Final Report

MEAM 445/446 Senior Design

Team 01 - Team Walk This Way

RehabiLiGait

September 2017 - April 2018

Submitted April 30th, 2018

Team Members:

Ryan Draper - ryandr@seas.upenn.edu

Adnan Jafferjee - adnanjaf@seas.upenn.edu

Daniel Stockburger - dstock@seas.upenn.edu

Alex Herlihy - aherlihy@seas.upenn.edu

En Hui Zou - zoue@seas.upenn.edu

Kwame Owusu - owusua@seas.upenn.edu

Advisors:

Dr. Paul Stegall - stegall@seas.upenn.edu

Dr. Carol Wamsley - Carol.Wamsley@uphs.upenn.edu

Dr. Lauri Bishop - lb2413@cumc.columbia.edu

Abstract:
RehabiliGait is a 2 DOF unilateral (one-legged) robotic rehabilitation exoskeleton for chronic-stage stroke patients. RehabiliGait is designed to improve the gait, or walking cycle, of chronic stroke patients in physical therapy sessions. These patients often suffer from drop foot and impaired gait after the incident. Other rehabilitative exoskeletons are assistive in nature, guiding the patient through the correct gait; however, research has shown that this lack of patient engagement leads to less recovery. RehabiliGait can result in higher rates of recovery due to higher patient engagement, as it requires the patient to provide the corrected movement themselves.

RehabiliGait uses encoders at the patient's hip and knee joints to track the deviations of their gait from an "ideal gait" prescribed by their physical therapist. Our device provides resistive haptic feedback to the patient via cable-driven friction brakes. This feedback allows them to correct their gait by following the path of least resistance. Our system has a response time of 300ms, which is under the human reaction time to a tactile response. A torque sensor is used to complete a closed-feedback control loop that ensures that the desired torque range (0 - 10 Nm) is applied on the patient. RehabiliGait streams live data via Bluetooth to a physical therapist's computer, allowing the physical therapist to tune the resistance using an interactive graphical user interface (GUI).

In future revisions, Team Walk this Way would explore using custom-designed friction brakes for better torque consistency and DC motor-based actuation for faster response times. The next step for RehabiliGait is to get IRB approval and test on stroke patients with physical therapist supervision. Overall, RehabiliGait succeeded as a proof-of-concept for a unilateral robotic exoskeleton that can apply haptic feedback based on error control of a patient's gait.

Table of Contents

Executive Summary

Strokes are the 10th most common cause of long term disability in the United States, and they affect roughly 800,000 individuals every year in the United States, 600,000 of which are first time stroke survivors [1 brault, 2 benjamin]. After six months, patients are considered to be in the chronic phase of recovery; however, it has been shown that recovery can continue up to at least two years post incident [3 wamsley]. Currently, over 7 million Americans live with chronic impairment due to a prior stroke [4 A. S. Go]. RehabiliGait is designed to be used by these chronic stage stroke survivors in conjunction with physical therapists in physical therapy sessions.

The main metrics which characterize impairment for these individuals in the chronic phase of recovery include slow walking speed and low walking endurance [5 wade]. Slow walking speeds seen in chronic stage stroke survivors can be primarily attributed to drop foot, the most common gait impairment. Drop foot is seen when patients are unable to raise their affected foot high enough off the ground to take a step. Patients naturally develop compensation mechanisms such as hip hiking and leg circumduction as a means to move their foot forward without fixing their underlying issue of drop foot.

Existing solutions range from physical therapists manually guiding patients through a desired motion, to more complicated and costly methods such as robotic exoskeletons. These existing gait rehabilitation methods use assistive forces to lead a patient through a desired gait trajectory, which relies on the patient passively learning the process using proprioception. Passive learning has been shown to have lower levels of patient engagement and thus lower levels of neurological stimulation compared to active motor learning [6]. Active motor learning is a process where patients correct their own deviations from a target based on real-time feedback. RehabiliGait provides patients with this feedback in haptic form using resistive torques at the knee and hip joints proportional to the patients error from a target gait. RehabiliGait also restricts the survivor's ability to use compensation mechanisms, which incentivizes them to focus on altering their gait closer to an ideal gait.

Physical therapists are able to program RehabiliGait with ideal gaits that are specific to individual patient needs. For instance, a patient whose gait features exaggerated hip hiking might be prescribed higher resistive torques at the hip joint, and lower resistive torques at the knee joint. This would lead the patient to correct their gait such that their knee joint is used more and their hip joint less prominently.

RehabiliGait aims to have a resistive torque range of 0.5 - 10 Nm, which is sufficient to influence patient motion. A response time of 100 ms was deemed sufficiently quick for applied resistance to feel natural. Our target sale price at production volume was $15,000, which is much less expensive than comparable rehabilitative exoskeletons. The weight, battery life and sound produced by the device were based on feedback from physical therapist advisors for easy integration into physical therapy sessions. Ergonomic metrics such as the time to put on, adjust, and take off the device were similarly defined. RehabiliGait has been designed to adhere to standards for wearable medical devices by the U.S. regulators and non-governmental organizations.

A variety of resistive subsystem solutions were considered, including magnetorheological dampers, magnetic particle brakes, eddy current brakes, back-drivable motors, hydraulic and cable-driven friction brakes. Ultimately, cable-driven friction brakes were chosen as the optimal resistive subsystem solution for RehabiliGait.

The overall system design is shown in **Figure 1a**. The patient's gait path is recorded using encoders coaxial with the patient's hip and knee joints. This trajectory is compared against an ideal gait path supplied by physical therapists. RehabiliGait's microcontroller is able to detect when a patient deviates from an ideal gait and sends a signal to the servo motor which actuates a cable-driven friction brake. A 4:1 gear ratio scales the torque applied on the patient to the desired values. A torque sensor completes a feedback loop which verifies the correct torque is applied on the patient. A user interface allows a physical therapist to tune the resistance, visualize the gait path, and save gait data for offline analysis. RehabiliGait is powered by a wearable nickel metal hydride battery pack that allows it to function for 4.5 hours under normal usage.

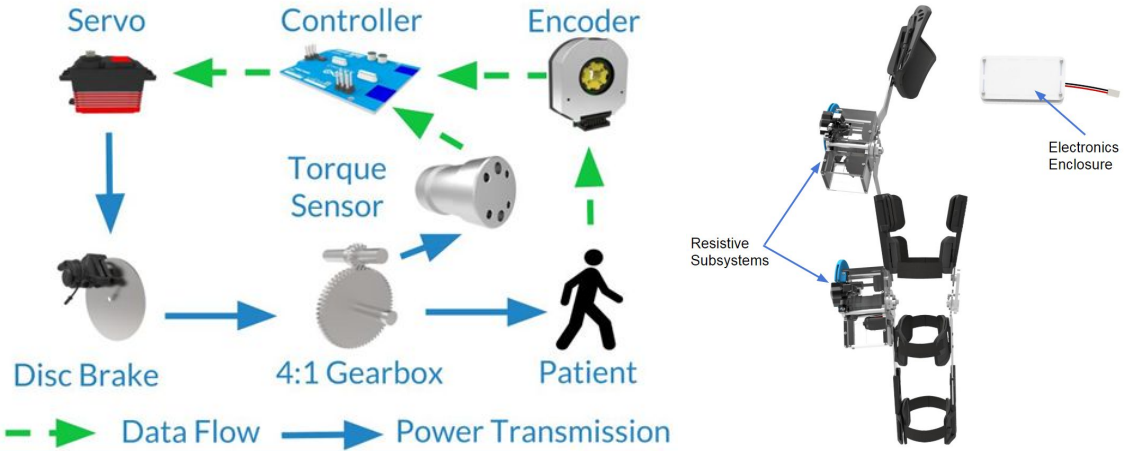


Figure 1: a) Overall System Design Organization **b)** RehabiliGait Final Form

RehabiliGait, seen in **Figure 1b**, is inherently safe because it can apply a maximum of only 10 Nm of torque, compared to the 30-50 Nm usually applied in passive learning robotic exoskeletons [7]. In addition, if the system were to shut down due to a lack of power or a malfunction, elastic springs automatically reset the friction brakes to configurations that apply no torque on the patient.

To validate RehabiliGait, we performed torque step response tests to determine the response time of our system and its ability to apply nominal torques across our target torque range. While we were able to apply nominal torques over the full 0.5 - 10 Nm range, our response time was 300 ms, which exceeds our target response time of 100 ms. **Table 1** below shows how RehabiliGait matches up to our target metrics. The system was fully validated by testing on one of the team members by comparing the user interface to videos of the actual gait.

Table 1: RehabiliGait Target Metrics vs. Actual Metrics

Parameter	Target Metric	Actual Metrics
Applied Torque Range	0.5 - 10 N*m	0.5 - 10 N*m
Response Time	100 ms	300 ms
Sale Price at Production Volume	$15,000	$10,000
Weight	5 kg	4 kg
Battery Life	4 hours	4.5 hours
Time to Put On	<7 minutes	30 seconds
Time to Take Off	<5 minutes	10 seconds
Sound Produced	65 dB	60 dB

The response time of the system was longer than anticipated due to latency of the servo motor. To reduce the response time, the spring-servo system could be replaced with a DC motor that directly actuates the friction brake. Inconsistency in the torque applied is caused by the brake rotor moving in and out of plane with respect to the cable-driven friction brake caliper jaws. Torque inconsistency could be improved through the design of a custom friction brake. Overall, RehabiliGait succeeded as a proof-of-concept for a unilateral robotic exoskeleton that can apply haptic feedback based on error control of a patient's gait.

Statement of Roles and External Contributors

Team Members

Candidates for B.S.E. in Mechanical Engineering and Applied Mechanics

Ryan Draper
Project Manager and Mechanical Analysis Engineer
ryandr@seas.upenn.edu

Adnan Jafferjee
Mechanical and Electronics Engineer
adnanjaf@seas.upenn.edu

En Hui Zou
Mechanical and Electrical Subsystems Engineer
zoue@seas.upenn.edu

Alex Herlihy
Fabrication & Validation Engineer
aherlihy@seas.upenn.edu

Daniel Stockburger
Sensing Subsystem and Controls Engineer
dstock@seas.upenn.edu

Kwame Owusu
Sensing Subsystem Engineer, UX Engineer
aowusu@seas.upenn.edu

Faculty Advisor

Dr. Paul Stegall PhD | University of Pennsylvania
stegall@seas.upenn.edu

Provided the project idea. Gave feedback on mechanical subsystem, sensing subsystem, and the controls scheme.

Technical Advisors

Dr. Lauri Bishop PhD | University of Columbia
lb2413@cumc.columbia.edu
Provided feedback on existing robotic rehabilitative solutions, customer needs and background.

Dr. Carol Wamsley PhD | Arcadia University
Carol.Wamsley@uphs.upenn.edu
Provided feedback on existing robotic rehabilitative solutions, customer needs, and background.

Daniel Harris | Teaching Assistant
dharris@seas.upenn.edu
Provided assignment-level, part-level, and overall system feedback.

Sponsors

HiTec RCD

HiTec RCD generously provided Team Walk This Way with a $600-value sponsorship of three HSB 9380TH servo motors to actuate our resistive subsystem.

Loadstar Sensors

Loadstar sensors kindly gave us a $500 sponsorship on two RS-T1 20N-m reactionary torque sensors that enabled us to have a closed-feedback look on torque applied on our device's users.

Background

Stroke Definition and Scope

A stroke is defined as death of brain cells due to a lack of oxygen. This can be caused by an arterial clot or artery rupture in the brain. Immediate side effects of a stroke can be loss of speech, general weakness, or paralysis on one side of the body. Long term side effects of strokes can include continued weakness or paralysis on one side of the body (hemiparesis), lowered cognition, and poor memory. Strokes are the 10th most common cause of long term disability in the United States, and they affect roughly 800,000 individuals every year in the United States, 600,000 of which are first time stroke survivors [1, 2]. A stroke patient is classified as acute in the 2 week period following the incident during which they are in active care. Once they are released from the hospital into inpatient care, the patients are considered to be in the subacute phase, which is where the majority of the improvement comes from. After this four to six month period, most insurance companies consider the patient's recovery to have plateaued and patients are considered to be in the chronic phase of recovery; however, it has been shown that recovery can continue up to at least two years post incident [3]. This demonstrates a clear time frame of 18 months for continued gait rehabilitation which can produce positive results in patients long after their incident. Currently, over 7 million Americans live with chronic impairment due to a prior stroke [4]. RehabiliGait is designed to be used by these chronic stage stroke survivors in conjunction with physical therapists in physical therapy sessions.

