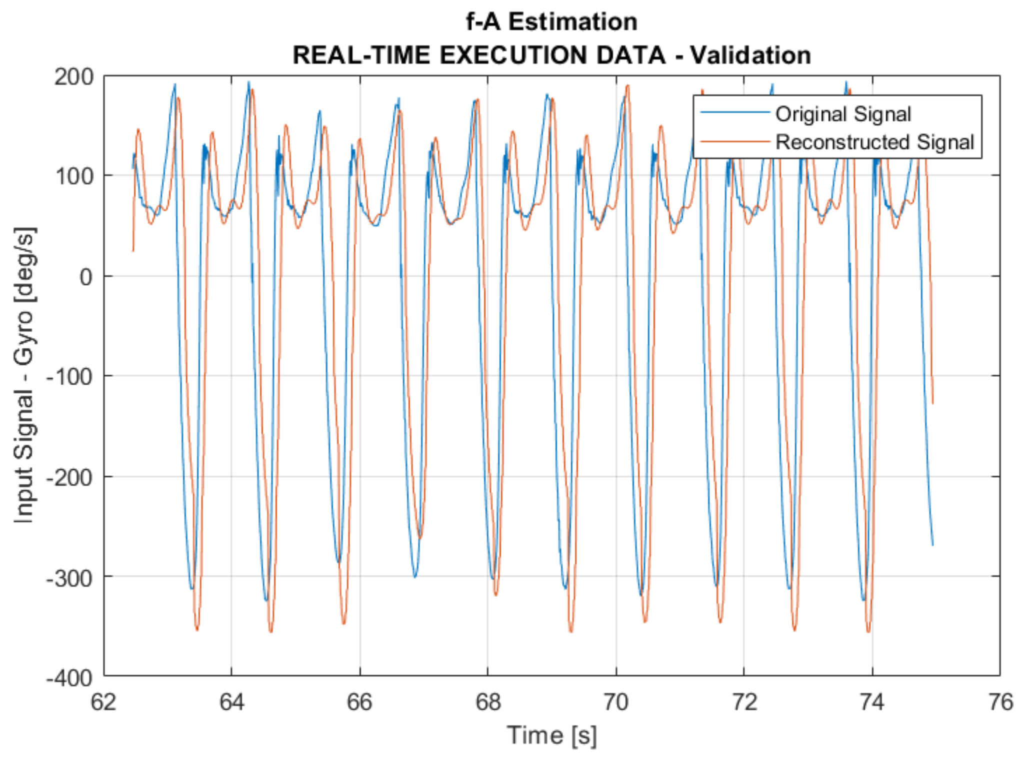


Figure 5.6: Validation of Real-Time Execution Results Through Reconstruction of the Signal

and calculation of the FFT. Note that just as in the simulation only the three most dominant frequencies were used to generate the reconstructed signal.

5.5 Summary and Conclusion

5.5.1 Summary

In this chapter frequency and amplitude estimation of time varying signals using the Fast Fourier Transform was presented. While there are many possible areas of application, our interest in this topic is motivated by its application to the control of exoskeletons or other devices that require extracting basic signal information from gait signals.

The chapter covered basic Fourier Analysis theory as well as its application in post processing, real-time simulation and real-time execution on hardware using a Microchip dsPIC33 16-bit microprocessor. Tibia angle measured with an IMU during walking was used as the signal from which frequency and amplitude information was extracted.

5.5.2 Discussion of Results

Post processing analysis using the FFT algorithm from Matlab revealed that the FFT works distinctly different from other methods such as PSM. The calculated frequencies are discretized and depend on the number of samples as well as the sampling frequency. Therefore, the FFT does not always yield an accurate value for what we consider the "basic" gait frequency. The basic gait frequency is usually specified as the inverse of the time it takes for a gait cycle to complete. For instance, when walking the stride time is the time from the heel strike of one foot until the next heel strike of the *same* foot. This finding has been briefly discussed in Section 5.3.1. However, this result does not mean that the FFT is unfit for this application but requires further analysis in order to determine its impact on machine learning algorithms that would use the FFT output data. It is conceivable that machine learning algorithms can actually be improved because we have more information about the signal (three frequencies and amplitudes) than we do with PSM (one frequency and amplitude). However, the important question is whether the data shows a significantly different pattern for different walking speeds and activities.

Determination of the time window length should be analyzed further as well. This is necessary because the FFT does not yield an exact value for the "basic" gait frequency and the most dominant frequencies change over time as seen in Figure 5.3 and Figure 5.5. Because the FFT does not care about peaks, it is possible that this approach is much more flexible to the length of the time window and that it can even be kept constant. If this is not the case, however, the issue of time window calculation must be addressed. One way to obtain a more robust estimate of the basic frequency is to weigh the two most dominant frequencies as well as the “zero-frequency”. This approach follows from Figure 5.3 and Figure 5.5. The two most dominant frequencies are alternating and always present, whereas the third most dominant frequency is less predictable. This approach could be mathematically formulated as

$$f_b = \frac{A_1 f_1 + A_2 f_2}{A_0 + A_1 + A_2} \tag{5.10}$$

whereas f_b is the basic frequency estimate. Digital signal processing also offers other approaches such as the application of filters to better condition the gyro signal either directly at the IMU or thereafter.

Second, the FFT method was implemented in real-time simulation as well as in real-time execution using the same data set. The simulation was performed in Simulink and hardware execution was performed on a dsPIC33 16-bit microprocessor. Most importantly, the results show that real-time execution of the FFT on embedded hardware is possible, the results were validated and qualitatively the results between simulation and execution agree very well. Further, the results are promising because the FFT is able to update its outputs much faster than the PSM method. As each new data sample comes in, it is considered in the frequency and amplitude calculation. This is in contrast to the PSM method, which at best can update the frequency only at the basic gait frequency.

Third, let us briefly discuss the area of processing. The fact that a comparably small 16-bit processor with only 40Mhz was able to calculate the FFT of a time window with over 100 samples at 50Hz is noteworthy. While in the simulation the FFT calculation was implemented at 100Hz, the sample rate had to be reduced to avoid the task from overloading. At 50Hz no task overloads occurred and real-time execution is intact. The CPU load was still high at this rate at about 95%. However, this is not a major concern because there are many approaches to address CPU load. First, we could investigate the effect of data size and optimize the length of the time window. The larger the time window becomes, the more complex is the calculation of the FFT. Second, one could look at more powerful processors. Even though it is very powerful, the dsPIC33 has been around for a long time and there are newer and many times more powerful processors available that would drop execution time instantly. Considering that the dsPIC33 is a 16-bit, 40MHz processor and we are able to perform complex floating point calculations like this shows that it is actually a quite powerful processor. Third, one should consider the question of whether it is even necessary to run the FFT at this relatively high frequency. Most likely, the results would be used for high level gait detection such as distinguishing between activities or gait speeds. From this perspective, it should be possible to drop the execution rate further about 10-20Hz. Moreover, it may make sense, both from a CPU load and architecture view, to use a two-processor approach and dedicate one processor to high level gait detection and the second one to mid and lower level control. Finally, we could look at the exact method Matlab uses to implement the FFT and investigate possible optimizations. As was mentioned in Section 5.2.1, the efficiency of the FFT depends mainly on the algorithm used and some algorithms may be more suitable for this particular processor than others. This approach is mentioned last because from our experience the auto-generated code is usually quite effective and the other approaches are more straight-forward to implement.

5.5.3 Conclusion

The results are promising for several reasons. First, all three evaluations show that the FFT is a feasible approach to obtain f-A information about a signal even in real-time execution on an embedded processor with limited resources. Second, the results have shown that the FFT is able to consider new signal information faster because it does not have to wait for a new peak to be detected. Third, the FFT is able to provide more information about the underlying signal because it provides more than just one amplitude and frequency.

The findings also show that further research is needed to examine the feasibility of this approach to be applied to activity recognition. First, different speeds and activities must be analyzed with this method. Second, the outputs must be examined with a machine learning approach to understand if the outputs are suitable for machine learning algorithms such as the Support Vector Machine (SVM) or k-Nearest-Neighbors (k-NN). Finally, the choice of the time window length should be investigated more closely.

Chapter 6

A NOVEL ACTIVITY RECOGNITION ALGORITHM USING A NEURAL ODE

The field of Neural Networks has gained a lot of ground and attention over the last decade. This development was aided by two main advances in technology. First, computers have become powerful enough in order to run these kinds of models that rely on a large amount of numbers (weights, biases, etc.) being tuned (trained) to fit a large set of data. Second, and more importantly, these large data sets have become more readily available through technologies like our smartphones. Most if not all of today's smartphones contain a 9-axis IMU. This IMU is used, for instance, to switch the screen of the smartphone from portrait (upright) to landscape (laying on its side) for our convenience by measuring the gravity vector with the accelerometer. However, the same sensor can be used to measure a person's hip angle while sitting in their pocket during a variety of activities. For example, Ronao et. al [56] and Mondal et. al [44] have used the data from smartphone sensors to classify different human activities using neural nets. Ronao uses a multi-layer convolutional network to extract six different activities from a 1D time vector of 128 readings over 2.56 seconds of 3D accelerometer and 3D gyroscope data. Mondal also uses 128 readings and a graph neural network (GNN) to predict up to 13 classes of activities.

Some limitations of these existing neural networks for activity recognition are that they require very large data sets and a fairly large amount of samples to look back on as well as a significant amount of computing resources. For instance, Mondal is using six different publicly available data sets each with a very large amount of data and an octa-core processor with 8GB GPU and 32GB system memory, which they call a "low-resource" system. One of Mondal's six data sets, the MobiAct data set [6], contains scenarios of daily living for 66 subjects and 3200 trials.

Second, if an algorithm requires 2.56 seconds of sensor inputs it will be very slow to detect activity transitions. Earlier presented algorithms like the ones requiring basic signal information such as amplitude and frequency also have similar limitations based on the algorithm used. For example, the peak search method requires a time window large enough to detect two major amplitude peaks of a gait signal, usually in the order of 1Hz or 1 second respectively.

In this chapter we present a new algorithm based on a recent neural network tool, the neural ordinary differential equation or neural ODE. Using a very simple six-layer (three connect, three activation) neural network with only three sensor signals and a 0.1 second time window we are able to distinguish between seven different activities at over 95% accuracy on an ultra low resource system.

6.1 Neural Networks

The ideas of neural networks go back to the late 19th century where Bain [1] and James [31] have proposed theories for basic neural networks. In today's terms a neural network is a form of artificial intelligence, which is composed of artificial neurons, which are connected together in a network. [29] Usually, the network consists of several layers of neurons, for example an input layer, an output layer as well as one or more "hidden" layers. Figure 6.1 shows a diagram of a simplified neural network with two inputs, three outputs and two hidden layers of six neurons each. Note that for simplicity not all connections are shown. In very simple terms, each neuron can be thought of as a number indicating its activation and activations in one layer determine the activations in the next layer by means of connections. Inputs are the data feeding the neural network, for example in the context of this dissertation they