The main metrics which characterize impairment for these individuals in the chronic phase of recovery include slow walking speed and low walking endurance [5]. These impairments are demonstrated by the 6 Minute Walking Test (6MWT), which tests how far a patient can walk in a 6 minute time period. This distance is used as a risk indicator for mortality, cardiovascular disease, and mobility disability [8, 9]. A patient is considered to be healthy if they can walk 400 m in this time period; however, stroke patients with hemiparesis that can finish the 6MWT average 216 m walked [10]. Even more concerning, only 86% of chronic stage stroke survivors can finish the 6MWT due to low endurance. This demonstrates that on average, stroke survivors lack the walking speed and endurance to live a healthy, normal lifestyle. Even after 6 months of traditional physical therapy, 39% of stroke survivors have regained independent walking yet cannot walk at a normal speed [5].

Drop Foot & Compensation Mechanisms

The lack of walking speed in chronic stage stroke survivors can primarily be attributed to the occurence of drop foot, the most common gait impairment. Drop foot is the dragging of the foot due to insufficient dorsiflexion, as shown by **Figure 2**. This lack of proper dorsiflexion results in insufficient step height to take a normal step, which can cause the patient to walk slowly. Patients naturally develop compensation mechanisms as a method to move the foot forward without fixing their underlying issue of drop foot.

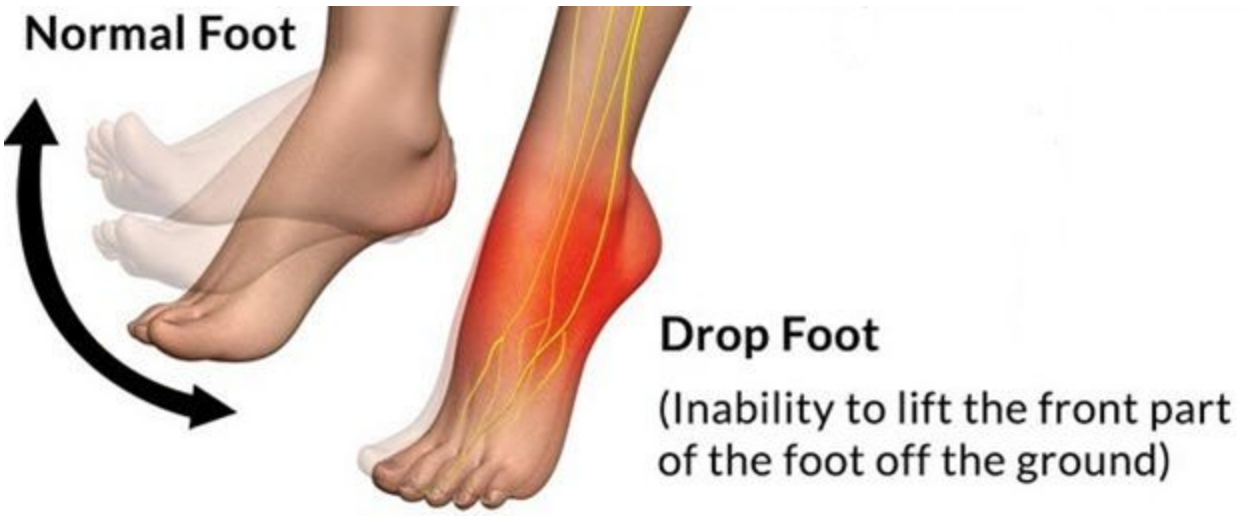


Figure 2: Drop Foot [11]

The most common compensation mechanisms include hip hiking and leg circumduction. Hip hiking, as shown by **Figure 3a**, is defined as elevation of the hip on the impaired side of the patient during the first half of the gait cycle, where the hip is normally lowered [12]. Leg circumduction, as seen in **Figure 3b**, is defined as the patient bringing the leg out of the sagittal plane (the plane parallel to the direction of walking extending from the hip joint to the ankle joint when the leg is vertical). Dr. Carol Wamsley, a physical therapist specializing in brain injuries, and Dr. Bishop, a physical therapist specializing in RAGT, have stated that preventing these compensation mechanisms has two objectives: avoiding long term health issues and reducing the drop foot in the patient. By preventing these compensation mechanisms, RehabiliGait will force the patient to focus on fixing drop foot. In this manner, RehabiliGait aims to improve the quality of walking in chronic stage stroke survivors through robotic assisted physical therapy.

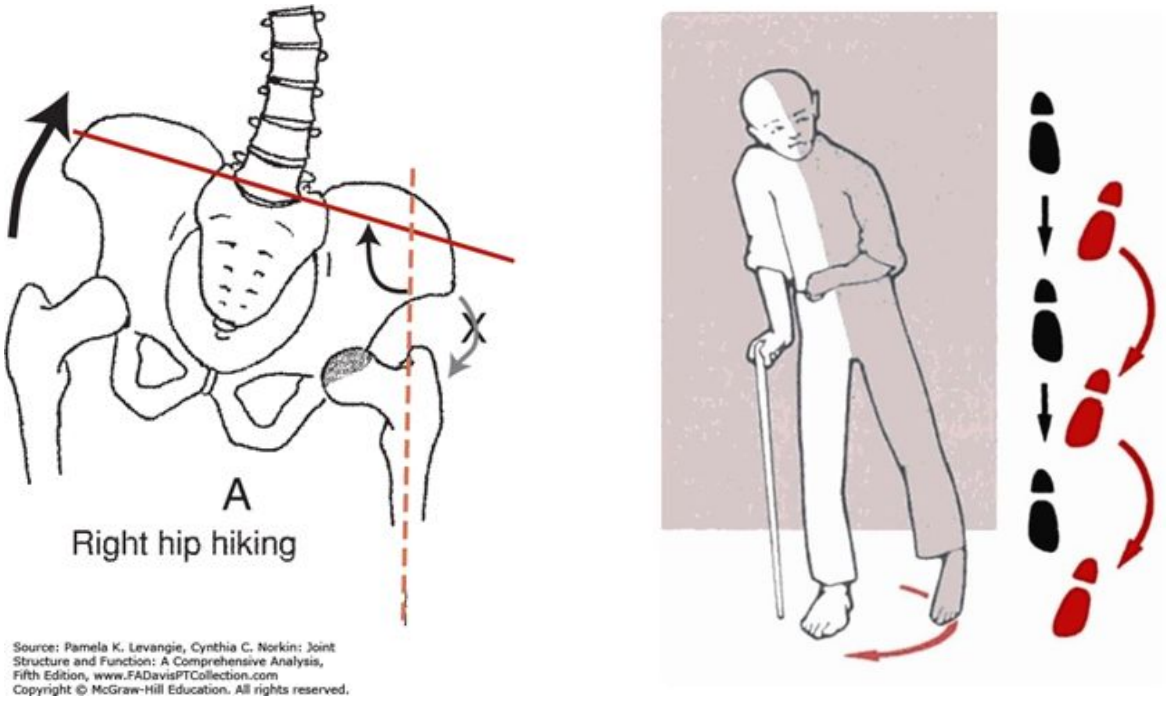


Figure 3: Gait Compensation Techniques. **a)** Hip Hiking Diagram [13] **b)** Leg Circumduction Diagram [14]

Existing Gait Rehabilitation Solutions:

Traditionally there have been three primary widely used rehabilitation methods to improve the lowered quality of walking of a chronic stage stroke patient. These methods are overground gait training (OGT), neurodevelopmental treatment (NDT), and body-weight supported treadmill training (BWSTT). In overground gait training, a physical therapist visually observes and corrects the patient's walking pattern without technological aids such as body weight support. While overground gait training is a common component of more comprehensive therapy, evidence of its efficacy is lacking. A 2009 review of clinical studies found that it had no statistically significant effect on gait function, but had minor improvements in gait velocity and endurance [15]. Alternatively, NDT has been the standard for gait rehabilitation in Europe, and remains one of the more popular rehabilitation methods [16]; however, NDT has been shown to have inferior results for gait rehabilitation compared to BWSTT [17]. Unfortunately, BWSTT has its own deficiencies. It involves body weight support of the patient above a treadmill and typically 2-3 physical therapists assisting the patient. For each impaired leg, a physical therapist must hold the patient's foot to carry and guide it through the proper motion. This process is labor intensive and physically strenuous for the physical

therapists involved. These issues motivated recent, more technologically complex solutions to gait rehabilitation.

These technologically complex solutions are separated into two main groups: neurologically stimulating (neurostim) devices and robotic rehabilitation devices. Neurostim devices have been used to treat drop foot as an additional rehabilitation mechanism alongside traditional gait therapy. Products such as the WalkAide System and the Bioness L300 Foot Drop System use functional electrical stimulation (FES) to stimulate dorsiflexion. These devices have been shown to eliminate drop foot syndrome in users whose drop foot stems from a variety of root causes, but currently require the patient to constantly wear the device. In addition, the devices are heavy, time consuming to apply everyday, must be used in tandem with normal therapy, are expensive, and are best used with sub-acute stroke survivors [18]. To treat gait impairments other than drop foot syndrome, stimulation and coordination between the hip and knee joints is necessary. While neurostim devices are ideal for users with persistent drop foot, they are an imperfect replacement to gait rehabilitation.

The other primary area of technological progress in gait rehabilitation comes from robot assisted gait training (RAGT). Current RAGT systems prescribe a gait and assist the user along the proper path, effectively eliminating the task of a physical therapist carrying the foot through the proper gait from BWSTT. Products such as Lokomat represent the robust end of the robotic rehabilitation spectrum. The Lokomat, as shown in **Figure 4a**, is an exoskeleton-treadmill system that supports the user's weight, attaches to the user's lower body, and moves the user through a proper gait. While the Lokomat has a high upfront cost, numerous comparative studies have shown that a combined RAGT, with the Lokomat or similar systems, and conventional therapy programs is superior to conventional OGT and BWSTT alone [19, 20]; however, the Lokomat has its own drawbacks. In addition to its high upfront cost and large form factor, it was designed for severely impaired neurological patients and restricts the patient along the correct path, making error detection and correction impossible [21]. Without error detection, the patient cannot interact with the system, severely limiting the Lokomat's efficacy as patient ability improves. Given these restrictions, the Lokomat is less appropriate for chronic or relatively capable subacute patients.

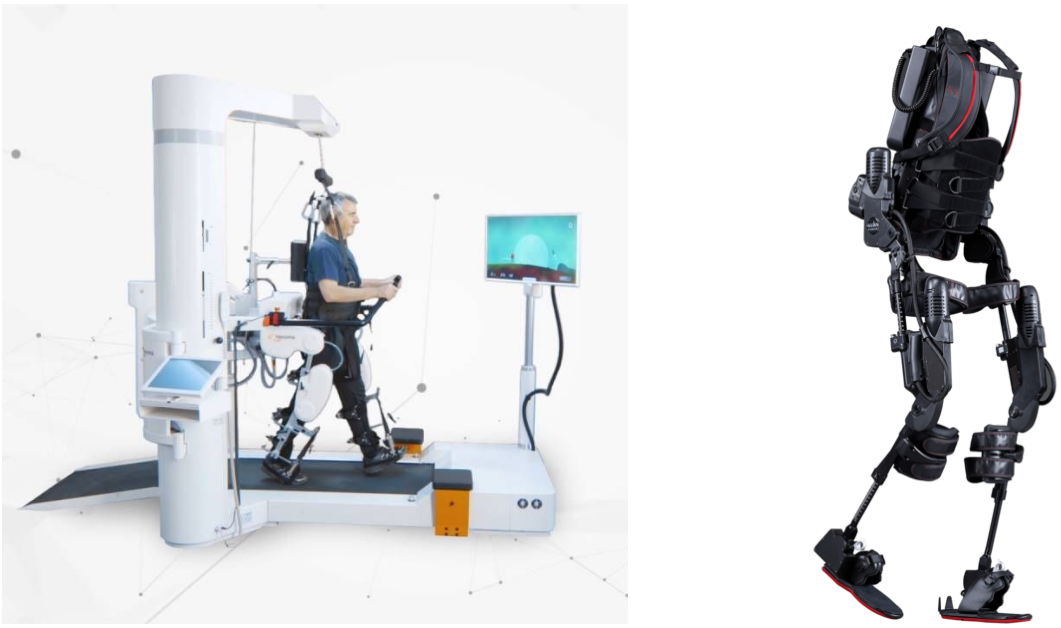


Figure 4: Existing Robotic Rehabilitation Devices. **a)** The Lokomat, a treadmill based assistive robotic gait rehabilitation system [22]. **b)** Ekso GT, a freeform assistive gait rehabilitation system [23].

Less impaired patients in the subacute and chronic phases typically use wearable exoskeletons such as the Ekso GT for robotic gait rehabilitation. The Ekso GT, shown in **Figure 4b**, works similarly to the Lokomat in that it is designed to automate BWSTT, although in this case, the patient walks freely instead of on a treadmill. Since the patient is required to support him or herself, with only minor support from a physical therapist who walks behind the user during standard practice, the Ekso GT is designed for patients who are already capable of independently walking. Due to its slimmer, less robust design, the Ekso GT costs around $120,000 [24]. Since it is an assistive rather than resistive system, it requires high power, which is provided by a large battery pack. In addition, the Ekso GT suffers from the same primary fault as the Lokomat: the user is prescribed a given path, and forced along that path. Without the ability to make the mistakes they would normally make in free movement, the patient has more difficulty learning to correct them.

Passive vs. Active Learning:

Existing gait rehabilitation exoskeletons use assistive forces to lead a patient through a desired motor trajectory. Physical therapists that Team Walk This Way have consulted believe that RehabiliGait presents an improvement over these solutions due to its incorporation of resistive forces to correct impaired gait. These forces engage the patient

in active learning by incentivizing them to correct their deviation from a target gait. RehabiliGait achieves this by making it difficult to move erroneously. Instead of being forcibly led through a motion (akin to current gait-training solutions), the patient would be incentivized to correct their own gait, which has been proven to have better motor-neural stimulation and overnight retention.

One study by Martin Lotze demonstrates increased motor-neural stimulation in patients that actively followed a desired pattern [6]. Patients who actively follow a desired pattern engage in active learning, compared to patients who are guided through a desired pattern, who undergo passive learning. Patients who engaged in active learning registered higher average motor-neural recruitments compared to patients led through the motion by a device. **Figure 5** shows a summary of TMS (transcranial magnetic stimulation) screenings of the contralateral primary motor cortex (cM1) of these patients. The cM1 is the area of the brain responsible for interfacing with the spinal cord to control movement. Patients who underwent active learning, through resistive motion, had higher rates of intracortical inhibition and facilitation, which measure one's motor-neural ability to facilitate or restrict motion, compared to patients who underwent passive learning through assistive motion. This demonstrates that active learning is more effective at neurally stimulating patients compared to passive learning, thereby leading to more effective rehabilitation sessions.

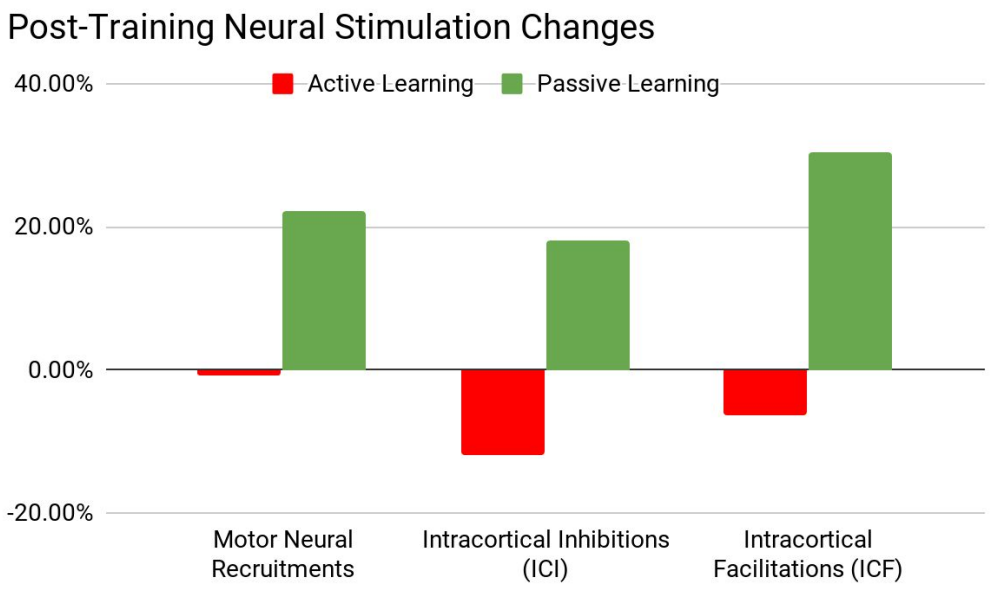


Figure 5: Assistive vs. Resistive Motion in Motor-Neural Stimulation [6]

Active learning has also been shown to be a more desirable rehabilitation technique for long term retention. One study from Northeastern University demonstrated that subjects that had to determine and correct their own errors during training were more likely to retain the information overnight [25]. Subjects that followed the active learning approach in this study showed a 73% reduction in error when following the same training path the day after a session, as compared to subjects which received explicit, supervised feedback. This implies that patients that engage in active learning are likely to experience improved session-to-session progress, and thereby progress faster to a better standard of living than with traditional gait training methods.

Another critical advantage of active learning is that it incites more patient engagement. A common pitfall of assistive motion in rehabilitation is that patients tend to reduce their active participation [26]. Moreover, active learning has been shown to provide a more rewarding treatment experience for patients, who perceive "discovering" the optimal gait by themselves as an "achievement" [27]. There is a strong relationship between this form of positive reinforcement and patient learning efficacy . Bettering patient engagement has been cited by physical therapists as a highly valuable potential area for improvement of RehabiliGait when compared to traditional rehabilitation techniques.

"A patient being engaged in the activity is by far and large one of the most impactful ingredients to successful rehab"

-Dr Lauri Bishop of Columbia University

RehabiliGait's Niche:

Compared to other devices on the market, the RehabiliGait device is most similar to the Ekso GT system. Other systems such as the LokoMat are designed for earlier stage stroke patients, often involve a hanging harness with significant assistive support, and do not offer error detection. The similarities between RehabiliGait and Ekso GT include its wearable exoskeleton design, which restricts motion to the sagittal plane, its use at physical therapy centers, and its use by less impaired (chronic stage) stroke patients who are capable of walking independently. By restricting motion to the sagittal plane, RehabiliGait prevents the patient from attempting leg circumduction, forcing them to focus on increased step height and dorsiflexion to fix drop foot. In addition to preventing leg circumduction, Ekso GT and RehabiliGait have additional compensation prevention mechanisms. RehabiliGait has a hip brace that conforms to the curvature of the patient's lower torso and straps around the waist and torso.

The main differences between RehabiliGait and the Ekso GT include its unilateral (one-legged) design and its use of resistive motion instead of assistive motion. The first difference between the RehabiliGait device and the Ekso GT is the unilateral design. Due to the lack of a second exoskeleton leg, the overall system weight will be less than Ekso GT. Weight of the system is a concern due to the lowered leg strength of chronic stage stroke patients compared to healthy adults; however, according to Dr. Bishop, chronic phase stroke patients often do not put enough weight on their impaired side. This can lead to harmful compensation mechanisms. By having the exoskeleton only on the impaired side, the patient will more evenly distribute their weight, reducing compensation mechanisms.

The other main difference between RehabiliGait and the Ekso GT is the use of resistive feedback. Using resistive feedback instead of assistive motion will reduce the required size of the battery pack, reduce the weight of the system, and reduce the cost of the system. The weight and cost of the system will be smaller due to a smaller battery pack, smaller motors, and a slimmer frame. The reduced size and weight of RehabiliGait compared to the Ekso GT and larger robotic rehabilitation devices such as the LokoMat improves the ease of setup and removal of the system from the stroke patient, which is a large problem with current robotic rehabilitation systems. High setup and removal times are a common issue in robotic rehabilitation according to Dr. Bishop and Dr. Wamsley.

Communicating with Stakeholders:

Although RehabiliGait is meant to help stroke patients improve the quality of their gait, the main stakeholder for this device is physical therapists, specifically those who work with robotic rehabilitation systems regularly. Stroke patients recovery paths vary on an individual basis and most stroke patients will not have any context on rehabilitation outside of their own. In contrast, doctors and physical therapists who have experience with robotic exoskeletons and gait rehabilitation know not only a wide variety of patients and patient recovery patterns, but also know different exoskeleton rehabilitation devices and how they interface with the recovery path. Team Walk This Way contacted and interviewed physical therapists and doctors to gain feedback on RehabiliGait. Team Walk This Way has extensively interviewed Dr. Carol Wamsley and Dr. Laurie Bishop, two physical therapists who work extensively with stroke patients and robotic solutions to rehabilitation. Dr. Wamsley is a board certified brain injury specialist and neurological clinical specialist with a doctorate in physical therapy who works as a coordinator for the Stroke Specialty Program at Penn Rehab. Dr. Wamsley

was also able to give the Walk This Way team a tour of the rehabilitation facility at Penn Rehab. Dr. Bishop is a certified physical therapist with a doctorate in physical therapy . In addition, Team Walk This Way has received feedback from Dr. Philippe Malcolm, an Assistant Professor at the Department of Biomechanics and Center for Research in Human Movement Variability at the University of Nebraska at Omaha. Team Walk This Way also reached out to stroke survivors to gain further insights on the ergonomic considerations of the project.

Objectives

Overall System Design

The desired system characteristics for RehabiliGait can be seen in **Table 2** below. These target metrics are based on findings about stroke patients and feedback from physical therapists. It is important to have defined and accurate system characteristics because stroke patients will only derive benefit from physical therapy sessions with RehabiliGait if it reaches minimum requirements for important parameters. System characteristics also form the basis for subsystem down selection and system design.

Table 2: System Characteristics

Parameter	Basic Goal	Reach Goal
Resistive Torque Range	0.5 - 10 N*m	
Reaction Time	100 ms	
Sale Price (at production volume)	$15,000	
Weight	<7 kg	
Battery Life	90 minutes	4.5 hours
Time to Put On and Adjust	<7 minutes	
Time to Take Off	<5 minutes	
Sound Produced	65 dB	

Applied Torque Range

Based on correspondence with Dr. Paul Stegall, only 10 N of force at the ankle (F_{oppose}) is necessary to disrupt a healthy individual's gait at normal walking speed. Forces above this range of magnitude may cause patients to trip and/or fall. Hence, the ideal range of resistive torque for each joint on the RehabiliGait device is between 0.5 and 10 Nm. The minimum torque is based on an estimate from Dr. Stegall for a torque that can just be felt by patient based on his research [28]. Based on the average male femur (thigh bone) and tibia (shin bone) length, 10 Nm of resistive torque at the hip and knee joint should deliver 11.62 N and 26.32 N of opposing force respectively at the ankle. This calculation assumes the leg is extended. Opposing force due to resistive torque at the hip is greater

if the leg is not extended, as pictured in **Figure 6**, as the lever arm is reduced. This means that a resistive torque range of 0-10 Nm is more than sufficient to hit the 10 N ankle force required to disrupt a healthy individuals gait. Since the users of this device can walk independently, although with gait irregularities, 10 N should be an upper-bound to the desired force applied.