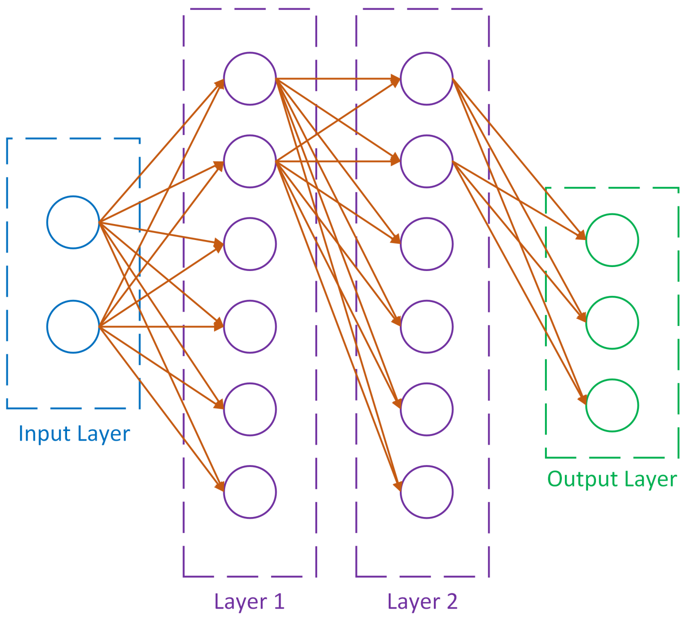
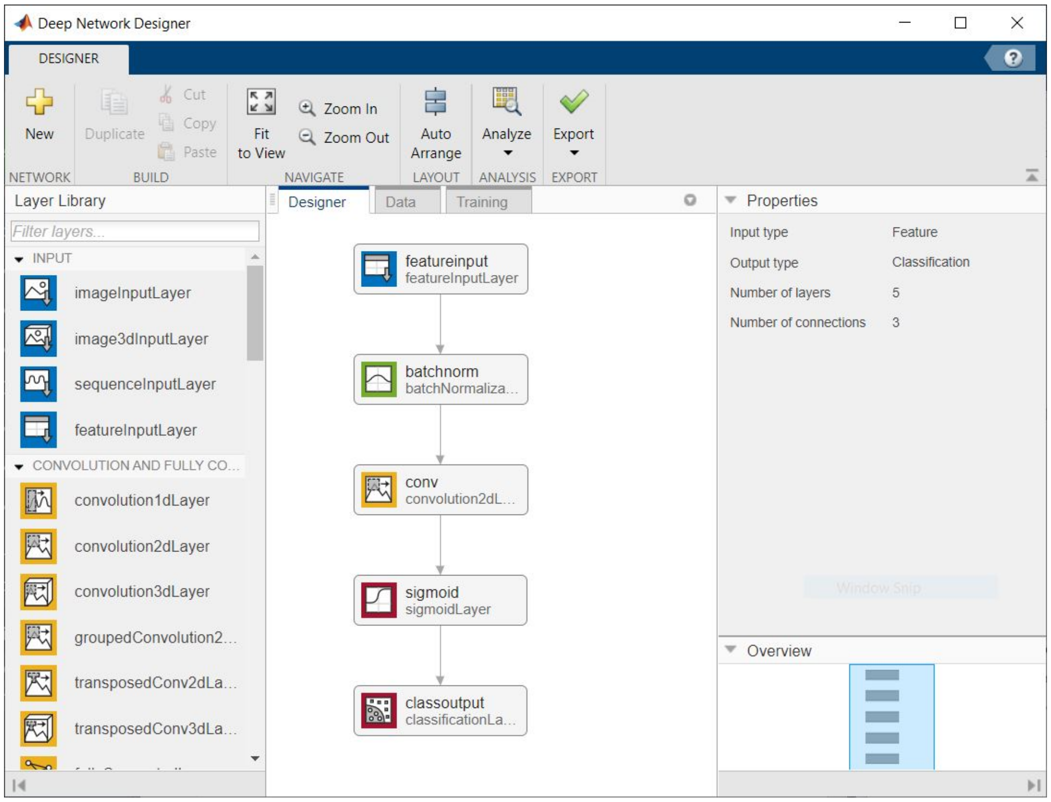


Figure 6.1: Simple Diagram of the Concept of a Neural Network. Circles Depict Neurons, Arrows Depict Connections. For Simplicity Not All Connections Are Shown.

would be sensor signals measured on the human body. Outputs are the results of the network, again in the current context they are the classification variables, which stand for the different activities we are trying to detect. For instance, if the first output neuron were to be activated this could indicate that the algorithm thinks the person is walking.

As mentioned before, Figure 6.1 is a very simplified depiction of a neural network. Today's neural networks can have many more than two inputs, three outputs and two hidden layers of six nodes each. In the often used example where a neural network is used to detect a handwritten letter or number, the input may be a 50x50 pixel image. Moreover, many different types of layers can be used depending on the task. For example, Matlab 2021b provides over 50 different types of layers in its "deep network design" tool, shown in Figure 6.2. All developments presented in this chapter have been performed using Matlab's "Deep Learning Toolbox".



Input type	Feature
Output type	Classification
Number of layers	5
Number of connections	3

Figure 6.2: Matlab Deep Network Design Tool

6.2 Definitions

For clarity we want to define that the term “stride” from hereon is to be understood in the way it is used for neural networks, i.e. the amount of movement of a filter or other machine learning construct over data. This distinction is necessary because we are also talking about activity classification and gait, where the term stride is used to describe the distance covered by a physical step. We use the term “gait stride” for those occasions where stride is meant in the sense of gait analysis.

6.3 Neural Ordinary Differential Equations

One very recently developed type of neural network is the neural ordinary differential equation or neural ODE. As the name implies this type of network is a combination of a neural network with the classical ODEs scientists use to describe almost all physical processes in nature. Mathematically, a neural ODE can be formulated as follows, whereas θ are a set of learnable parameters. [9; 10]

$$\dot{y} = f(y, t, \theta) \tag{6.1}$$

Note, that as with all system modelling, the goal is to learn the parameters, hence the ODE and not the function $y(t)$. As such, neural ODEs can be seen as a new tool of system identification, we are trying to model the dynamics of the underlying system.

While the neural ODE has been used for prediction and estimation, there is little existing research to use this tool for human activity recognition. For example, Siu et. al [60] have used neural ODEs among other algorithms to estimate ankle torques based on electromyography and accelerometer signals. Knowing an estimate of joint torques is not only clinically valuable but can also be used for the design of robotic controls. Siu captured EMG and accelerometer data from eight wireless EMG sensors with embedded 3D accelerometers. Using the neural ODE Siu showed a mean squared error of equal to or less than 0.15 for estimating joint torques from the sensor data for four different activities, standing, walking, running and sprinting.

Kim et. al [34] have used neural ODEs in order to further characterize human gait, specifically walking and running. Their idea is that higher dimensional observations (e.g. large sets of data from smartphones) can be explained better by significantly lower dimensional characteristics, which they call latent variables. They show that neural ODEs are especially useful for this because they allow for a much smaller neural network (with less parameters) due to the fact that they are naturally good at modelling derivatives of states.

Finally, Li et. al [38] use neural ODEs and smartphone data to predict Parkinson's Disease. In the past, Parkinson's has been diagnosed in the office by a physician through evaluation of the quality of a set of activity tests performed by the patient. According to the authors, smartphone data has opened up the possibility of evaluating patients while they are home, however, this method is error-prone because out-of-lab tests have quality problems including irregularly collected observations. Using data from a large scale Parkinson's disease study called "mPower" [8], the authors show improvements in predicting the disease over baseline methods such as recursive neural networks (RNN).

6.3.1 Modelling a Second Order ODE

Let us consider a simple example from Mathworks [41] of modelling a second order differential equation below, given in state-space form, using a neural ODE.

$$\dot{x} = \begin{bmatrix} -0.1 & -1 \\ 1 & -0.1 \end{bmatrix} x \tag{6.2}$$

Knowing the system we are trying to model, we can plugin any number of initial conditions $x_i = [x_{i_1} \quad x_{i_2}]^T$ to calculate the states $\dot{x} = [\dot{x_1} \quad \dot{x_2}]^T$ and use this data to train the neural ODE. We can then compare the trained result to the actual system. Note, we are not solving for $y(t)$ but trying to obtain the underlying dynamics A in $\dot{x} = Ax + Bu$, whereas $B = 0$.

The neural network structure used to model the neural ODE is a simple multi-layer perceptron consisting of five total layers, three fully connected layers and two tanh activation layers, see Figure 6.3. Values in parantheses denote the size of the layers.

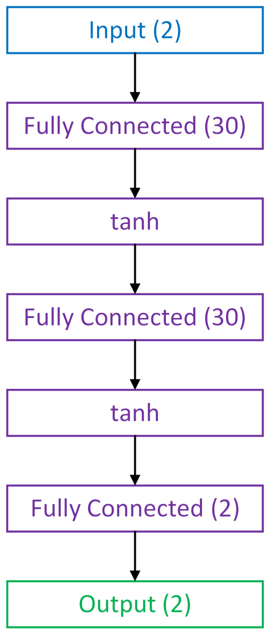


Figure 6.3: Neural Network Architecture for Modelling the 2nd Order ODE From Equation 6.2

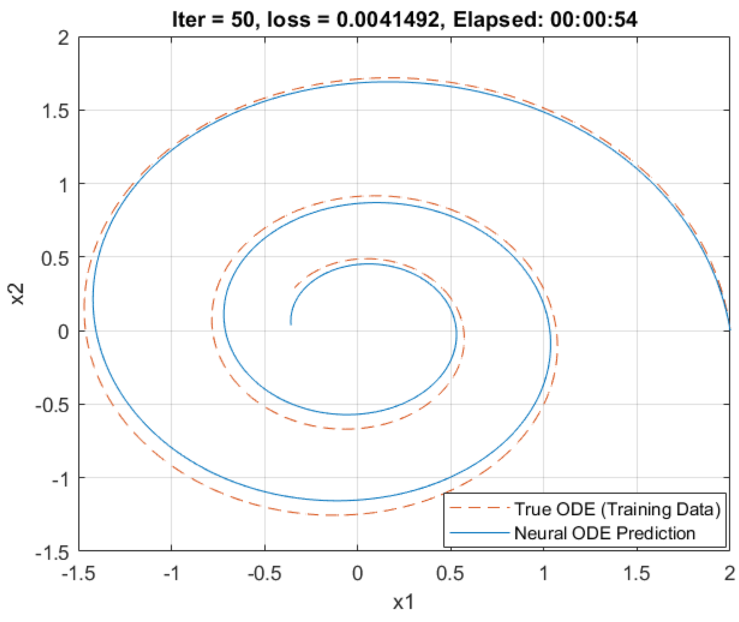
From Equation 6.2 we can see that it describes a second order dynamic system with stiffness and damping and no forcing function in the form of

$$\ddot{y} + b\dot{y} + Ky = 0. \tag{6.3}$$

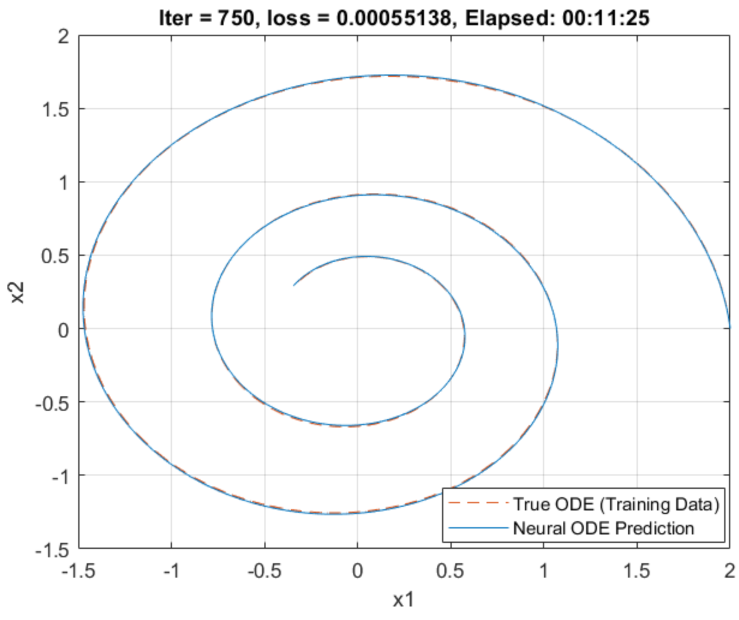
Therefore, we expect the states x_1 and x_2 to be damped sinusoidal wave-forms. When plotting the states against each other in a phase plot during an initial condition response, we expect a function, which starts at the initial condition and then spirals towards zero. This result is shown in Figure 6.4 at two different times during the training cycle. The top and bottom graphs show how the neural ODE prediction matches the true ODE at 50 and 100 iterations, respectively. Even after just 50 iterations, or about one minute of training, the neural ODE approximates the true ODE fairly well. After 750 iterations the two are almost indistinguishable from each other.

We see that neural ODEs on their own are essentially a neural network approach to what we call "system identification" in control theory. However, while a neural ODE on its own is a sophisticated tool to model the dynamics of an unknown system with significant prediction capability, it cannot *directly* be applied for classification tasks. This is because ODEs are inherently continuous while classification is inherently binary.

Let us for example consider the classification of different human activities such as walking, jogging, etc. In order to model this with a neural network we would want to input one or more gait signals (e.g. hip angle) and as result receive a number from 1 to n indicating which activity is being performed. As we see in the next section this requires wrapping the neural ODE with an "outer" network.



(a) True ODE States and States Predicted by the Neural ODE After Training for 50 Iterations



(b) True ODE States and States Predicted by the Neural ODE After Training for 750 Iterations.

Figure 6.4: Training and Prediction Results for the Neural ODE

6.3.2 An Augmented Neural ODE for Classification

In the previous section we have seen how neural ODEs can be used to model the dynamics of a system and therefore they have powerful prediction capability. However, we have also seen that a neural ODE cannot be used for classification directly because ODEs are not meant to be used for producing binary or integer outputs.

One approach to this problem is to wrap the neural ODE with an "outer" neural network. Mathworks [42] provides an example of this approach for the old problem of image processing. The goal is to classify image tiles containing both handwritten and printed numbers into the actual numbers they depict. Figure 6.5 shows an example of 8x8 tiles of images each showing one differently written number. A convolutional

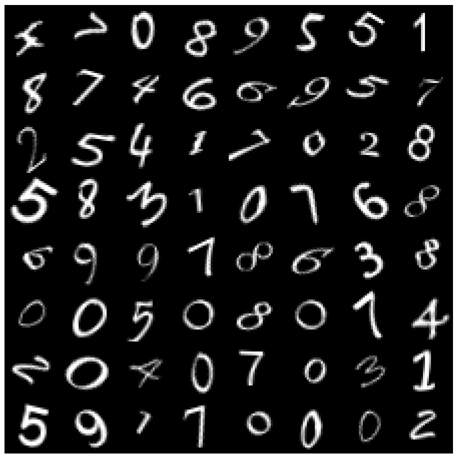


Figure 6.5: Examples of Handwritten Numbers

neural network is proposed for this task and the architecture can be seen in Figure 6.6. An input image of 28x28 pixels is passed through a convolution and ReLU layer. The convolution layer uses 8 3x3 filters and a stride of 2, which results in 8 channels of 14x14 images. The channels are then augmented from 8 to 16. This type of "augmented" ODE has been discussed by [19] and has several advantages including

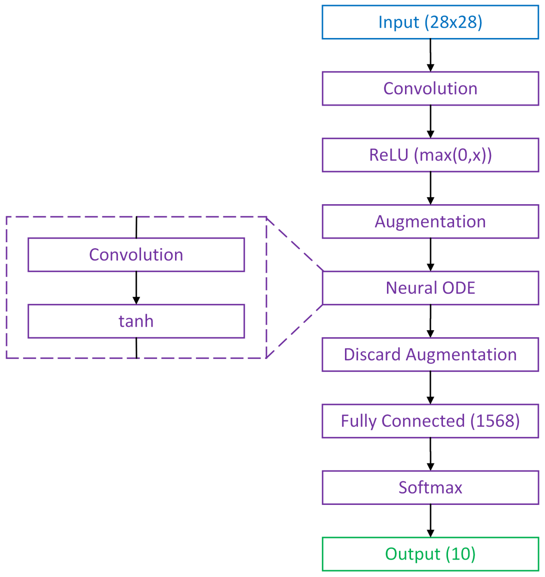


Figure 6.6: Neural Network Structure Using ODE and Suitable for Classification

stability and lower computational cost. After the neural ODE operation the augmentation is removed, the result passed through a fully connect operation and finally a softmax-function is used to obtain the classifier weights. The softmax-function is commonly used as the final operation for neural network classification tasks and defined in Equation 6.4 below. It approximates the max-function using the Euler and logarithmic functions and has the advantage of being smooth and therefore we can use the gradient descent method for optimization. [69]

$$\text{soft}(s_1, s_2) = \log\left(e^{s_1} + e^{s_2}\right) \tag{6.4}$$

As can be seen from the architecture presented above, this type of neural network is suitable for classification tasks. Further, as existing literature has shown, convolutional networks are suitable for human activity recognition.

6.4 A Novel Activity Recognition Algorithm Based on Augmented Neural ODEs

Based on the previously presented architecture we propose a new activity recognition algorithm using neural ODEs, which is suitable to classify human activities. As we will see, this new algorithm requires significantly less data and computing power than earlier presented algorithms based on convolutional networks and graph neural networks.

The classification motivations for this novel algorithm are based on the requirements of the APEx project, which was shown earlier. Using a minimal amount of available sensors, we seek to classify activities into walking, jogging, pushing, squatting, sitting, going up stairs and going down stairs. All these are common tasks that a user performs when wearing the exo-skeleton, although the device only provides power during pushing as well as when returning to an upright position from squatting or sitting. Further, most of these tasks are also common tasks that we all perform every single day. A summary of the activities as well as their assigned classification numbers is shown in Table 6.1.

The proposed algorithm is very similar in architecture to the augmented neural ODE network presented in the previous section. However, we propose to use "gait pictures" as inputs to the neural ODE based network. A schematic of the updated architecture is shown in Figure 6.7.

APEx has angular encoders mounted to each hip as well as 3D IMUs mounted at the pelvis. In terms of their classification value, the hip angle sensors and pelvis accelerometers are the most important. The hip angle sensors allow measuring hip angle, a direct measure of gait, and the 3D accelerometers can be used to obtain an approximate measure of pelvis tilt, which is especially useful for activities like pushing, squatting or sitting. Therefore, we chose to use the left and right hip angles as well as the pelvis tilt as inputs to the algorithm. We further chose to provide the algorithm with a gait picture, consisting of the last 10 samples of the aforementioned signals, resulting in an input gait picture of size 3 by 10, see Figure 6.7. For reference, one such gait picture taken during a walking step is shown in Figure 6.8. Further sample gait pictures can be found in the appendix.

First, the input gait picture is passed through a convolution layer. This first convolution operation consists of 10 3x3 filters with a stride of 1, resulting in a 10 channel output and original gait picture size. During the augmentation phase 10 channels are added for a total of 20 channels being passed to the neural ODE operation. This results in a neural ODE structure of 20 filters to keep the output consistent. During the discard step, the 10 augmentation channels are discarded, resulting in a total of $3 \cdot 10 \cdot 10 = 300$ elements being passed to the fully connect operation. Finally, the output of the fully connect operation is the size (7) of our classification table.

Classification No.	**Activity**
1	walk
2	jog
3	push
4	squat
5	sit
6	stairs-up
7	stairs-down

Table 6.1: Activities As Well As Their Classification Numbers to Be Classified by the Novel Algorithm

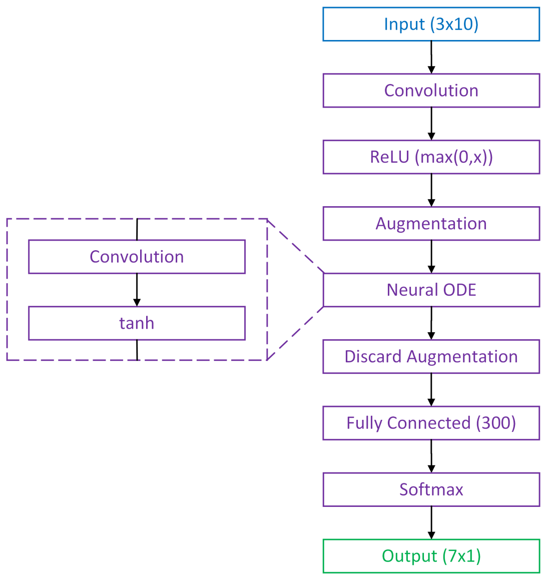


Figure 6.7: Neural Network Structure of a Novel Approach for Human Activity Classification Based on Neural ODE

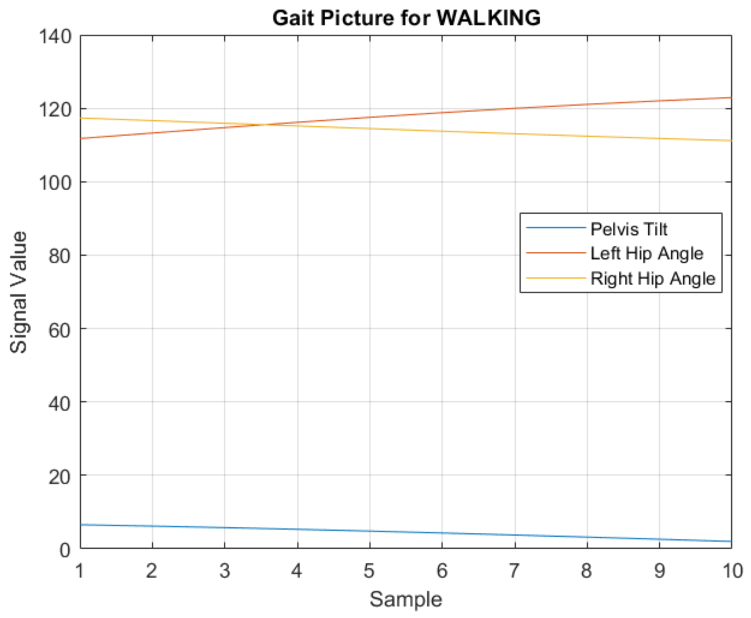


Figure 6.8: A “Gait Picture” Taken During Walking. Pictures Like this One Are the Input to the Neural ODE Algorithm.

The training algorithm, which follows the same procedure as in the previous section, uses mini-batching to reduce the chances that the optimizer, which uses a form of gradient descent, finds merely a local minimum. First, the network is computed until the end of the augmentation step. At this point, the augmentation output is passed as the initial condition to a deep learning ODE45 solver (dlode45). After solving the ODE, the rest of the network is computed. In the final optimization step, the loss is calculated using cross-entropy and a new set of network parameters is determined using ADAM optimizer.

The algorithm is trained using data collected while wearing the APEx but with the APEx being inactive. Only two trials of each activity were collected, one trial was used to train the algorithm and one trial was used to test the trained algorithm. Before being passed to the activity recognition algorithm, both test and training data, are passed through a low-pass filter for cleanup and finally cut into arrays of gait pictures like the one shown in Figure 6.8.

6.4.1 Training and Test Results

The training progress of a typical training run is shown in Figure 6.9. This figure shows the loss over the duration of the training with iterations shown on the x-Axis. The current loss and minimum loss encountered is updated on-the-fly in the title of the figure along with the duration of the training. In terms of absolute improvement it looks like the gains during the initial phase (0-500 iterations) are much larger than for the remainder of the training run. However, in terms of relative improvement, the gain from 0 to 500 iterations is about 14 times while the improvement from 500 to the end (about 3000) is more than another 70 times with a final loss of 0.007.

Once training has been completed, the resulting trained model is provided with the test data set, which the algorithm has never seen before. The algorithm's prediction

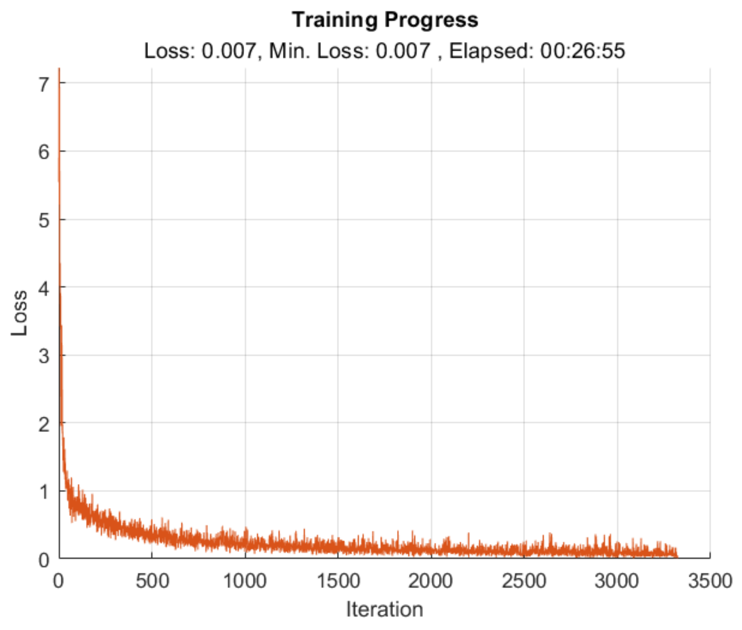
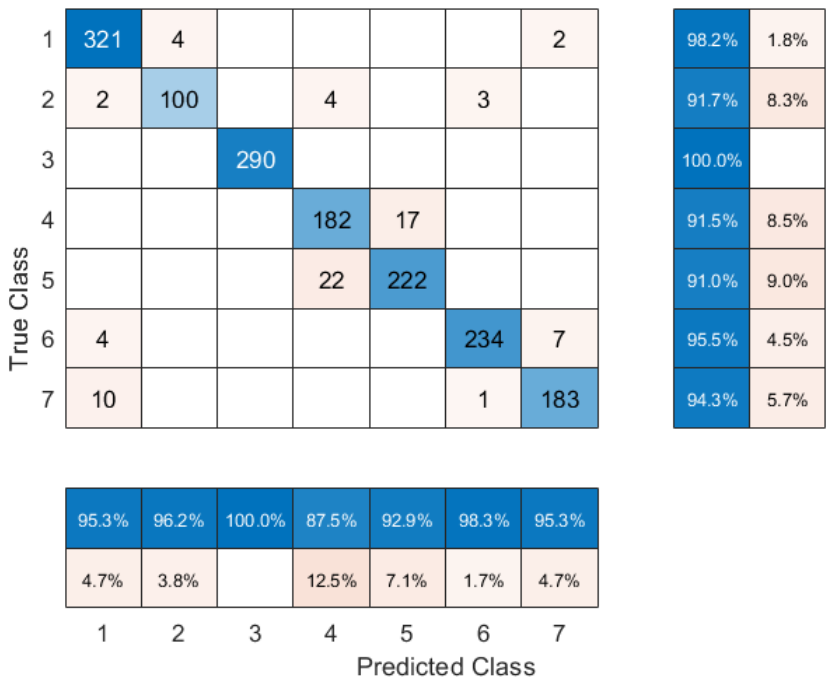