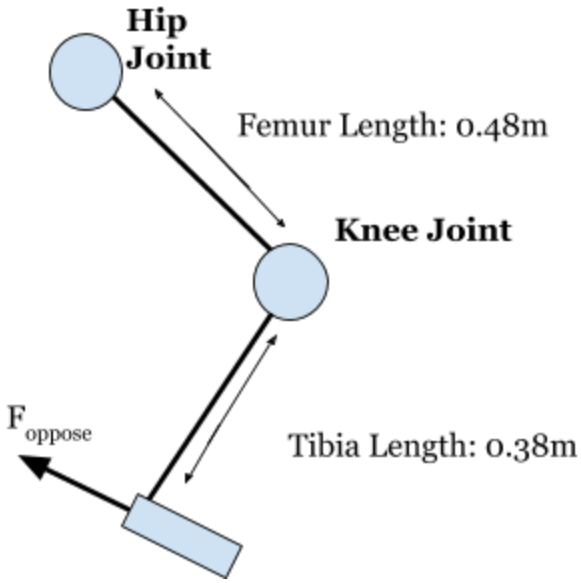


Figure 6: First Order Torque Calculation Sketch

Response Time

In order to ensure that our product can accommodate a wide array of stroke survivors, Team Walk This Way aims to provide a quick enough reaction time to be effective for average gait speeds of healthy individuals. Reaction time is defined as the time between the occurrence of the undesired gait and the actuation of the correction mechanism. If the device has a response time quick enough for healthy patients, then it will be quick enough for chronic stage stroke survivors as well. The average human tactile response time is 385 ms [29]. The total response time of the patient to an incorrect gait will be the sum of the system response time and the user's tactile response time. Therefore our system should react quickly in comparison to this response time for motion and resistance to feel natural. We estimate that 100 ms, just under one third of the tactile response time, would keep overall response time sufficiently low.

Price

One of the major motivating factors behind our project was that chronic stroke patients have the potential to make significant recovery, but have limited access to the physical therapy required to achieve these gains. Based on the professional experience of one of our technical advisors, Dr Lauri Bishop, the typical chronic stroke patient's insurance coverage allows them to have 5-6 physical therapy sessions annually after the 6 month mark following a stroke.

Medicare, the largest provider of insurance for chronic stroke patients in the US, has a annual deductible limit of $1980 on physical therapy provided [30]. This implies that each physical therapy session costs ~$360. If a physical therapy center were to buy one of our devices per year it would cost them $12.50, as demonstrated by **Eq. 1** below.

$$Cost_{Sess} = \frac{\$15000}{((5\frac{patients}{day}) \times (5\frac{days}{week}) \times (4\frac{weeks}{months}) \times (12\frac{months}{year}))} = \$12.50$$

$$(1)$$

This fixed cost represents just ~3.5% of the cost of a physical therapy session, which our physical therapist stakeholders determined to be far more feasible than existing robotic exoskeletons. Ekso GT, a comparable (albeit assistive) exoskeleton for stroke patients, costs $120,000, which represents 8x the fixed cost to physical therapy centers [24]. Our hope is that the reduced cost of our smaller and less actuated resistive exoskeleton will allow chronic stroke patients to follow physical therapy programs that better optimize their outcome.

The estimated total cost to purchase all necessary mechanical, electrical, and miscellaneous components for the construction of RehabiliGait is $1,353. A more detailed breakdown of these costs can be found in the Budget Chapter of this report.

Weight

RehabiliGait needs to weigh under 7 kg. This target weight was motivated by feedback from our physical therapist advisors. The system weight is helpful within a certain range because it helps stroke survivors to feel their impaired leg more and even up their

weight distribution across their legs. This is positive for the patient because they often avoid putting their full weight on the impaired leg, which can limit the effectiveness of recovery; however, too much weight will impede the patient as they engage in physical therapy due to both the length of the session and the distance walked. If RehabiliGait is under 7 kg, it won't impede the patient but can still provide some helpful weight to help the patient's weight distribution.

Battery Life

Upon consulting our physical therapist advisors, Team Walk This Way determined that the typical physical therapy session for chronic stroke patients lasts 60-90 minutes. This informed us that our device needed to last for a minimum of 90 minutes. Physical therapists may see up to three consecutive patients before a break, thus it would be ideal if the battery could last for 4.5 hours before needing to be recharged. The reach goal of 4.5 hours would ensure that our battery would last for physical therapists from break to break.

Ergonomics

According to physical therapists, one of the biggest issues with current gait rehabilitation systems is set-up time. Complicated systems can take up to half an hour for an inexperienced physical therapist to set-up. PTs are not paid for this time and it lessens the time they can help other patients. To be convenient for physical therapists, RehabiliGait needs to be put on and adjusted for a patient in less than 7 minutes. RehabiliGait also needs to be removed from a patient in under 5 minutes. These targets were provided by current physical therapists and doctors who regularly work with robotic gait rehabilitation devices.

For the RehabiliGait system to correctly fit stroke patients, it must have adjustable upper and lower leg lengths as well as adjustable hip circumference. To fit over 90 % of United States patients between 70 and 79, RehabiliGait must be able to fit patients between the heights of 5'4" and 6'3" [31]. This age range was chosen because the average age of a stroke victim is 74.5 [2]. In addition to height, hip and leg circumference must be accounted for in order to comfortably fit stroke survivors.

Sound Produced

RehabiliGait is designed to be used in physical therapy sessions. The friction brake produces noise which could potentially impact the physical therapist's communication

with the patient. The patient is the limiting factor as they are closer to the device in most scenarios. Therefore RehabiliGait must be quiet enough to allow the patient to hear the physical therapist at a normal conversation voice level. The closest friction brake to the patient's ear is located at the hip. The distance between this friction brake and the hip is about 2.5 feet. Thus, at a distance of 2.5 feet , RehabiliGait must be quieter than 60 dB to not impede regular conversation between the physical therapist and the patient. 60 dB is roughly equivalent to normal conversation at 3 feet away [32].

Design Impact of Standards

Most hospitals in the United States require all electrical physical medical devices to operate within safety standards issued by U.S federal regulators and non-governmental organizations (NGOs). The electrical safety organizations that will be referenced are the Food and Drug Administration (FDA), the National Fire Protection Association (NFPA), the American National Standards Institute (ANSI), International Code Council (ICC), and International Organization for Standardization (ISO).

Most hospitals in the United States require all electrical physical medical devices to operate within safety standards issued by U.S federal regulators and non-governmental organizations (NGOs). The electrical safety organizations that will be referenced are the Food and Drug Administration (FDA), the National Fire Protection Association (NFPA), the American National Standards Institute (ANSI), International Code Council (ICC), and International Organization for Standardization (ISO).

FDA's General Physical Medical Safety Standards:

The FDA provides a systematic review of how physical medical devices should function to guarantee safety for the patient according to Section 890.3450 [33]. Under the Code of Federal Regulations (CFR) Title 21, Volume 8, Section 890, there must be adequate amounts of testing and analysis over the electromagnetic compatibility and interference, electrical safety, battery performance and safety, and wireless performance [33]. To ensure that the medical device behaves properly on the patient, there must be enough supervision over the equipping and performance stages of the medical device [33]. Thus, the mechanical design should be as simple as possible to allow easy operation by physical therapists. Furthermore, there must be safeguards to prevent the patient from getting hurt due to the machinery or the environment [33]. For an explanation of how our inherent system design adheres to these standards, see the "Safety Inherent in System Design" subsection in the Safety section of "Design and Realization" of the report.

Electrical Standards Issued by Non-Governmental Organizations:

Electrical safety standards cite the leakage current allowed to flow from a physical medical device. Leakage current is the current that may unintentionally flow to the user

from the chassis of the device. The National Fire Protection Association limits battery-powered and cord-connected physical medical devices to a maximum leakage current of 100 μA . [34].

Furthemore, ANSI, specifies that physical medical devices should not operate on more than 120V and 60Hz (analogous to standard power lines). This is defined by the specifications of similar medical devices with chassis' in physical contact with patient, as seen in Table 1 on Page 5 issued by the ANSI [34]. Our patient stakeholders expressed discomfort at the idea of wearing a device that output high voltages. For this reason, RehabiliGait will operate at a max voltage of 7.8V, placing it significantly lower than the accepted standard.

Sanitation Standards:

RehabiliGait will be used in physical therapy clinics, which treat a wide variety of patients. Wearable devices used in medical environments must use antimicrobial fabrics or synthetic polymers stated in the Code of Federal Regulations (CFR) under the Medical Devices Chapter and subchapter B. The CFR defines polymers as antimicrobial if they consist Chloroprene, which is one of the main ingredients of the neoprene polymers on the knee and hip braces [35]. Antimicrobial fabrics are non-porous and have strong chemical bonds on the surface to prevent harmful bacteria and microbes from seeping into the fabric and festering. The braces integrated within RehabiliGait are made of neoprene, an antimicrobial fabric. This ensures that RehabiliGait will be easy to clean after physical therapy sessions.

Environment Interactions:

The Code of Federal Regulations specifies how a physical medical device interacts with its environment. RehabiliGait will be used in physical therapy clinics; therefore, it must be able to fit within the size of a door. The International Residential Code specifies that the minimal width of a doorway can be no less than 28 inches [36]. RehabiliGait protrudes less than 4 inches from a patient's leg; thus, a patient who is less than 24 inches wide can wear our device and move within a clinic.

Design and Realization

Resistive Subsystem Down-Selection

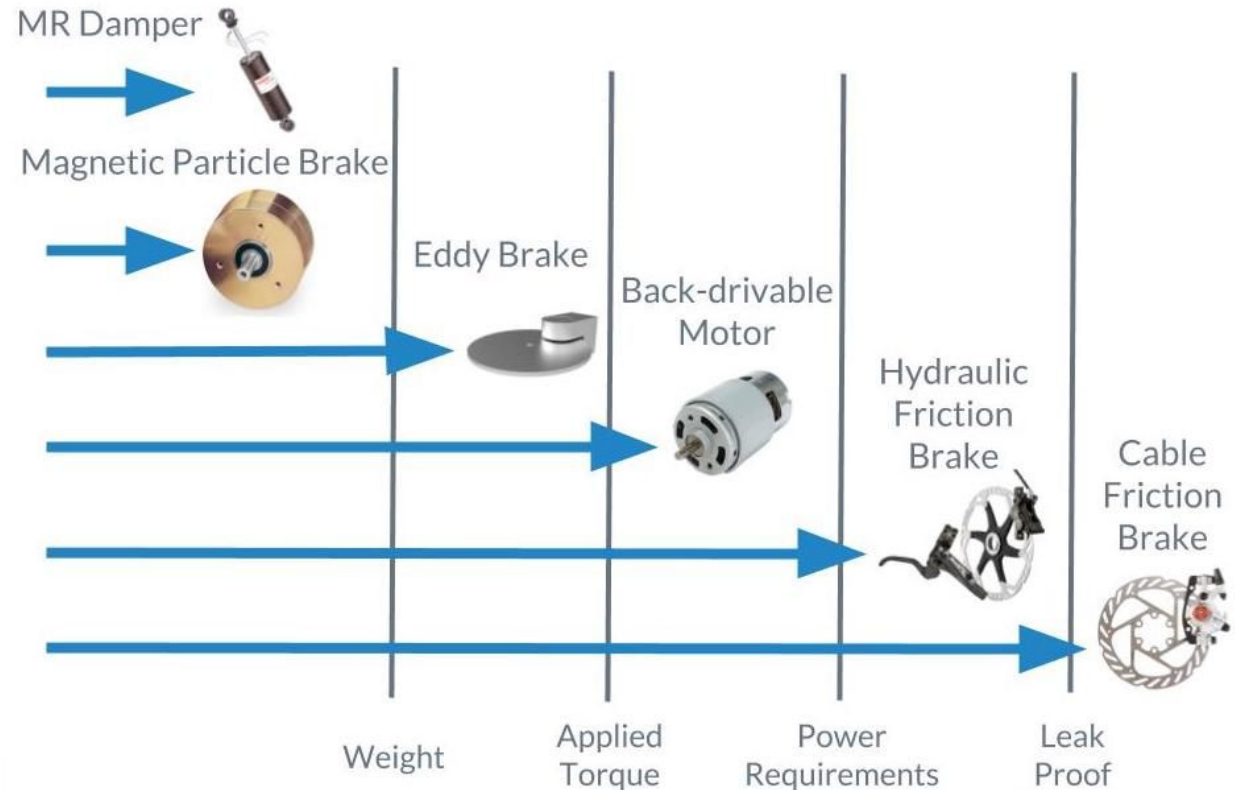


Figure 7: Resistive Subsystem Down-Selection [37] [38] [39] [40] [41]

RehabiliGait provides haptic feedback to patients via applied resistive torques that are proportional to detected errors in their gait. Upon consultation of our physical therapist advisors, we determined that these resistive torques need to be applied as continuously as possible. This is necessary in order to avoid uncomfortable jerks in the patient's haptic experience. We quantitatively compared six actuators as potential solutions in **Table 3**, and selected cable friction brakes as the optimal in a process outlined below. A qualitative summary of the process can be seen in **Figure 7**.