Figure 6.9: Example of Training Progress (Loss) of the Novel Algorithm

of activity classification is then compared with the true activity class. The resulting confusion matrix is shown in Figure 6.10. The overall accuracy across the entire test data set including all activities is more than 95%. As can be seen, the algorithm has the most difficulty distinguishing between squatting (4) and sitting (5) as well as between walking (1) and going down-stairs (7). These problems are not new. Gait signals for these two pairs of activities are very similar. The largest error rate at 12.5% and 7.1% occurs between squatting and sitting respectively followed by 4.7% down-stairs activity, which is classified as walking. However, these results are promising because small data set was used and a simple consumer laptop with only four cores at 1.8GHz and 16GB RAM was used for training. Most importantly, this algorithm requires a time window of only 100ms, significantly faster than any previous literature we reviewed in this work. Also, note that for the APEx device, the above mentioned errors in classification would be of reduced impact because the device behaves similarly during squatting and sitting, both are activities where the device



True Class \ Predicted Class	1	2	3	4	5	6	7		
1	321	4					2	98.2%	1.8%
2	2	100		4		3		91.7%	8.3%
3			290					100.0%	
4				182	17			91.5%	8.5%
5				22	222			91.0%	9.0%
6	4					234	7	95.5%	4.5%
7	10					1	183	94.3%	5.7%
	95.3%	96.2%	100.0%	87.5%	92.9%	98.3%	95.3%		
	4.7%	3.8%		12.5%	7.1%	1.7%	4.7%		

Figure 6.10: Confusion Matrix Obtained Using the Separate Test Data Set

should support the user. At the same time, both walking and down-stairs activities are unpowered and should be treated the same way by the device. Finally, the algorithm is able to classify 100% of pushing activities, which is especially important for APEx because the device must recognize it reliably in order to provide support.

6.5 Conclusion and Discussion

In this chapter a novel algorithm using neural ODEs for human activity classification was presented. Neural ODEs are of particular interest for activity classification because of three reasons. First, they represent a neural network approach of the proven tool all scientists and engineers use to analyze the real world, ordinary differential equations. Second, as a result of this basic characteristic they are good at reducing the dimensionality of neural network problems, which has the potential to simplify computation. Third, as differential equations, they also provide support for time sequences.

Training and test results for this novel algorithm were presented and showed that the neural ODE based network can achieve good classification accuracy with a small data set. Of particular importance is that this approach requires only a small time window of 100ms, which is of particular advantage during gait transitions. However, we have also seen that some open questions remain, especially if this algorithm is to be used for other devices where a higher reliability in distinguishing between squatting and sitting as well as walking and going down stairs is required.

Finally, this work also opens up new possibilities for further research. For example, recurrent neural networks such as long-short-term memory (LSTM) could be combined with neural ODEs instead of convolutional layers. The effect of sample times, time window and also additional sensor data could be investigated. We will present a more thorough discussion of these possibilities in the final chapter.

Chapter 7

CONTRIBUTIONS TO THE FIELD OF ROBOTICS

In this chapter a summary of work completed and contributions to the field of Robotics are presented. At the end of this chapter we make suggestions for future areas of research based on the work presented in this dissertation.

This dissertation examined wearable robotic systems from a systems perspective and covered the areas of electronics and controls. Even though technology has made a lot of progress and thus has made wearable devices more feasible, difficult design challenges remain due to the unique requirements of wearable devices.

In Chapter 3 we have seen that a “one-fits-all” solution does not exist, neither for important devices such as motors nor for the design process. The importance of searching for “hidden” characteristics and the impact of large currents were highlighted and how they can influence the entire design. Moreover, two successful electronic designs for the HeSA and APEx exoskeletons were presented.

In Chapter 4 important control system aspects were covered. If the reader only remembers one of the presented elements it should be that operation in **real-time** is crucial for any kind of robotic device. Real-time is of particular importance because without it our entire underlying framework of mathematics and physics vanishes. Further, the successful control system architecture was shown for the HeSA and APEx exoskeletons.

In Chapter 5 we presented a real-time implementation of the FFT on a 16-bit micro-controller. The FFT was used to compute frequencies and amplitudes of a gait signal at an update rate of 50Hz. The results have shown that the FFT has potential as a real-time signal characteristic extraction tool.

Finally, in Chapter 6 a novel activity recognition algorithm based on neural ODEs was presented. Using "gait pictures" consisting of a window of samples of three different sensors, we showed good classification accuracy with a comparatively small data set. Most importantly, this approach allows activity classification with a time window of only 0.1 seconds. Therefore, this algorithm performs especially well during gait transitions where existing neural networks often fall short.

7.1 Completed Publications

C1. *AW Boehler, KW Hollander, TG Sugar, D Shin, "Design, implementation and test results of a robust control method for a powered ankle foot orthosis (AFO)", ICRA 2008.*

In this paper I present detailed DC/EC motor modelling as well as implementation and test results of a novel controls approach for AFOs. The state based approach uses easily measurable signals to transition between five states during gait.

C2. *MA Holgate, TG Sugar, AW Bohler, "A novel control algorithm for wearable robotics using phase plane invariants", ICRA 2009.*

I have performed in-depth real-time analysis and implementation of this algorithm for a powered robotic ankle prosthesis.

C3. *MA Holgate, AW Bohler, TG Sugar, "Control algorithms for ankle robots: A reflection on the state-of-the-art and presentation of two novel algorithms", RAS & EMBS international conference on biomedical robotics and biomechatronics, 2008.*

In this paper I present a novel control algorithm for ankle robots, which uses the Fast Fourier Transform, polynomial fitting and matrix manipulations to scale a gait pattern both in time and amplitude.

C4. *Joseph Hitt, A Mehmet Oymagil, Thomas Sugar, Kevin Hollander, Alex Boehler, Jennifer Fleeger, "Dynamically controlled ankle-foot orthosis (DCO) with regenerative kinetics: Incrementally attaining user portability", ICRA 2007.*
I have developed the electronics system and control code for the orthosis presented in this paper.

C5. *Martin, W. B., Boehler, A. Hollander, K. W. Kinney, D., Hitt, J. K., Kudva, J., Sugar, T. G., "Aerial Porter Exoskeleton (APEX) for Lifting and Pushing", International Symposium on Wearable Robotics, 2020.*
I have developed the electronics system as well as the control system architecture and base control code for the exoskeleton presented in this paper.

J1. *Martin Grimmer, Matthew Holgate, Robert Holgate, Alexander Boehler, Jeffrey Ward, Kevin Hollander, Thomas Sugar, André Seyfarth, "A powered prosthetic ankle joint for walking and running", Biomedical engineering online 2016.*
I have developed the electronics system as well as the control system architecture, base control code and parts of the higher level algorithms for the prosthetic ankle presented in this paper.

P1. *T Sugar, JK Hitt, A Boehler, K Hollander, JA Ward, "Method and apparatus for harvesting energy from ankle motion", US Patent 8,716,877, 2014.*
I have developed actuator and motor (generator) modelling for this patent.

7.2 Expected Future Publications

J1. *Martin, W. B., Boehler, A. Hollander, K. W. Kinney, D., Hitt, J. K., Kudva, J., Sugar, T. G., "Development and Testing of the Aerial Porter Exoskeleton (APEx)", Submitted to Wearable Technologies, 2021.*

I have developed the electronics system as well as the control system architecture and base control code for the exoskeleton presented in this paper.

7.3 Future Work

7.3.1 The Area of FFT for Activity Recognition

In Chapter 5 a real-time implementation of the FFT was presented. A big advantage of the FFT over most neural networks is that it is real-time ready and has been implemented in real-time in the past. Unfortunately, the FFT alone is not sufficient to recognize gait activities, it represents merely a processing step to extract basic information about a signal.

A future opportunity for research in this field would be to experiment with the size of the time window to research a feasible lower end of the window, which still yields usable information for activity recognition. It is important to note, that a smaller time window also results in a reduction of FFT complexity because the amount of samples is directly related to the size of the matrices used for computation. Therefore, a combination of a smaller time window FFT with a "light" neural network approach may yield a real-time capable solution.

7.3.2 The Area of Neural ODEs for Activity Recognition

In Chapter 6 an activity recognition algorithm based on neural ODEs was presented. The most promising aspect of this approach is that it requires a time window of only 100ms and therefore is able to detect gait transitions. This is significant because most of today's activity recognition models require much longer time windows and thus deserves further investigation. Unfortunately, the path to real-time for neural ODEs and neural networks in general is less clear. Neural networks are good at learning complex relationships but, just like the human brain they are modelled after, they require a large amount of learning units (connections). In a mathematical sense neural networks are essentially matrix calculations of epic proportions because the size of the matrices can be easily in the hundreds or even thousands. For example, the size of the weights of the neural ODE layer presented in Chapter 6 is 3x3x20x20 for a total of 3600 elements. Generally, processors are quite fast at multiplication but the sheer number of multiplications and additions required to compute the network output is likely to put a large strain on most currently available micro-controllers.

Therefore, one potential area of research is how to enable real-time capability for the presented approach. This likely requires an approach from all sides, optimization of the algorithm as well as careful and efficient code design. Fortunately, because neural networks will likely be used on a high level, i.e. detection of activity, rather than a low level, i.e. actuator command generation, the network can be computed at a much lower rate, for example 10Hz.

From a high level perspective, a real-time capable algorithm must address the following challenges:

- Calculation of the network output
- Neural ODE solver
- Determinism

For a neural ODE algorithm to be real-time capable the first two items above must be able to execute within each time step. This is because the computation of the network output requires the neural ODE solver as part of the neural ODE layer. Therefore, calculating the network output requires a processor to perform all the weights and bias computations in addition to finding an ODE solution, for example with an ode45 algorithm. It is conceivable that a state-of-the-art embedded 32-bit micro-controller dedicated to the neural network computation in combination with a relaxed sample time (e.g. in the order of 50-200ms) could be up to the task.

Finally, there is the challenge of *determinism.* Determinism is impacted by both, the ode45 solver (ODE solvers in general are not deterministic) as well as the nature of the neural network, which is also non-deterministic. For example, a particular gait picture may cause the ode45 solver to take a long time to find a solution due to being close to a singularity. If this happens, it must be detected and resolved such that real-time execution is not compromised. The operating system or other monitoring entity can be used for this purpose. On the other hand, neural networks are non-deterministic by nature. For instance, it is possible that even just changing one data point in a gait picture can lead to a completely different categorization of the picture. However, this shortcoming can be addressed to some degree by using the neural network *only* for high level control rather than for lower level path generation.

It is also possible that we can use other gait information, for example a center-of-gravity (COG) model, to cross-check the output of the neural network.

In regards the neural ODE algorithm itself, there are also a multitude of opportunities for further research. The main goals of these activities are to increase fidelity and optimize the approach.