Table 3: Resistive Subsystem Quantitative Down-Selection

Solutions	Weight/ Joint (kg)	Applied Torque Range (Nm)	Peak Current/ Joint (A)	Peak Voltage/ Joint (V)	Actuation Time (ms)	Cost/ Joint
Magnetorheological Dampers [42] [43]	6.0	0-20	1.5	12	15	$600
Magnetic Particle Brakes [44] [45]	8.0	0-10	0.8	24	85	$700
Eddy Current Brakes	1.5	2-10	1.5	6	60	$200
Back-Drivable Motors [46]	2.0	0-10	3.0	24	5	$150
Hydraulic Friction Brakes	1.2	0-10	1.0	8	65	$50
Cable Friction Brakes	1.0	0-10	1.0	8	50	$50

Magnetorheological (MR) Dampers

Magnetorheological fluids (MRF) are commonly used in semi-active energy-dissipative applications that require variable damping. The key component of these dampers is magnetorheological oil, whose yield stress can be modified by electronically increasing the strength of the applied magnetic field on the device (as seen in the cross-section in **Figure 8**). While these dampers offered fine control over resistive torque applied they were prohibitively heavy (6kg/joint) [42] for use on our wearable system.

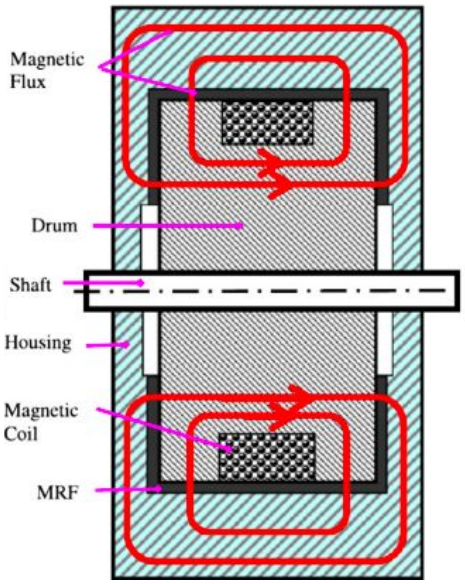


Figure 8: Cross-section of rotary MR Damper [47]

Magnetic Particle Brakes (MPB)

Similar to MR dampers, magnetic particle brakes simulate viscous damping with electronic torque magnitude-control. These brakes are comprised of a shaft separated from a housing filled with coils of copper by an air gap filled with fine magnetic powder, as seen in **Figure 9**. When current is applied through the copper coil, a magnetic field is applied, linking the shaft to the housing and producing torque.

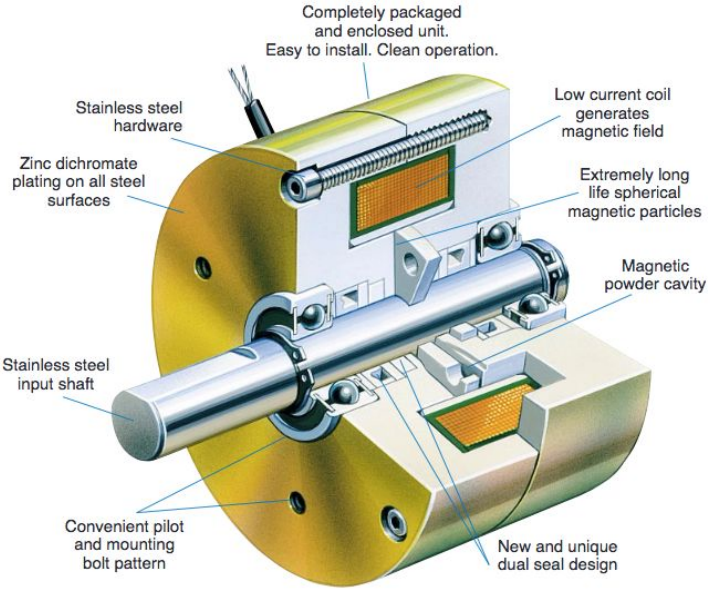


Figure 9: Cross-section of Commercially Available MPB [48]

These brakes were advantageous because they suffer minimal wear. The fine magnetic particles are less susceptible to harsh degradation, and have been rated to have cycle lives on the order of 50 million [44]. Similar to MR dampers, MPBs offer convenient and smooth electronic control over resistive torque applied. However, they are also prohibitively heavy (8kg/joint) [45], and thus, were not a viable solution.

Eddy Current Brakes

Eddy current brakes induce resistive torque by moving a conductor through a magnetic field. The magnitude of resistive torque is proportional to the relative velocity between the magnet and conductor, as can be seen by **Eq. 2** and accompanying **Figure 10** [49]**.** These brakes are advantageous in comparison to contact-based dampers (MR dampers, MPBs, friction brakes, etc.) as they have a linear relationship between angular velocity and resistive torque applied.

$$\tau_{diss} = \frac{P_d}{\omega} = \frac{\pi\sigma}{4} D^2 d B^2 R^2 \qquad (2)$$

τ_{diss} = *resistive torque*$(N * m)$
σ = *conductivity of disc* $(\Omega^{-1} * m^{-1})$
D = *diameter of magnetic core*(m)
d = *thickness of conductor disc*(m)
r = *radius of conductor disc*(m)
B = *magnetic field* (T)

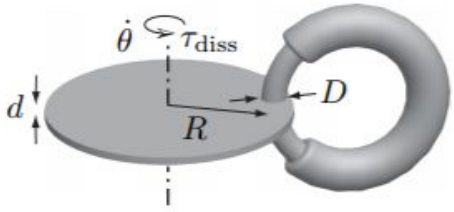


Figure 10: Eddy Brake Diagram [49]

The primary disadvantage of eddy current brakes is that they are unable to produce our desired maximum torque of 10 Nm without either prohibitively large magnets (>10" diameter) or a very high gear ratio (64:1). Not only was the latter geometrically impossible to accommodate in our system, the resulting inertial torque (5.5 Nm) was far higher than our desired minimum of 0.5 Nm. This is demonstrated in the calculations in **Appendix A2.** For this reason, they were not a viable resistive solution.

Back-Driveable Motors

One of the most commonly used actuators for applying haptic feedback in robotics is using back-drivable motors coaxial to the joints of the robot. This solution is simple to

control using motor drivers, and has the fastest response time out of all the options considered (5ms) [46]. However, the current and voltage requirements for motors capable of delivering our desired resistive torque range were 3A and 24V respectively. Physical therapists and patient stakeholders consulted expressed discomfort with using a wearable device with such high currents and voltages due to safety concerns. This eliminated this solution from our consideration.

Friction Disc Brakes

Friction disc brakes inhibit rotary motion by compressing two friction pads against a brake rotor coaxial with the rotating joint as shown in **Figure 11**.

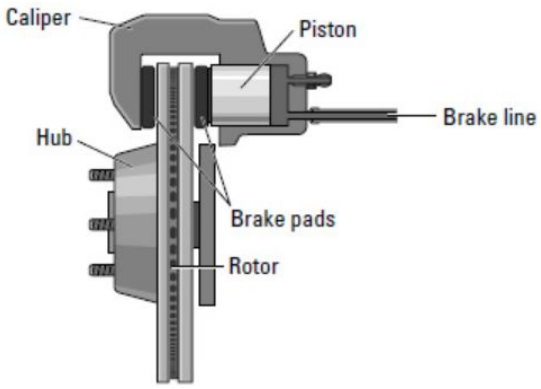


Figure 11: Cross-sectional view of a friction disc brake [50]

The resistive torque applied on the joint can be varied by controlling the compressive force applied on the rotor. **Eq. 3** and corresponding **Figure 12** [51], can be used to determine the magnitude of this resistive torque.

$$M = \frac{4}{3}\mu_k F_{load}\left(\frac{R_o^3 - R_i^3}{R_o^2 - R_i^2}\right) \quad (3)$$

M = *resistive torque* $(N * m)$
μ_k = *coefficient of kinetic friction*
F_{load} = *compressive force* (N)
R_o = *distance from rotor center to outer edge of pad* (m)
R_i = *distance from rotor center to inner edge of pad* (m)

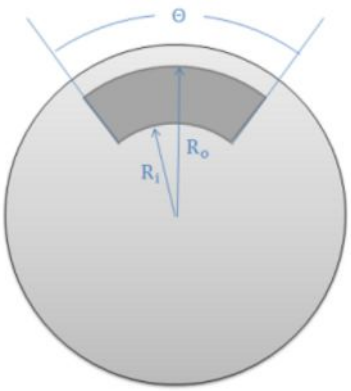


Figure 12: Disk Brake Formula Diagram [51]

The two most common means of applying braking force are using cable-driven and hydraulic systems. Cable-driven friction brakes are actuated by applying tensioning on a cable attached to a torsion spring, as seen in **Figure 13**. When the torsion spring is extended, the brake pads of the caliper close upon the brake rotor, thereby applying friction. The major advantages of this solution are that it is the lightest (1.0kg/joint) and cheapest ($50/joint). One downside to this system is that it requires maintenance as the brake pads undergo wear and need to be replaced. This is addressed in more detail in the "Wear Test" portion of our Validation.

Figure 13: Cable-driven friction brake

Hydraulic friction brakes use braking fluid to transfer pressure from a master mechanism to a slave mechanism, as seen in **Figure 14**.

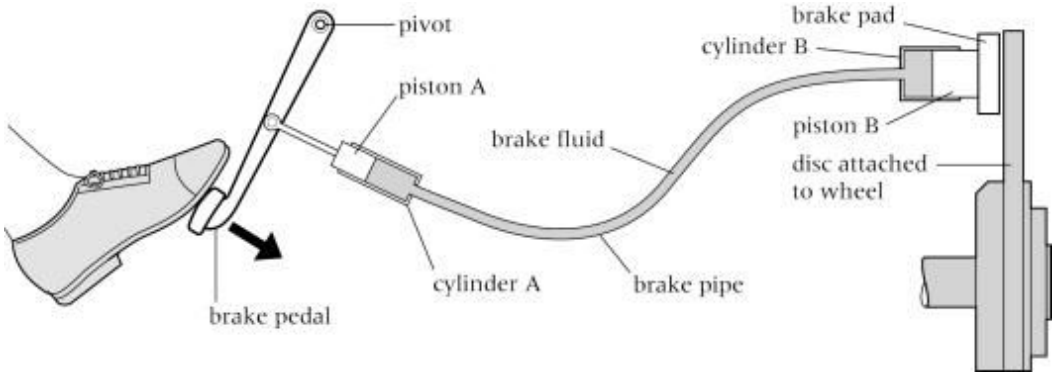


Figure 14: Transfer of pressure in a hydraulic brake [52]

These brakes offered similar benefits to cable friction brakes in terms of low weight and cost. One detriment of this solution was the risk of leakage of braking fluid. This risks damage to our device, as the braking fluid could interrupt electrical connections that run along the body of the device. Hydraulic brakes also require more maintenance than cable-driven brakes, as braking fluid has to be replaced regularly. For these reasons, we decided that cable-driven friction brakes were the optimal resistive solution for our system.

Overall System Design

RehabiliGait monitors the user's gait and applies a resistive torque on the user's knee and hip joints when the user deviates from a pre-programmed ideal gait. The flow of information and energy in the system is outlined in **Figure 15**. The patient's gait is recorded using encoders coaxial with the patient's hip and knee joints. This trajectory is compared against an ideal gait path supplied by physical therapists. RehabiliGait's microcontroller is able to detect when a patient deviates from a ideal gait and sends a signal to the servo motor which actuates a cable-driven friction brake. A 4:1 gear ratio scales the torque applied on the patient to the desired values. A torque sensor then completes a feedback loop which verifies that the correct torque is applied on the patient.