- Research the performance of alternative layers such as long-short-term memory (LSTM).
- Research the influence of different time window sizes.
- Train the algorithm with larger data sets.
- Investigate the use of different sensor signals or possibly add sensor signals to improve fidelity.
- Research using the outputs of existing algorithms, such as a center-of-gravity model, as additional inputs to the neural network.

REFERENCES

[1] Bain, A., *Mind and body : the theories of their relation* (London : Routledge/Thoemmes Press 2000, 1818-1903).

[2] Barry, P. and P. Crowley, "Chapter 4 - embedded platform architecture", in "Modern Embedded Computing", edited by P. Barry and P. Crowley, pp. 41–97 (Morgan Kaufmann, Boston, 2012), URL `https://www.sciencedirect.com/science/article/pii/B9780123914903000047`.

[3] Bionics, E., "Ekso nr", `https://eksobionics.com/eksohealth/`, [Online; accessed Oct 14, 2021] (2011-2019).

[4] Birbaumer, N., "Brain–computer-interface research: coming of age.", Clin. Neurophysiol. **117**, 479–483 (2006).

[5] Bluetooth SIG, I., "Bluetooth sig", `https://www.bluetooth.com/`, [Online; accessed Oct 14, 2021] (2021).

[6] BMI, L., "Biomedical informatics and ehealth laboratory", `https://bmi.hmu.gr/the-mobifall-and-mobiact-datasets-2/`, [Online; accessed Oct 14, 2021] (2021).

[7] Boehler, A. W., K. W. Hollander, T. G. Sugar and D. Shin, "Design, Implementation and Test Results of a Robust Control Method for a Powered Ankle Foot Orthosis (AFO).", IEEE International Conference on Robotics and Automation (ICRA) pp. 2025–2030 (2008).

[8] Bot, B. M., C. Suver, E. C. Neto, M. Kellen, A. Klein, C. Bare, M. Doerr, A. Pratap, J. Wilbanks, E. R. Dorsey, S. H. Friend and A. D. Trister, "The mpower study, parkinson disease mobile data collected using researchkit", Scientific data **3**, 1, 160011–160011 (2016).

[9] Callh, S., "Forecasting the weather with neural odes", `https://sebastiancallh.github.io/post/neural-ode-weather-forecast/`, [Online; accessed Oct 14, 2021] (2020).

[10] Chen, R. T. Q., Y. Rubanova, J. Bettencourt and D. Duvenaud, "Neural ordinary differential equations", (2019).

[11] Chinimilli, P. T., S. Redkar and T. Sugar, "A Two-Dimensional Feature Space-Based Approach for Human Locomotion Recognition.", IEEE Sensors Journal **19**, 11, 4271–4282 (2019).

[12] Chinimilli, P. T., S. C. Subramanian, S. Redkar and T. Sugar, "Human Locomotion Assistance using Two-Dimensional Features Based Adaptive Oscillator.", Wearable Robotics Association Conference pp. 92–98 (2019).

[13] christopherreeve.org, "One degree of separation: Paralysis and spinal cord injury in the united states.", https://www.ncart.us/uploads/userfiles/files/one-degree-of-separation.pdf, [Online; accessed Oct 14, 2021] (2009).

[14] Contreras-Vidal, J. L. and R. G. Grossman, "NeuroRex: A Clinical Neural Interface Roadmap for EEG-based Brain Machine Interfaces to a Lower Body Robotic Exoskeleton.", 35th Annual International Conference of the IEEE EMBS pp. 1579–1582 (2013).

[15] Corinthios, M., K. Smith and J. Yen, "A parallel radix-4 fast fourier transform computer", IEEE transactions on computers **C-24**, 1, 80–92 (1975).

[16] Das, A. D. and K. K. Mahapatra, "Real-time implementation of fast fourier transform (fft) and finding the power spectrum using labview and compactrio", Proceedings of the 2013 International Conference on Communication Systems and Network Technologies pp. 169–173 (2013).

[17] De La Fuente, J., S. C. Subramanian, T. G. Sugar and S. Redkar, "A robust phase oscillator design for wearable robotic systems", Robotics and autonomous systems **128**, 103514– (2020).

[18] Do, A. H., P. T. Wang, C. E. King, S. N. Chun and Z. Nenadic, "Brain-computer interface controlled robotic gait orthosis.", Journal of NeuroEngineering and Rehabilitation **10**, 111, 1–9 (2013).

[19] Dupont, E., A. Doucet and Y. W. Teh, "Augmented neural odes", (2019).

[20] Fleischer, C., C. Reinicke and G. Hommel, "Predicting the Intended Motion with EMG Signals for an Exoskeleton Orthosis Controller.", IEEE International Conference Robotics Autonomous Systems pp. 2029–2034 (2005).

[21] Force, U. A., "Travis afb partnership springs air force forward with new aerial porter exoskeleton", https://www.af.mil/News/Article-Display/Article/2621624/travis-afb-partnership-springs-air-force-forward-with-new-aerial-porter-exoskel/, [Online; accessed Oct 14, 2021] (2021).

[22] Foundation, B., "beagleboard.org", https://beagleboard.org/, [Online; accessed Oct 14, 2021] (2021).

[23] Foundation, R. P., "raspberrypi.org", https://www.raspberrypi.org/, [Online; accessed Oct 14, 2021] (2021).

[24] Group, E. T., "Ethercat website", https://www.ethercat.org, [Online; accessed Oct 14, 2021] (2021).

[25] Guizzo, E. and H. Goldstein, "The rise of the body bots.", IEEE Spectrum **32**, 10, 50–56 (2005).

[26] Heideman, M. T., D. H. Johnson and C. S. Burrus, "Gauss and the history of the fast fourier transform", Archive for History of Exact Sciences **34**, 3, 265–277, URL `http://www.jstor.org/stable/41133773` (1985).

[27] Hinterberger, T., A. Kübler, I. Iversen, J. Perelmouter, E. Taub, H. Flor, N. Ghanayim, N. Birbaumer and B. Kotchoubey, "A spelling device for the paralysed.", Nature **398**, 297–298 (1999).

[28] Hollander, K. W. and T. G. Sugar, "A Robust Control Concept for Robotic Ankle Gait Assistance.", IEEE 10th International Conference on Rehabilitation Robotics (ICORR) pp. 119–123 (2007).

[29] Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences of the United States of America **79**, 8, 2554–2558 (1982).

[30] Huo, W., S. Mohammed, J. C. Moreno and Y. Amirat, "Lower Limb Wearable Robots for Assistance and Rehabilitation: A State of the Art.", IEEE Systems Journal **10**, 3, 1068–1081 (2016).

[31] James, W., *The principles of psychology*, Classics in psychology, 1855-1914 ; 27-28. (Thoemmes Press, Bristol, U.K, 1998 - 1890).

[32] Kawamoto, H. and Y. Sankai, "Power assist system HAL-3 for gait disorder person.", Proc. ICCHP **2398**, 196–203, [Lecture Notes on Computer Science] (2002).

[33] Kazerooni, H. and R. Steger, "The Berkeley lower extremity exoskeleton.", Trans. ASME Journal of Dynamic Systems Measurement and Control **128**, 14–25 (2006).

[34] Kim, J., J. Lee, W. Jang, S. Lee, H. Kim and J. Park, "Two-stage latent dynamics modeling and filtering for characterizing individual walking and running patterns with smartphone sensors", Sensors (Basel, Switzerland) **19**, 12, 2712, 2554–2558 (2019).

[35] Kübler, A., B. Kotchoubey, J. Kaiser, J. R. Wolpaw and N. Birbaumer, "Brain–computer communication: unlocking the locked in.", Psychol. Bull. **127**, 358–375 (2001).

[36] Kübler, A., N. Neumann, J. Kaiser, B. Kotchoubey, T. Hinterberger and N. P. Birbaumer, "Brain-computer communication: Self-regulation of slow cortical potentials for verbal communication", Archives of physical medicine and rehabilitation **82**, 11, 1533–1539 (2001).

[37] Lebedev, M. A. and M. A. L. Nicolelis, "Brain–machine interfaces: past, present and future.", TRENDS in Neurosciences **29**, 9, 536–546 (2006).

[38] Li, W., W. Zhu, E. R. Dorsey and J. Luo, "Predicting parkinson's disease with multimodal irregularly collected longitudinal smartphone data", CoRR **abs/2009.11999**, URL `https://arxiv.org/abs/2009.11999` (2020).

[39] Magotra, N., J. Bitto and J. McCoy, “Real time implementation of multidimensional fast fourier transforms”, in “International Conference on Acoustics, Speech, and Signal Processing”, pp. 1767–1770 vol.3 (IEEE, 1990).

[40] Martin, W. B., *Development of an Aerial Porter Exoskeleton and Exoskeleton Standardization Metrics*, Ph.D. thesis, Arizona State University (2021).

[41] Mathworks, I., “Training a neural ode network”, `https://www.mathworks.com/help/deeplearning/ug/train-neural-ode-network.html`, [Online; accessed Jun 8, 2021] (2021).

[42] Mathworks, I., “Training a neural ode network”, `https://www.mathworks.com/help/deeplearning/ug/train-neural-ode-network.html#TrainAugmentedNeuralODENetworkExample-12`, [Online; accessed Oct 14, 2021] (2021).

[43] McDaid, A. J., S. Xing and S. Q. Xie, “Brain Controlled Robotic Exoskeleton for Neurorehabilitation.”, IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) pp. 1039–1044 (2013).

[44] Mondal, R., D. Mukherjee, P. K. Singh, V. Bhateja and R. Sarkar, “A new framework for smartphone sensor-based human activity recognition using graph neural network”, IEEE sensors journal **21**, 10, 11461–11468 (2021).

[45] NSCISC, “Spinal cord injury facts and figures at a glance”, `https://www.nscisc.uab.edu/Public/Facts%20and%20Figures%202020.pdf`, [Online; accessed Oct 14, 2021] (2020).

[46] Obermaier, B., G. Muller and G. Pfurtscheller, “”virtual keyboard” controlled by spontaneous eeg activity”, IEEE transactions on neural systems and rehabilitation engineering **11**, 4, 422–426 (2003).

[47] Obermaier, B., C. Neuper, C. Guger and G. Pfurtscheller, “Information transfer rate in a five-classes brain-computer interface”, IEEE transactions on neural systems and rehabilitation engineering **9**, 3, 283–288 (2001).

[48] Ostroff, J., S. Gerhart, D. Craigen, T. Ralston, N. G. Leveson, J. Bowen and V. Stavridou, *Real-Time and Safety-Critical Systems. In: High-Integrity System Specification and Design* (Springer, London, 1999).

[49] Pateriya, P. and A. Chaora, “Design and implementation of real-time 16-bit fast fourier transform using numerically controlled oscillator and radix-4 ditfft algorithm in vhdl”, International Research Journal of Engineering and Technology (IRJET) **3**, 5, 2904–2908 (2016).

[50] Patil, S. and E. W. Abel, “Real time continuous wavelet transform implementation on a dsp processor”, Journal of Medical Engineering & Technology **33**, 3, 223–231 (2009).

[51] Perera, N., A. D. Rajapakse and A. M. Gole, "Real-time implementation of discrete wavelet transform for transient type protection applications", 2010 IEEE Electrical Power Energy Conference pp. 1–6 (2010).

[52] Pfeiffer, O., A. Ayre and C. Keydel, *Embedded Networking with CAN and CANopen* (Annabooks, 2003).

[53] Pons, J. L., "Rehabilitation Exoskeletal Robotics.", IEEE Engineering in Medicine and Biology Magazine **29**, 3, 57–63 (2010).

[54] Quotb, A., Y. Bornat and S. Renaud, "Wavelet transform for real-time detection of action potentials in neural signals", Frontiers in Neuroengineering **4**, 7 (2011).

[55] ReWalk, "Rewalk personal 6.0 system", `http://rewalk.com/rewalk-personal-3/`, [Online; accessed Oct 14, 2021] (2019-2019).

[56] Ronao, C. A. and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks", Expert systems with applications **59**, 235–244 (2016).

[57] Ronsse, R., N. Vitiello, T. Lenzi, J. van den Kieboom, M. C. Carrozza and A. J. Ijspeert, "Human–Robot Synchrony: Flexible Assistance Using Adaptive Oscillators.", IEEE Transactions on Biomedical Engineering **58**, 4, 1001–1011 (2011).