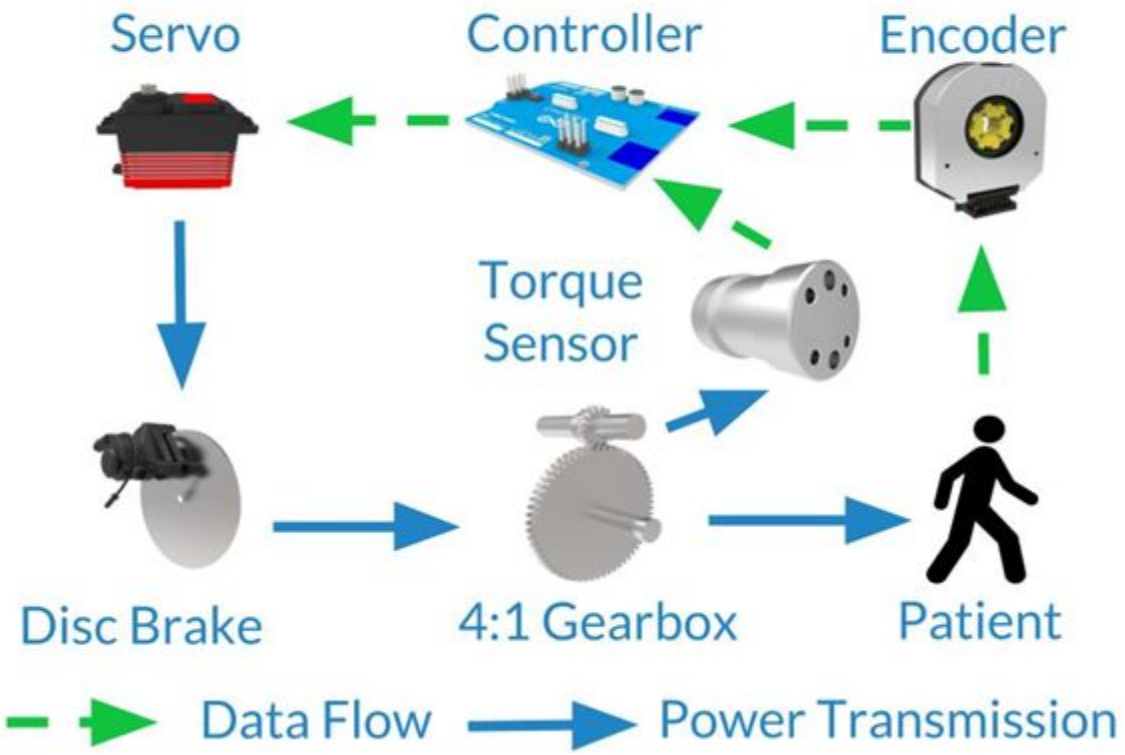


Figure 15: Overall System Design Organization

Resistive Subsystem

Bike Brake Selection

Having selected cable disc brakes as our resistive solution, we needed to downselect among available cable disc brakes on the market. Metrics we used include cost, ease of brake pad replacement, physical dimensions and weight.

Fig 16: Afterpartz bicycle cable disc brake [53]

Combining these metrics, Afterpartz bicycle cable disc brake stood out as the optimal choice (**Figure 16**). These brakes retail for $32 dollars per pair, which is lower than comparable bike brakes. Afterpartz also comes packaged with brake cables, which meant we didn't have to purchase them separately. Most importantly, the brake pads are very easy to replace. Unlike conventional brake pads that are held in place with a split pin, Afterpartz brake pads can be removed by simply disengaging a spring; however, this brake is heavier (150 grams) and larger than some competitors. Although Afterpartz brakes are slightly bigger and heavier than other leading brakes, they do not prevent us from reaching our design objectives.

Brake Rotor

The material of the disc brake rotor was quantitatively selected based on material properties such as thermal expansion, compressive strength, density, wear resistance rate and rust resistance as seen in **Table 4**. Thermal expansion refers to the rate of change of a material's spatial volume with respect to heat. Thermal expansion of the rotor was considered as it would affect the consistency with which friction could be applied on the disk. Minimizing thermal expansion was ideal as it would limit the variance of friction as the disk become warmer during periods of intense use.

The compressive strength of the rotor measures the maximum caliper clamping force that can be applied on the disk without the latter deforming. Maximizing this value was desirable to avoid compromising rotor flatness and concentricity, which determine how consistently torques can be applied. Material density was preferred to be low in order to keep the total system weight within our target metric. Wear and rust resistance were also considered in order to prolong rotor life and thereby minimize maintenance frequency. Based on these characteristics, stainless steel was chosen as the brake rotor material.

Table 4: Brake Rotor Material Properties

	Thermal Expansion (10^{-5} 1/C) [54]	Compressive Strength (GPa) [55]	Density (g/cm^3) [55]	Wear Resistance Rate [55]	Resistant to Rust
Aluminum 6061	2.4	0.171	2.7	C	Yes
High Carbon Steel	1.26	1.32	8.26	A	No
Grey Cast Iron	1.1	0.821	7.2	A	No
Stainless Steel 304	1.7	0.205	7.7	B	Yes

Servo, Spring

Servo

Servo was chosen to actuate RehabiliGait's resistive subsystem because they were easily electronically controllable and offered fast response times. We identified Hitec's HSB 9380TH servos as capable of providing at least 1 Nm of torque , which is enough to actuate our cable-driven friction brake (see **Figure 17**). The servos weight just 68 g each [56] , which is beneficial to our goal of minimizing total system weight. These servos feature a titanium gearbox, which allows them to sustain high loads while facing minimal risk of wear. Longevity was an important consideration for this selection, as we approximate that the servo will rotate ~ 5.2million cycles per year of use, as shown in **Equation 4**.

$$\#\ of\ Cycles\ = \frac{1\ cycle}{sec} \times \frac{3600\ sec}{hour} \times \frac{6\ hours\ (of\ use)}{day} \times \frac{5\ days}{week} \times \frac{48\ weeks}{year} \approx 5.2\ million \qquad (4)$$

As these servos are expensive ($200/unit), Team Walk This Way was fortunate that Hitec generously sponsored three units for our project.

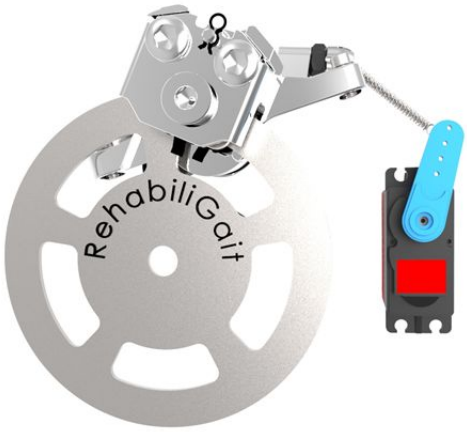


Figure 17: Servo used to regulate the braking torque of the friction brake

Spring

Team Walk This Way sought to get the highest resolution on braking torque applied as possible to make the transition of applied resistance on the patient feel gradual and "smooth." We placed a spring between the servo horn and the brake caliper to amplify the resolution of the servo by increasing the range of servo angles that correspond to resistive torques in the desired range. Based on our friction characterization, the caliper requires a tension of 2.2 lbf for the brake pads to begin applying friction on the rotor.

Similarly, the maximum braking torque applied corresponds to a spring tension of 3.7 lbf. Based on our servo's maximum arc length, we could only extend our spring by one inch. Our design also only allowed for a minimum spring length of one inch. We subsequently calculated the desired stiffness of the spring as 2 lbf/in using **Eq.5**.

$$k_s = \frac{F_{max} - F_i}{\Delta d} \tag{5}$$

The actual spring used for the design was limited by the availability of readily-orderable springs. **Table 5** shows the dimensions of the ideal and actual springs for the design. Compromises were made on minimum load as well as total displacement as finding a spring that met the exact specifications required wasn't feasible.

Table 5: Optimal versus ideal spring metrics

	Minimum Load (F_i)	**Maximum Load at displacement (F_{max})**	**Total Displacement (Δd)**	**Spring Constant (k)**
Optimal	2.0 lbf	3.7 lbf	0.7 in	2.4 lbf/in
Actual	0.5 lbf	3.7 lbf	0.25 in	13 lbf/in

Gear Train

Based on preliminary testing, the bicycle brakes can apply 3 N-m of torque. Thus, a 4:1 gear train can be used to scale the resistive torque up to the needed 0-10 N-m; however, we need to verify that disc rotor will not be spinning too fast to apply a large torque due to disc mass moment of inertia. Based on gait data [7] and proposed 4:1 gear ratio, the disc rotor will be spinning at 200 RPM. Using disc rotor physical properties in **Table 6** we can compute the resultant torque using **Eq. 6**, where m is the mass of the disc rotor in kg, r is the disc radius in meter, and α is the disc angular acceleration in rad/s².

Table 6: Disc Rotor Metrics

Disc Radius (m)	**Mass (kg)**	**Angular Acceleration**
0.05	0.10	100 rad/s²

$$T_{inertia} = \frac{1}{2} m r^2 \alpha \tag{6}$$

Using the above equation $T_{inertia}$ is 0.01 N-m, which is significantly less than the torque Rehabiligait is applying. Therefore, it should not significantly affect the torque patients experience. Using the identified 4:1 gear ratio along with the physical constraint that one of the gears needs to have a diametrical pitch greater than the diameter of the torque sensor, we selected gears listed in **Table 7**. We choose a 20° pressure angle because of its general availability and higher load carrying capacity compared to a 14.5° pressure angle.

Table 7: Gear Metrics

	Diametrical Pitch	**Number of Teeth**	**Pressure Angle (θ)**	**Face Width**	**Material**
Gear 1	63.5 mm	60	20°	4.76 mm	Stainless Steel 303
Gear 2	15.875 mm	15	20°	7.95 mm	Stainless Steel 303

Gear Force Analysis:

Gear 1 is concentric to the torque sensor and respective joint (see **Figure 18a**). Gear 2 is concentric to the disc rotor and encoder (see **Figure 18b**). Force exerted by gear 2 against gear 1 is designated as F_{21} and vice versa. Force exerted on each gear can be further broken down into radial and tangential components, which are designated using superscripts such as F_{21}^{R} and F_{21}^{T}. T_{A1} is the torque exerted by shaft A on gear 1 and T_{B2} is the torque exerted by shaft B on gear 2.

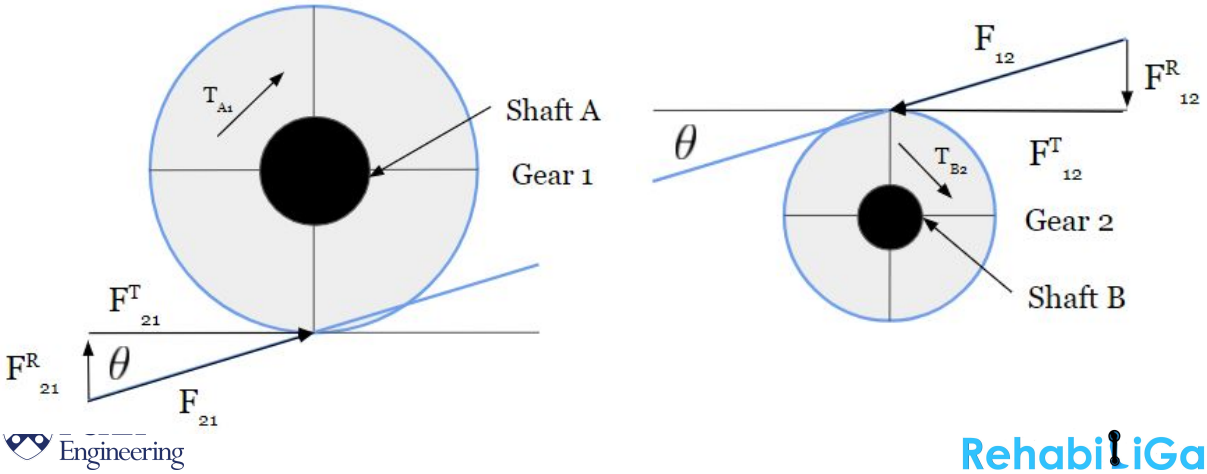


Figure 18: Gear Force Diagrams **a)** 48 Tooth Gear **b)** 12 Tooth Gear

Since the radial component of gear force does not transmit power, transmitted load (W_t) can be expressed as **Eq 7**.