[58] Sams, H. W., *Handbook of electronic tables & formulas* (H. W. Sams, Indianapolis, 1979), 5th ed. edn.

[59] Schroeder, K. A., J. De La Fuente, T. G. Sugar and T. Flaven, "A Novel Force Sensitive Resistor Wheatstone Bridge for Prosthesis Control", 38th Mechanisms and Robotics Conference **Volume 5B**, URL `https://doi.org/10.1115/DETC2014-34452`, v05BT08A074 (2014).

[60] Siu, H. C., J. Sloboda, R. J. McKindles and L. A. Stirling, "A neural network estimation of ankle torques from electromyography and accelerometry", IEEE Transactions on Neural Systems and Rehabilitation Engineering **29**, 1624–1633 (2021).

[61] Sugar, T. G., E. Fernandez, D. Kinney, K. W. Hollander and S.Redkar, "HeSA, Hip Exoskeleton for Superior Assistance.", Robotics: Challenges and Trends. Biosystems & Biorobotics **16**, 319–323, [Online; accessed Oct 14, 2021] (2016).

[62] Tan, D. W., M. A. Schiefer, M. W. Keith, J. R. Anderson, J. Tyler and D. J. Tyler, "A neural interface provides long-term stable natural touch perception", Science Translational Medicine **6**, 257, 257ra138–257ra138, URL `https://stm.sciencemag.org/content/6/257/257ra138` (2014).

[63] Technologies, B., "Iot wireless technologies", `https://behrtech.com`, [Online; accessed Oct 14, 2021] (2020).

[64] Technology, P. S., "Wire gauge and current limits including skin depth and strength", https://www.powerstream.com/Wire_Size.htm, [Online; accessed Oct 14, 2021] (2000).

[65] U-BloxAG, "U-blox website", https://www.u-blox.com, [Online; accessed Oct 14, 2021] (2021).

[66] UNFPA, "Ageing in the twenty-first century: A celebration and a challenge", https://www.unfpa.org/sites/default/files/pub-pdf/Ageing%20report.pdf, [Online; accessed Oct 14, 2021] (2012).

[67] Valvano, J. W., *Embedded Systems, Real-time Operating Systems for Arm Cortex-M Microcontrollers* (Self-published, Online, 2017).

[68] Van Loan, C., *Computational Frameworks for the Fast Fourier Transform* (Society for Industrial and Applied Mathematics, 1992), URL https://epubs.siam.org/doi/abs/10.1137/1.9781611970999.

[69] Watt, J., R. Borhani and A. K. Katsaggelos, *Classification*, p. 73–128 (Cambridge University Press, 2016).

[70] Website, A., "arduino.cc", https://www.arduino.cc/, [Online; accessed Oct 14, 2021] (2021).

[71] Wikipedia, "Bluetooth", https://en.wikipedia.org/wiki/Bluetooth, [Online; accessed Oct 14, 2021] (2021).

[72] Wikipedia, "Inductance", https://en.wikipedia.org/wiki/Inductance, [Online; accessed Oct 14, 2021] (2021).

[73] Wikipedia, "Rs-485", https://en.wikipedia.org/wiki/RS-485, [Online; accessed Oct 14, 2021] (2021).

[74] Wolpaw, J. R., N. Birbaumer, D. J. McFarland, G. Pfurtscheller and T. M. Vaughan, "Brain–computer interfaces for communication and control.", Clin. Neurophysiol. **113**, 767–791 (2002).

[75] Yamamoto, K., K. Hyodo, M. Ishii and T. Matsuo, "Development of power assisting suit for assisting nurse labor.", JSME Int. J. Series C. **45**, 3, 703–711 (2002).

[76] Young, A. J. and D. P. Ferris, "State of the Art and Future Directions for Lower Limb Robotic Exoskeletons.", IEEE Transactions on Neural Systems and Rehabilitation Engineering **25**, 2, 171–182 (2017).

[77] Zhou, F., L. Li, K. Zhang and G. Trajcevski, "Urban flow prediction with spatial–temporal neural odes", Transportation research. Part C, Emerging technologies **124**, 102912– (2021).

APPENDIX A

COMPILATION OUTPUTS

A.1 Compilation Output Using m-Function y=fft()

```
xc16-ld 1.40 (A)

Default Code Model: Small
Default Data Model: Large
Default Scalar Model: Small

"program" Memory  [Origin = 0x200, Length = 0x2aa00]

section                    address   length (PC units)   length (
   bytes) (dec)
-------                    -------   -----------------
   --------------------
.text                        0x200             0x13ac          0
   x1d82  (7554)
.const                      0x15ac              0xff0          0
   x17e8  (6120)
.text                       0x259c             0x5528          0
   x7fbc  (32700)
.dinit                      0x7ac4              0x2d2           0
   x43b  (1083)
.text                       0x7d96              0x4aa           0
   x6ff  (1791)

                 Total "program" memory used (bytes):         0
                    xc060  (49248) 18%


"data" Memory  [Origin = 0x800, Length = 0x7800]

section                    address      alignment gaps    total
   length  (dec)
-------                    -------      --------------
   -------------------
.nbss                        0x800                   0
   0x3e  (62)
.ndata                       0x83e                   0
               0x4  (4)
.nbss                        0x842                   0
               0x4  (4)
.ndata                       0x846                   0
               0x2  (2)
.nbss                        0x848                   0
               0x2  (2)
.data                        0x84a                   0            0
   x294  (660)
.bss                         0xade                   0            0
   x1b2  (434)
.bss                         0xc90                   0
   0x6c  (108)
.data                        0xcfc                   0
```

```
   0x54  (84)
.bss                        0xd50                   0
               0x4  (4)
.data                       0xd54                   0
               0x2  (2)
.bss                        0xd56                   0
               0x2  (2)
.heap                       0xd58                   0           0
   x1000  (4096)
_057FEE405df313bd          0x7e00                   0
   0x3c  (60)
.bss                       0x7e3c                   0
   0x8a  (138)
.data                      0x7ec6                   0
   0x22  (34)
.bss                       0x7ee8                   0
   0x18  (24)
_04CCD9605df313be          0x7f00                   0
   0x80  (128)
_057FEF605df313bd          0x7f80                   0
   0x3c  (60)
.data                      0x7fbc                   0
   0x1a  (26)
.bss                       0x7fd6                   0
   0x2a  (42)

                 Total "data" memory used (bytes):          0x1758
                     (5976) 19%


Dynamic Memory Usage

region                    address                      maximum
   length  (dec)
------                    -------
   ---------------------
heap                        0xd58                               0
   x1000  (4096)
stack                      0x1d58                               0
   x60a8  (24744)

                 Maximum dynamic memory (bytes):          0x70a8
                    (28840)

Note: Project is using a large data memory model when small data
   memory model is sufficient.
```

APPENDIX B

SUPPORTING MATERIAL FOR THE NEURAL ODE ALGORITHM

B.1 Definitions

For clarity we want to define that the term "stride" in this chapter is to be understood in the way it is used for neural networks, i.e. the amount of movement of a filter or other machine learning construct over data. This distinction is necessary because we are also talking about activity classification and gait, where the term stride is used to describe the distance covered by a physical step. We use the term "gait stride" for those occasions where stride is meant in the sense of gait analysis.

B.2 Gait Pictures

Gait pictures taken at a random time during each activity are shown below. Pictures like these are the input for the neural ODE activity recognition algorithm.