$$W_t = F_{12}^T = F_{21}^T \tag{7}$$

The applied torque T and transmitted load can be related using **Eq.8**, where d is the diametrical pitch of the gear.

$$T = \frac{d}{2} W_t \tag{8}$$

Combining **Eq.7** and **Eq.8** we can use **Eq.9** to calculate the power H transmitted through the gear. w is the angular velocity of the gear.

$$H = Tw = (W_t \frac{d}{2})w \tag{9}$$

Combining **Eq.7**, **Eq.8**, and **Eq.9** we can use **Eq.10** to compute tangential gear forces. W_1 is the transmitted load in kN, d is the gear diametrical pitch in mm, H is the transmitted power in kW, and n is the angular speed in rev/min.

$$W_t = \frac{60,000H}{\pi dn} \tag{10}$$

Using **Eq.9**, the max operating angular velocity of Rehailigait (200 RPM), and gear parameters in **Table 7**, H = 0.05 kW. Using **Eq.10**, W_1=300.8 N. The resultant gear force can be calculated using **Eq.11**

$$F_{12} = F_{21} = \frac{F_{12}^T}{cos\theta} = \frac{F_{21}^T}{cos\theta} = \frac{W_t}{cos\theta} = 316N \tag{11}$$

Gear Lewis Safety Factor

Knowing the magnitude of force exerted on the gears, Team Walk This Way needed to verify that the selected gears can withstand the load. Lewis safety factor is the most commonly used metric for first order gear failure calculation. Lewis safety factor treats each gear tooth as a simple cantilever and assumes only one tooth is in contact at a time. **Eq.12** is the Lewis safety factor equation where σ is one-third the bending strength of gear material, F is the gear face width, Y is the dimensionless Lewis form factor, K_v is the speed dependent dynamic load factor, and P is the gear pitch.

$$L_{f.o.c} = \frac{\sigma F Y}{P K_v W_t} \tag{12}$$

Using **Eq.12**, both gears have safety factors of 1.5. Note that this is the minimum safety factor which is only true when Rehabiligait is applying the max torque of 10 N-m and the patient is walking quickly. Realistically, the safety factor is much higher for normal use conditions.

Gear FEA Verification

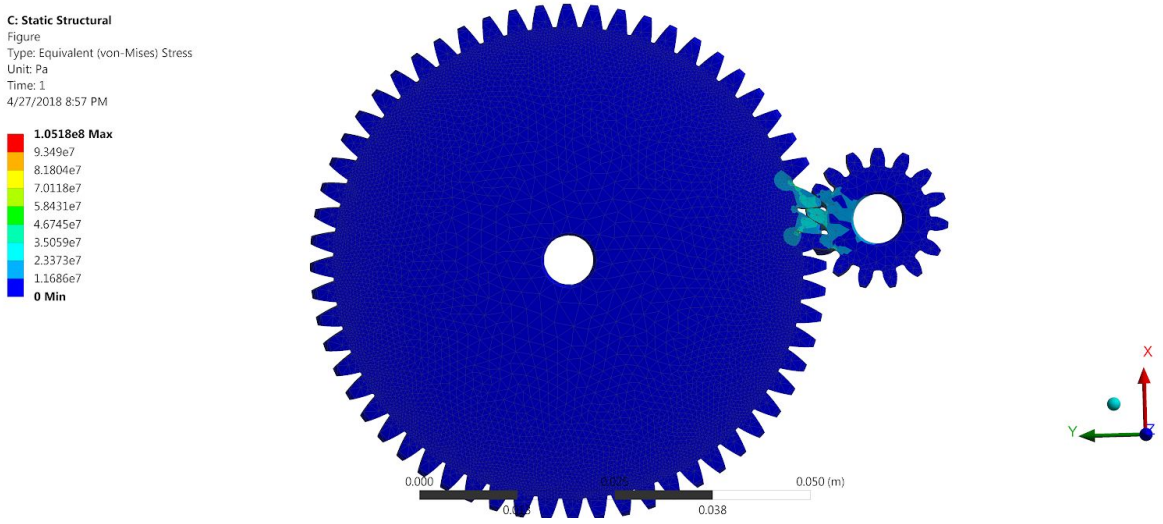


Figure 19: Gear Interaction FEA

FEA simulation indicates that max gear stress is 1 Mpa with negligible deformation (see **Figure 19**). This gives a safety factor of 6.5, which is greater than that of Lewis safety factor. This result makes sense as the Lewis safety factor does not take into account the

tooth root fillet. Both manual computation and FEA result demonstrate that the selected gear train on Rehabiligait should not fail during use.

Shaft Design

We selected a different sized shaft for each gear to reduce weight. We also chose 303 stainless steel as the shaft material due to its corrosion resistance and machinability. Referring to **Table 8**, shaft A is concentric to the 60 tooth gear and either the knee or the hip joint, while shaft B is concentric to the 15 tooth gear and the disc rotor. We need to verify that each shaft can withstand their respective maximum torque applied during use using the maximum shear stress theory.

Table 8: Shaft Metrics

	Diameter	**Maximum Torque (T)**	**Material**	**Maximum Bending Moment (M)**
Shaft A	9.525 mm	10 N-m	303 Stainless	8.32 N-m
Shaft B	6.35 mm	2.5 N-m	303 Stainless	2.76 N-m

Strength Criteria

Shafts on Rehabiligait are subject to bending and shear stress, which can be calculated using **Eq.13** and **14** respectively. Note M and T are bending moment and twisting moment applied on shafts.

$$\sigma_{bending} = \frac{32M}{\pi D^3} \tag{13}$$

$$\tau = \frac{16T}{\pi D^3} \tag{14}$$

Since shafts are subject to both bending and twisting moments, **Eq.14** cannot be used to calculate maximum shear stress directly. Based on Mohr's circle, maximum shear stress in combined loading can be calculated using **Eq.15**.

$$\tau_{max} = \sqrt{((\frac{\sigma_{bending}}{2})^2 + \tau^2)} \tag{15}$$

Substituting **Eq.13** and **14** into the above equation, we arrive at **Eq.16**, where $\sqrt{(M^2 + T^2)}$ is the equivalent twisting moment.

$$\tau_{max} = \frac{16}{\pi D^3}\sqrt{(M^2 + T^2)} \tag{16}$$

Based on **Eq.16**, maximum shear stress present in shaft A and B are 76 Mpa and 74 Mpa respectively. Using shaft material properties, the safety factor is approximately 5.5 for both shafts.

Shaft FEA Verification

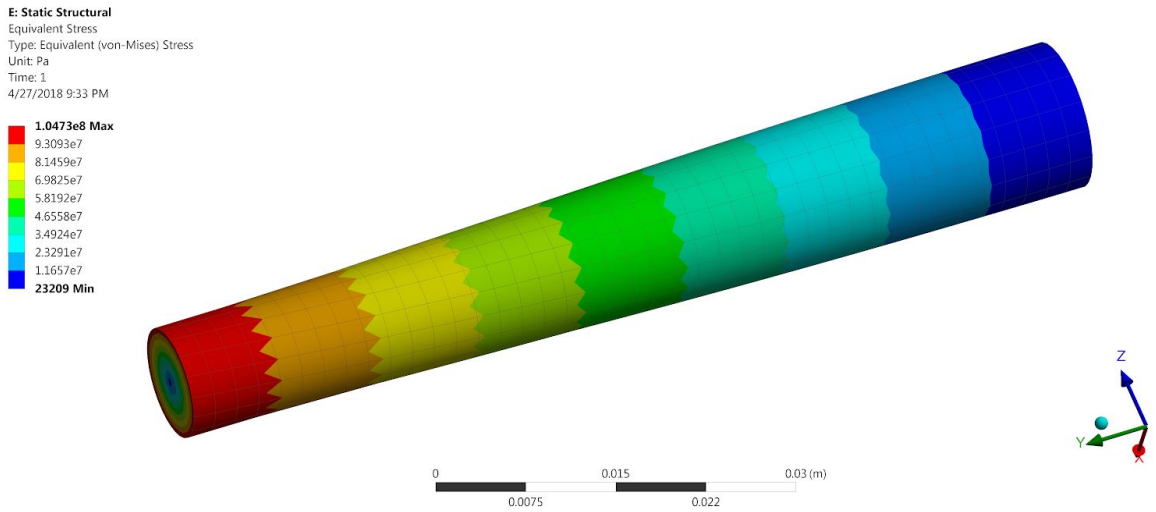


Figure 20: Shaft A stress distribution

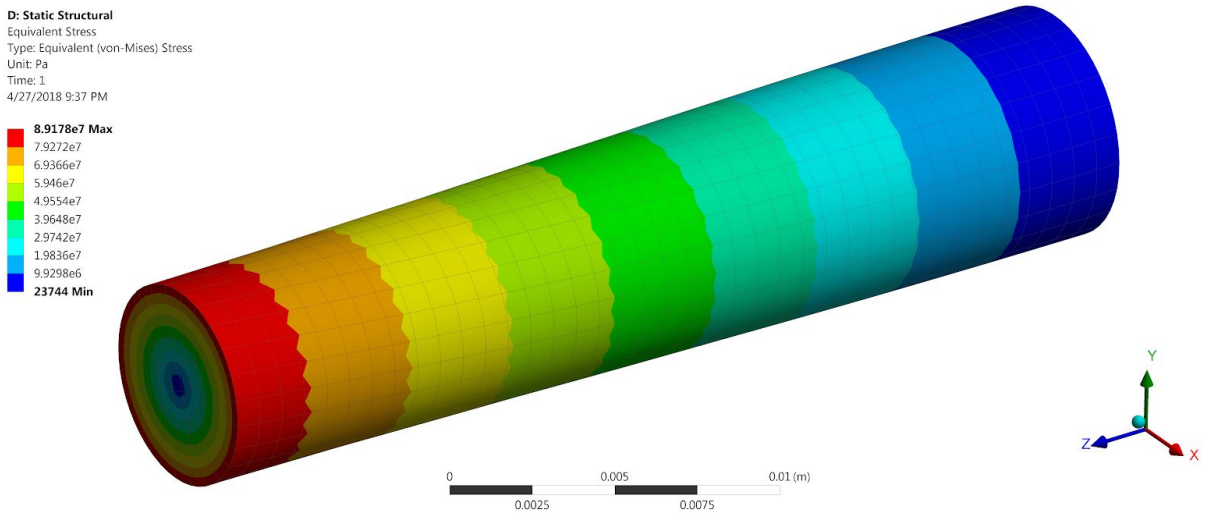


Figure 21: Shaft B stress distribution

FEA indicates safety factors of 4 and 4.6 for shafts A and B respectively (as seen in **Figures 20** and **21**). Both manual computation and FEA result demonstrate that selected shaft sizes on Rehabiligait should not fail during use.

Controls and Electronics

Encoders

To ensure that RehabiliGait's sensing subsystem was as minimalist as possible without losing its effectiveness, Team Walk This Way used two encoders; one placed on the hip and the other on the knee of the impaired leg. These two angle readings together with forward kinematic methods enabled us to determine the ankle position of the impaired leg. This ankle position was compared to a reference gait to determine deviations from an optimum (reference) gait. A resistive torque proportional to the error was then applied at the deviant joints.

Encoders were selected as our primary sensing subsystem solution because they are cost effective ($48.14), accurate (+/- 0.1 degrees), robust, and have a high data rate (1 MHz) [57]. We decided to use absolute encoders as opposed to incremental encoders because incremental encoders involve drift of the zero position. Thus, absolute encoders are easier to work with and program, making them the optimal solution.

Torque Feedback

Given the number of variables that can affect the output of a friction brake, additional assurance that RehabiliGait was outputting the correct torque was necessary. Additionally, according to our physical therapist stakeholders, being able to see the exact torque being applied to the patient would be useful information. Both of these issues are resolved with the addition of a torque sensor coaxial with the friction brake. The onboard Arduino would use the torque readings to apply PD feedback to the servo to adjust the output for servo error. The Arduino would also send these torque measurements back to the physical therapists display through Bluetooth.