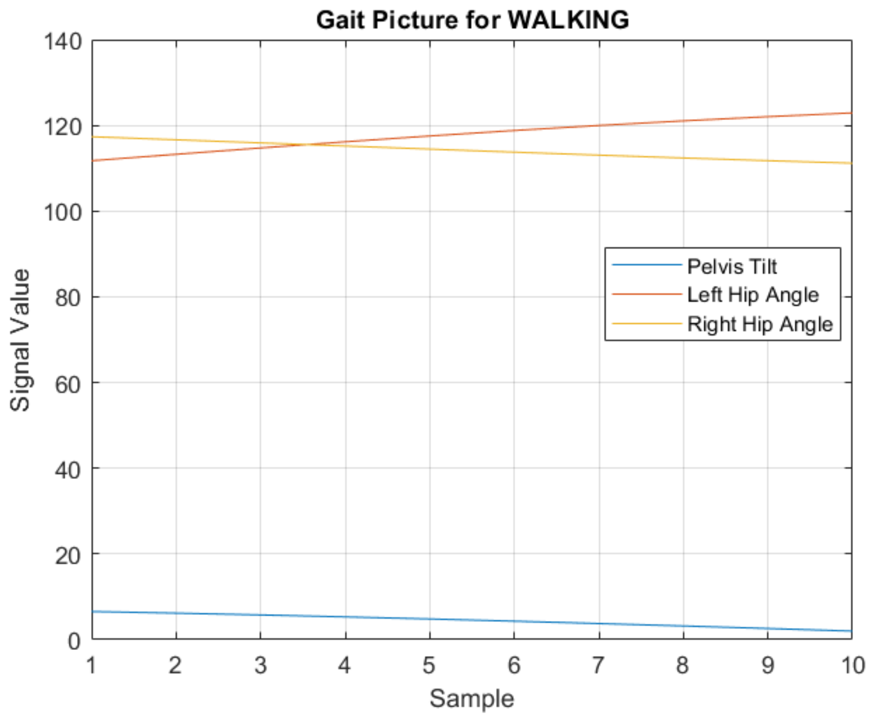


Figure B.1: Gait Picture Taken During Walking

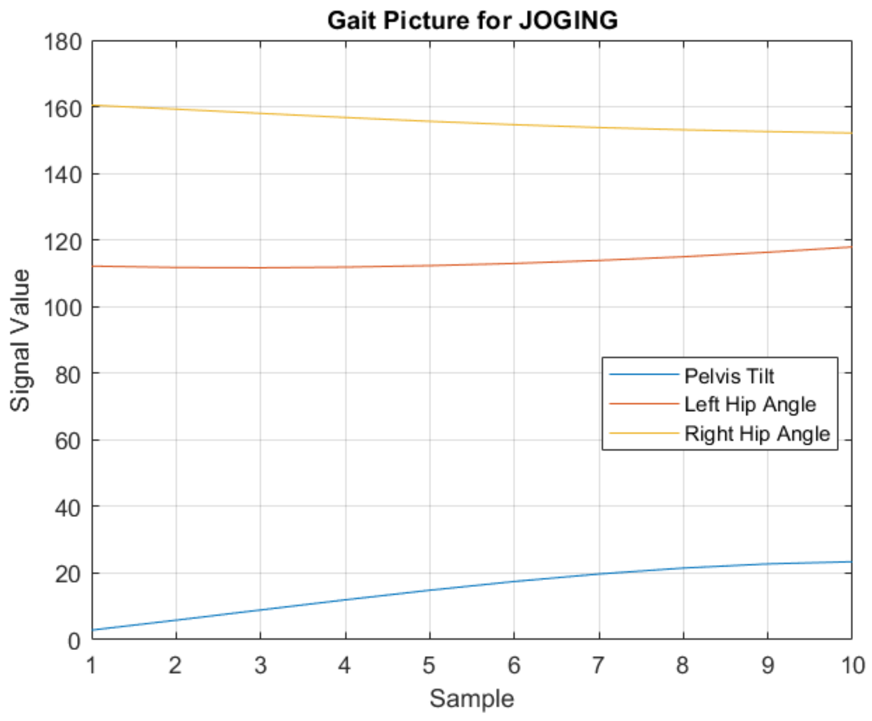


Figure B.2: Gait Picture Taken During Jogging

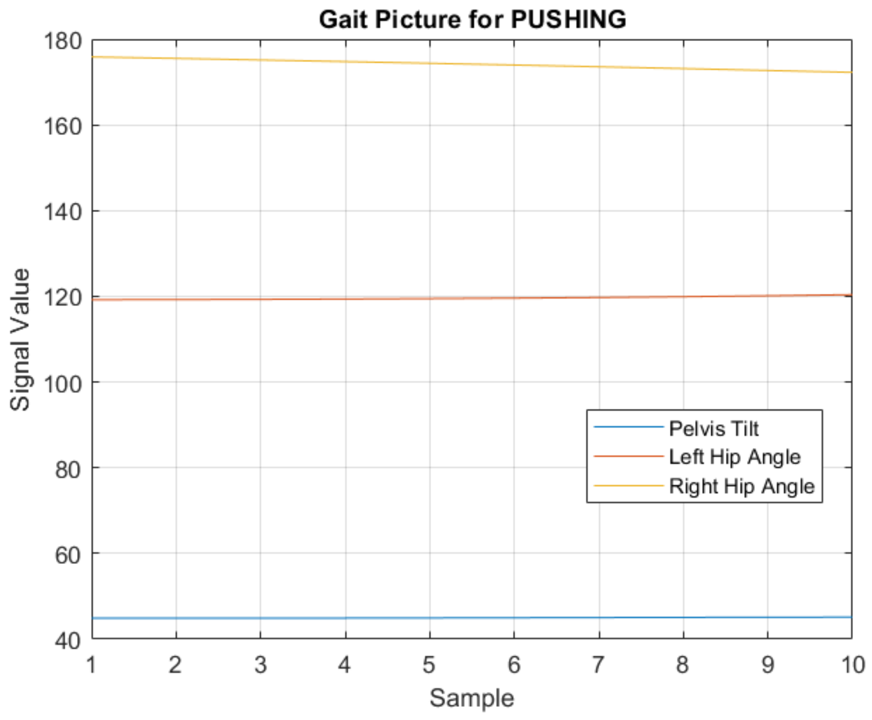


Figure B.3: Gait Picture Taken During Pushing an Object

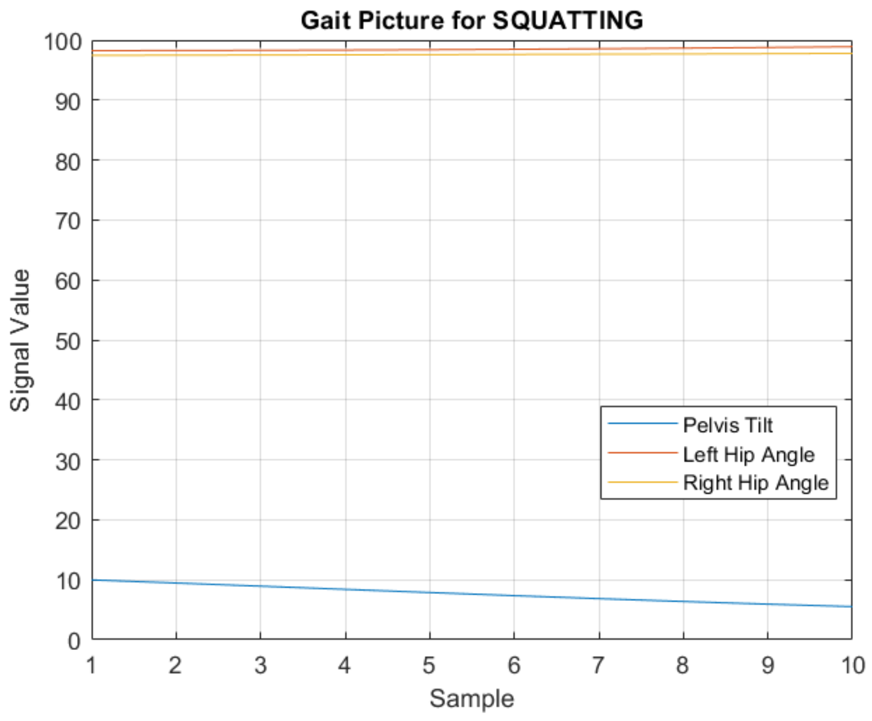


Figure B.4: Gait Picture Taken During Sqatting

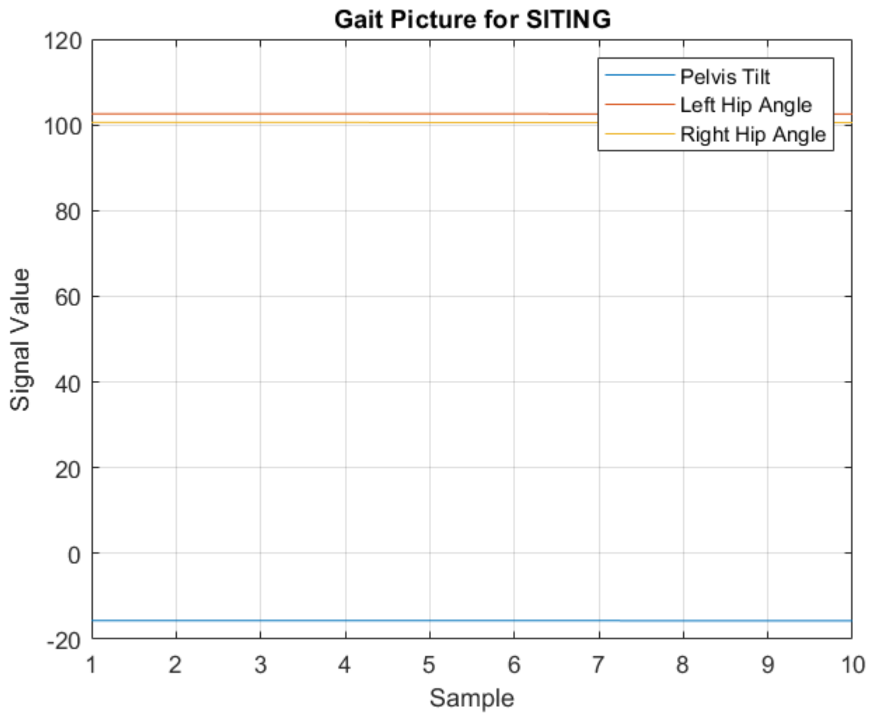


Figure B.5: Gait Picture Taken During Sitting

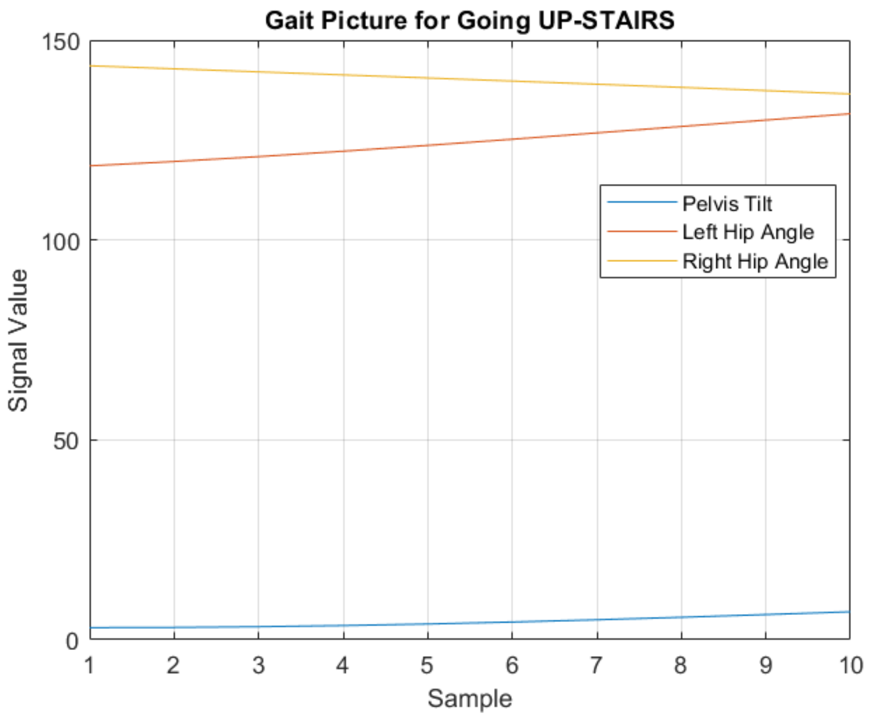


Figure B.6: Gait Picture Taken During Going Up Stairs

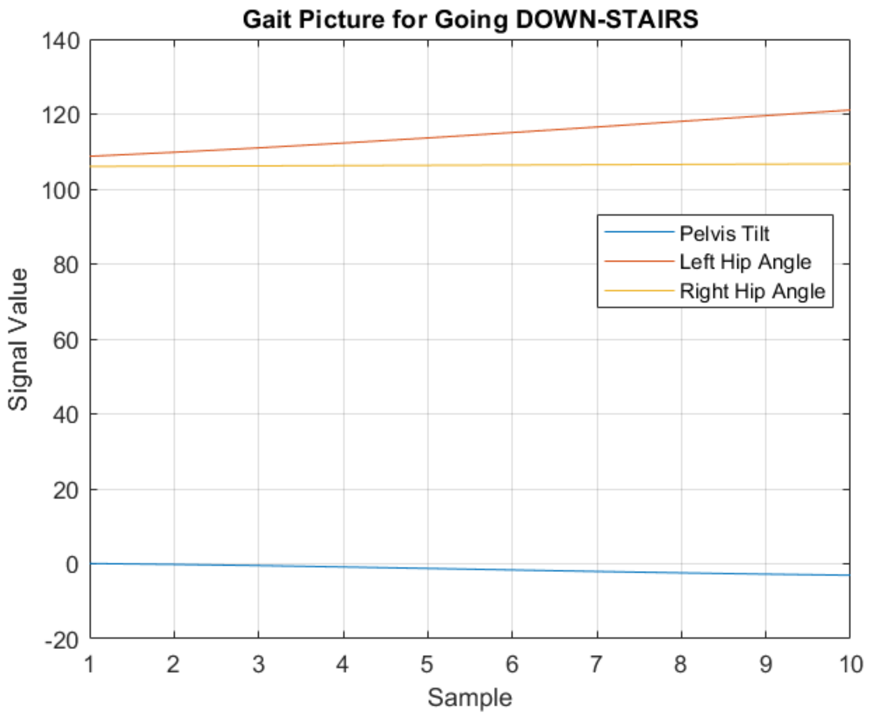


Figure B.7: Gait Picture Taken During Going Down Stairs

B.3 Algorithm Results Using Differently Sized Gait Pictures

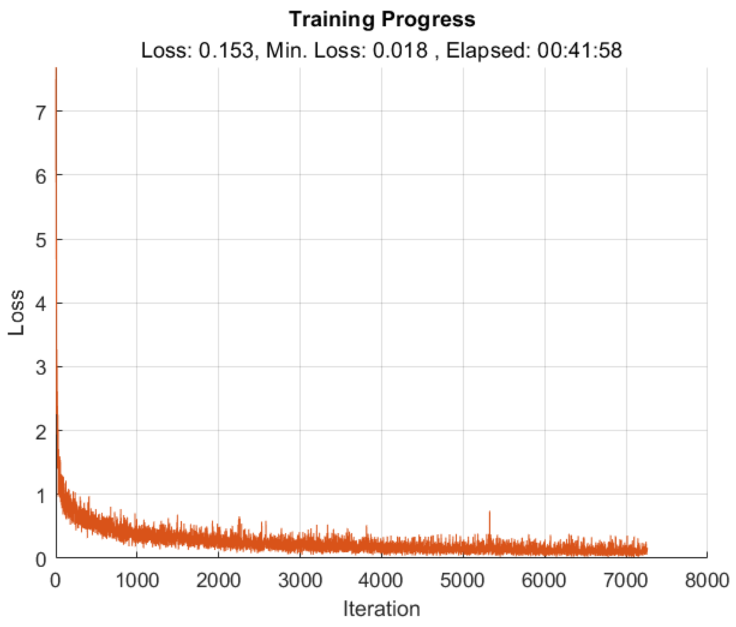


Figure B.8: Training Using 5-Sample Gait Pictures

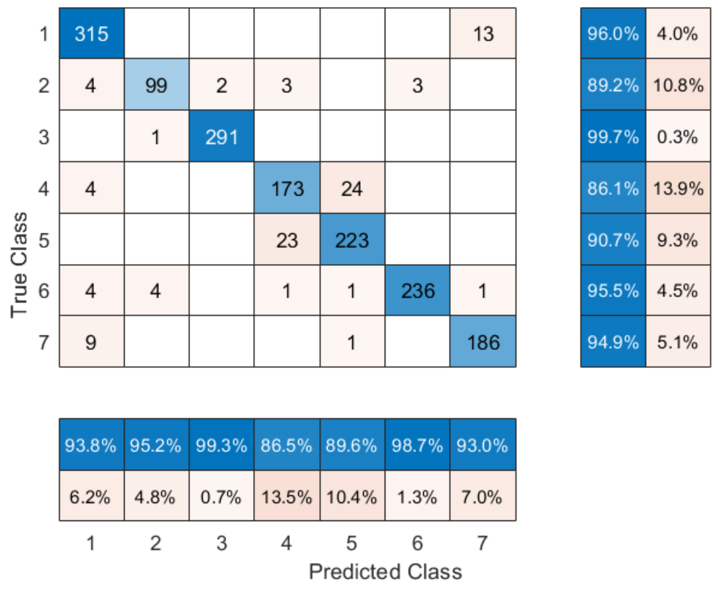


Figure B.9: Confusion Matrix Using 5-Sample Gait Pictures. Overall Accuracy: 94%, Desktop Computer Execution Time: 131ms

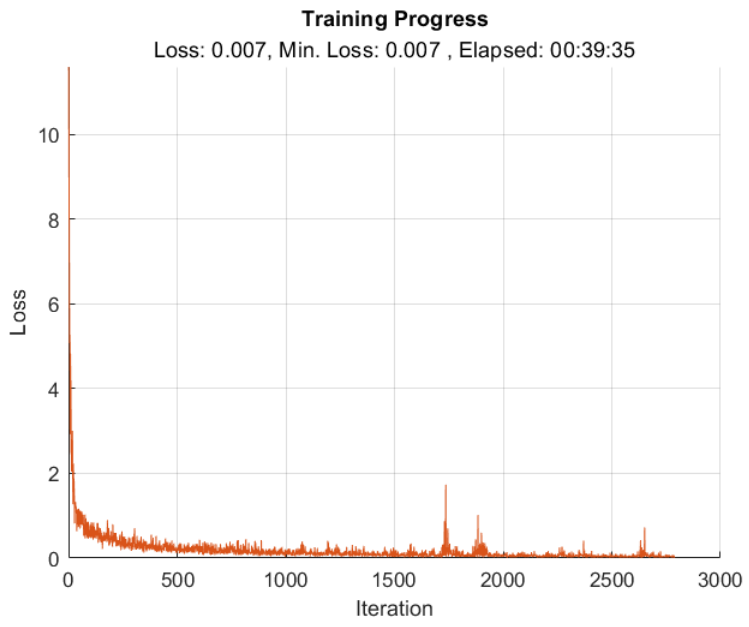
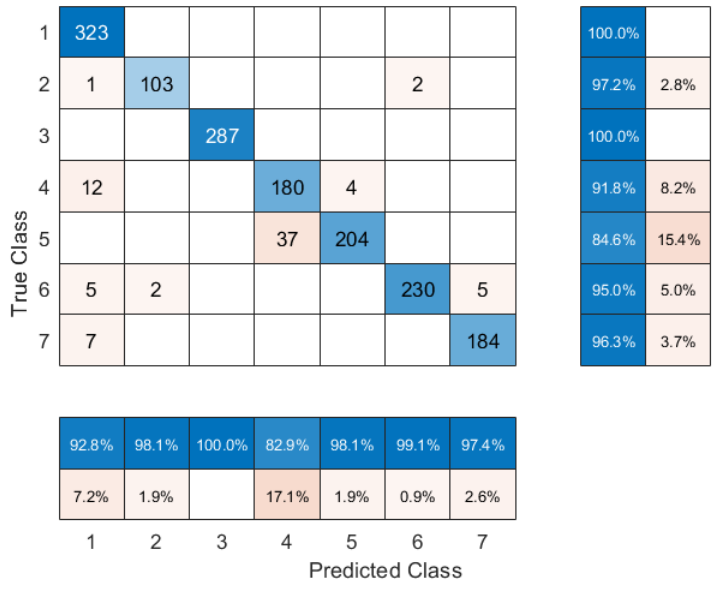
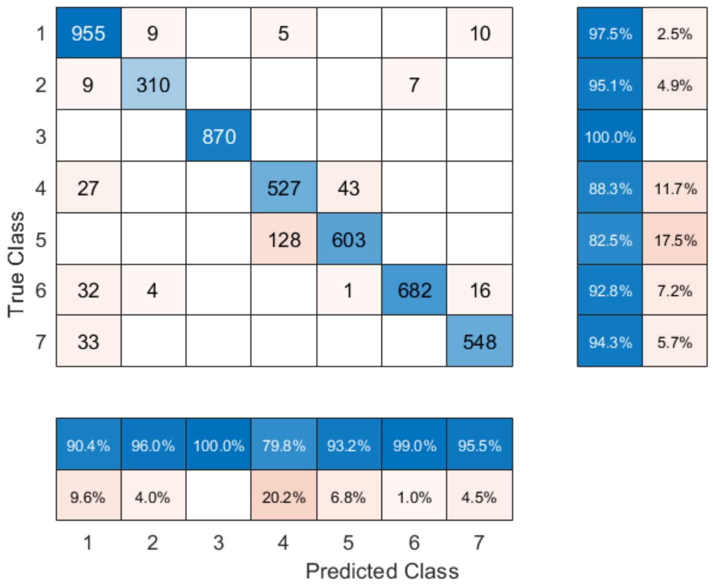


Figure B.10: Training Using 20-Sample Gait Pictures



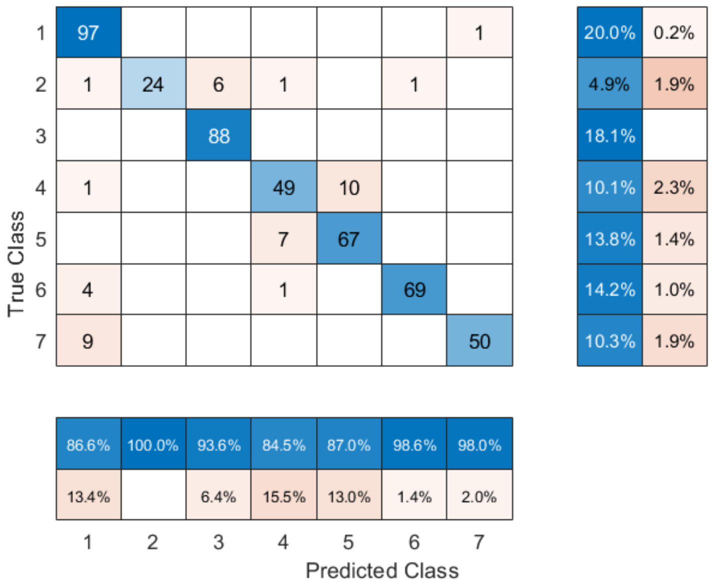
True Class \ Predicted Class	1	2	3	4	5	6	7		
1	323							100.0%	
2	1	103				2		97.2%	2.8%
3			287					100.0%	
4	12			180	4			91.8%	8.2%
5				37	204			84.6%	15.4%
6	5	2				230	5	95.0%	5.0%
7	7						184	96.3%	3.7%
	92.8%	98.1%	100.0%	82.9%	98.1%	99.1%	97.4%		
	7.2%	1.9%		17.1%	1.9%	0.9%	2.6%		

Figure B.11: Confusion Matrix Using 20-Sample Gait Pictures. Overall Accuracy: 95.3%, Desktop Computer Execution Time: 161ms



True Class	1	2	3	4	5	6	7		
1	955	9		5			10	97.5%	2.5%
2	9	310				7		95.1%	4.9%
3			870					100.0%	
4	27			527	43			88.3%	11.7%
5				128	603			82.5%	17.5%
6	32	4			1	682	16	92.8%	7.2%
7	33						548	94.3%	5.7%
	90.4%	96.0%	100.0%	79.8%	93.2%	99.0%	95.5%		
	9.6%	4.0%		20.2%	6.8%	1.0%	4.5%		
	Predicted Class								

Figure B.12: Confusion Matrix Using 10-Sample Gait Pictures With a Stride of 1. Overall Accuracy: 93.3%. Training Time Exceeded 50 Minutes.



True Class	1	2	3	4	5	6	7		
1	97						1	20.0%	0.2%
2	1	24	6	1		1		4.9%	1.9%
3			88					18.1%	
4	1			49	10			10.1%	2.3%
5				7	67			13.8%	1.4%
6	4			1		69		14.2%	1.0%
7	9						50	10.3%	1.9%
	86.6%	100.0%	93.6%	84.5%	87.0%	98.6%	98.0%		
	13.4%		6.4%	15.5%	13.0%	1.4%	2.0%		
	Predicted Class								

Figure B.13: Confusion Matrix Using 10-Sample Gait Pictures With a Stride of 10, i.e. No Samples Are Reused. Overall Accuracy: 91.4%.

B.5 Algorithm Results Using Extended Gait Pictures

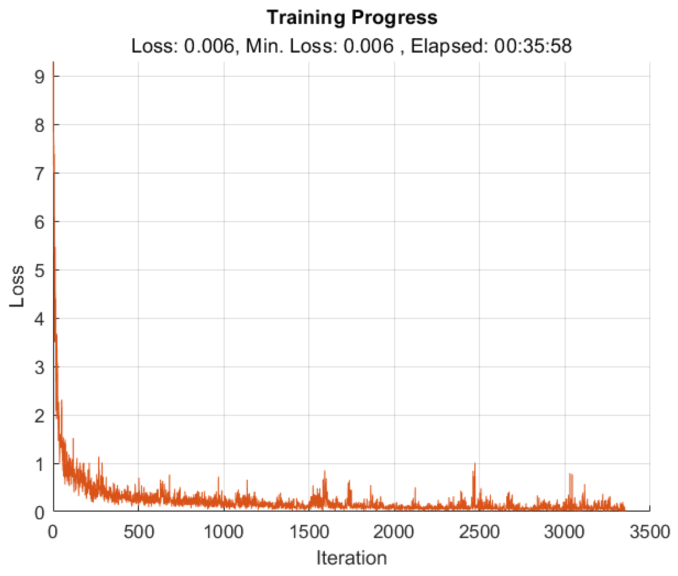
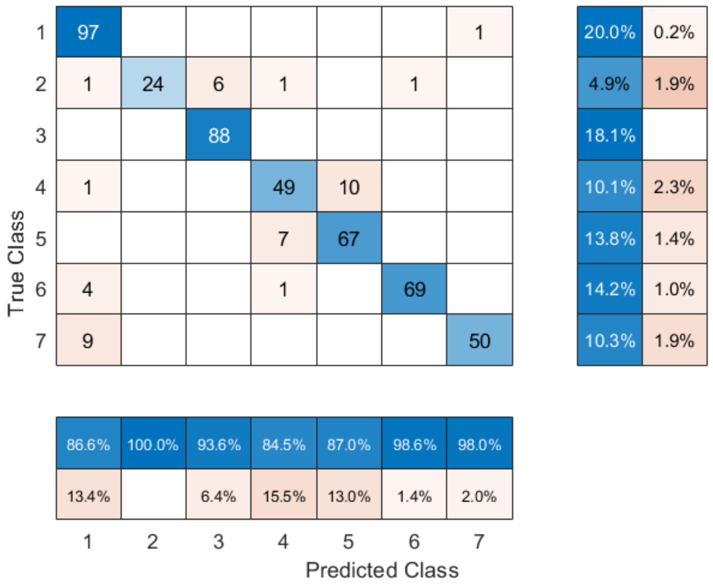


Figure B.14: Training Using 10-Sample Gait Pictures With a Stride of 3 and 5 Input Signals: Pelvis Tilt, Left & Right Hip Angle, Left & Right Hip Angular Velocity



True Class	1	2	3	4	5	6	7		
1	97						1	20.0%	0.2%
2	1	24	6	1		1		4.9%	1.9%
3			88					18.1%	
4	1			49	10			10.1%	2.3%
5				7	67			13.8%	1.4%
6	4			1		69		14.2%	1.0%
7	9						50	10.3%	1.9%
	86.6%	100.0%	93.6%	84.5%	87.0%	98.6%	98.0%		
	13.4%		6.4%	15.5%	13.0%	1.4%	2.0%		

Figure B.15: Confusion Matrix Using 10-Sample Gait Pictures With a Stride of 3 and 5 Input Signals: Pelvis Tilt, Left & Right Hip Angle, Left & Right Hip Angular Velocity. Overall Accuracy: 92%.