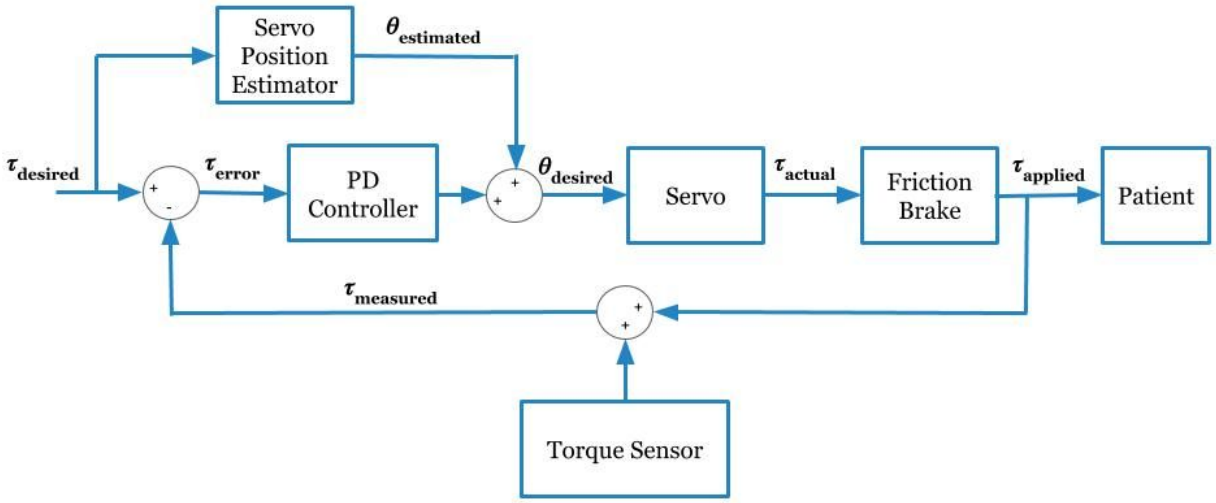


Figure 22: Controls Summary Diagram

The controls feedback loop can be seen in **Figure 22**. The torque feedback leveraged the brake characterization to provide a linear estimation for necessary servo angle to apply a desired torque. PD feedback corrected non-linearities in the system, transient effects, and the effects of unmeasured variables (temperature, velocity, humidity, wear, etc.). Integral feedback was not used as the state of the system changes quickly enough that it would never help with steady state error, but could still contribute to overshoot. Ultimately, during our torque step response validation efforts we found that a K_p and K_d of 40 µs/N and 0.0015 µs*s/N respectively were the optimal values.

The torque sensor used was the RST1-060NM [58]. This is a reactionary torque sensor can measure torques under 60 Nm. A reactionary torque sensor is sufficient since the torque sensor will only rotate the same angle range as the joint. At 120 degree variation,

as long as some slack is put in the wiring, there is little fear of the wires ever experiencing significant tension. The torque sensor was powered at 5 V, and its signal was passed through an AD711 instrumentation amplifier that amplified the signal to a range that the Arduino could read [59].

Arduino

A number of microcontrollers were reviewed as part of our down selection process. Our choice was based on the robustness of the microcontroller board, familiarity with the system, cost, extensibility (i.e. the ease of adding peripherals such as a wireless communication system), and the presence of copious amounts of documentation. Based on these requirements, the Arduino Uno was selected [60].

Code Structure

Our system uses onboard Arduino code to complete all calculations and system updates, and Matlab code on a desktop that provides a user interface to view the system as well as initiate minor adjustments to the system. The arduino and laptop interface by bluetooth using the HC-05 bluetooth module. The Arduino automatically reports system updates to the user interface approximately on an 80 ms interval. The user can send a torque application scale adjustment or recalibration command to the Arduino as desired by the user. A summarized diagram of the code structure can be found in **Figure 23** below, and the full code is attached in the **Appendix A5 & A6** and in the code folder of the report. The user interface is discussed in more detail in the User Interface section.

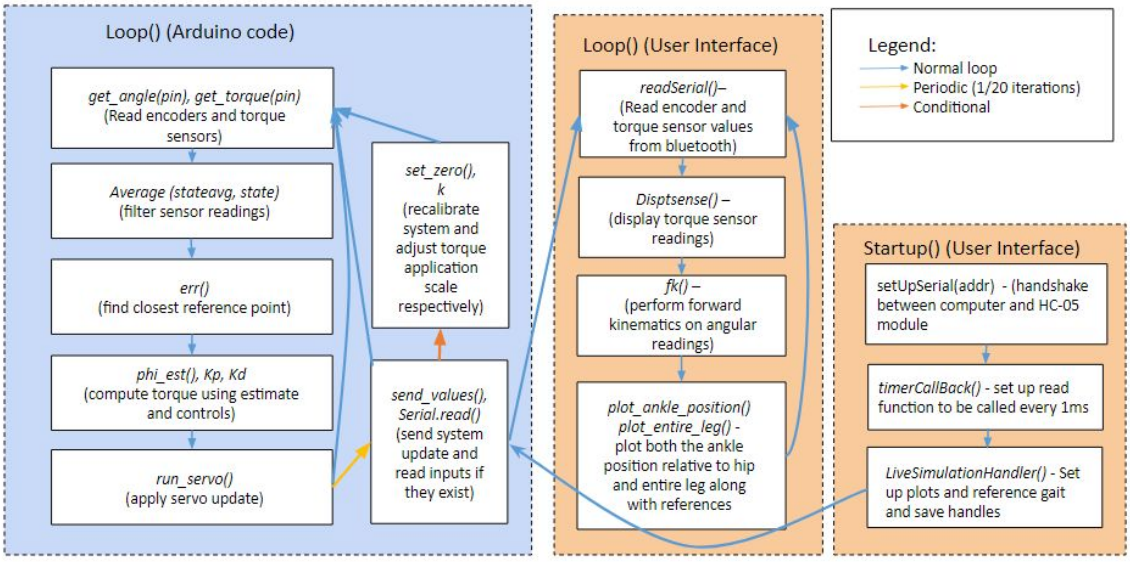


Figure 23: Code Structure Diagram

The onboard arduino runs a continuous loop that performs all operations at an approximate 250 Hz rate (4 ms period). As **Figure 23** illustrates, the loop reads sensor data, runs it through a filter, performs error correction, and then uses the feed forward model as well as controls to perform servo correction. The system outputs an update via bluetooth every 20th iteration as more frequent updates lead to buffer overflow. A simple running filter was implemented using **Eq.17** with a $\beta = 0.8$:

$$\bar{x}_i = \beta \bar{x}_{i-1} + (1 - \beta) x_{read} \tag{17}$$

The most expensive operation in the loop is the error correction. This algorithm uses a reference gait stored as discrete points in joint space, and finds the point in the reference gait closest to the current joint configuration. The current implementation is able to reduce runtime by assuming reference gait points are finer than necessary to determine the overall shape of the reference gait. Using this assumption the algorithm skips every 2nd and 3rd point until it has narrowed down to a much smaller region. The system might be further optimized by running error correction at a different frequency than system readings so that the filter could run faster than the error correction. While error correction would technically occur at a slightly lower frequency, the inputs would update more quickly, likely leading to a faster response time. Time limitations prevented the team from attempting this form of optimization.

Electronics Architecture

RehabiliGait's full electronics architecture is displayed below in **Figure 24.** The Arduino Uno microcontroller interfaces with all components, including the HSB 9380TH servos, AD711 instrumentation amplifiers, AMT-203v encoders and the HC-05 Bluetooth module. The RS-T1 torque sensors are preliminarily fed through the AD711 amplifiers, which transmit the amplified torque readings to the microcontroller. The servos and Bluetooth module are powered from 7.2V and 3.3V lines respectively. All other components are powered using a regulated 5V supply from an LM7805 regulator. A T1 2N3904 NPN transistor is used to control the encoder's read rate.

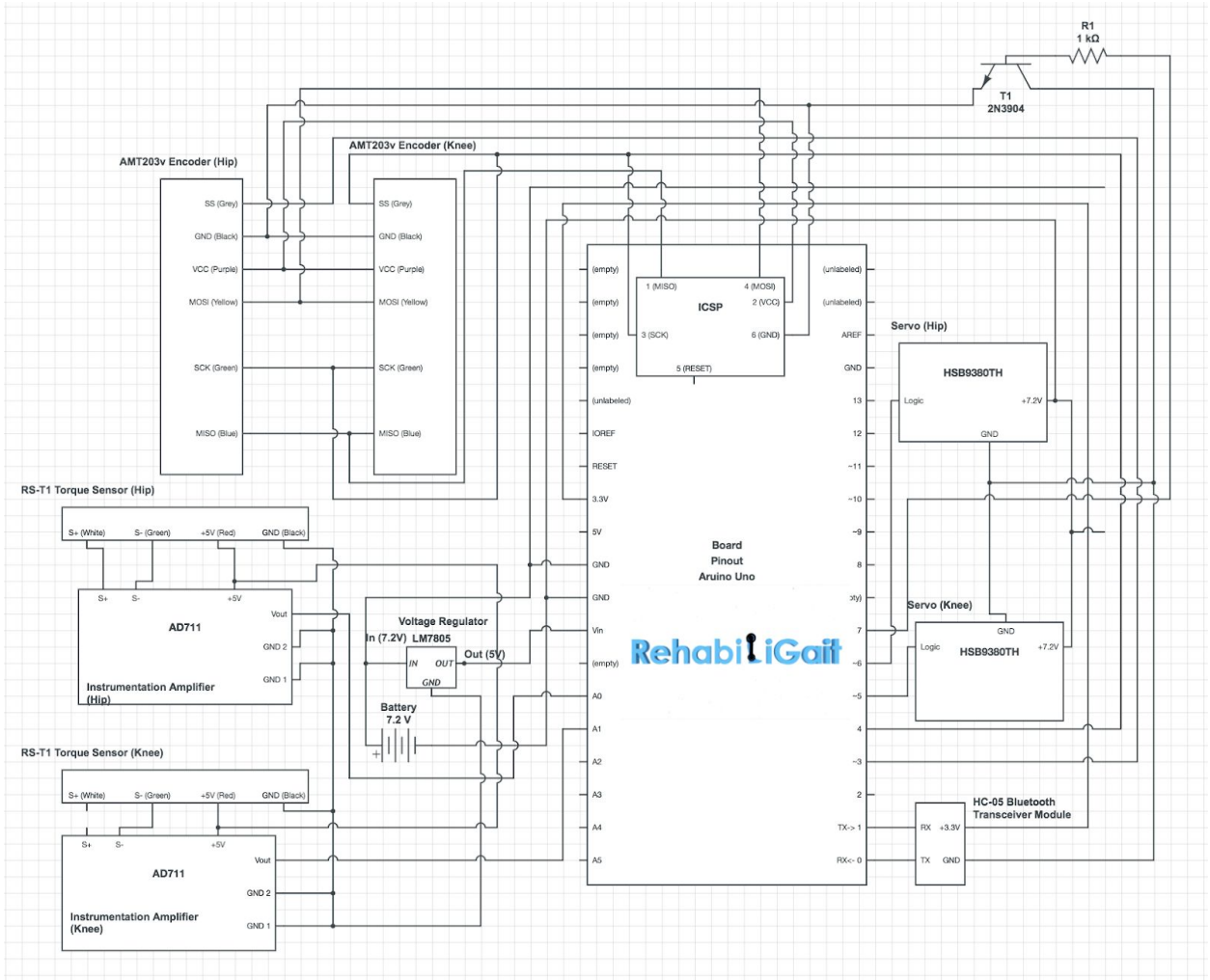


Figure 24: Full System Electronics Schematic

Power Characterization

We performed characterization of the major electronic components of our system. The results are displayed below in **Table 9**.

Table 9: Power Consumption Breakdown

Component	Operating Voltage	Peak Amp. Draw	Average Amp. Draw	Quantity
Hitec HSB9380TH *Servo Motor* [56]	7.8V	1.200A	0.375A	2
Arduino Uno *Microcontroller* [60]	5V	0.200A	0.050A	1
AD711 *Instrumentation Amplifier* [59]	5V	0.025A	0.025A	2
Loadstar RS-T1 *Torque Sensor* [58]	5V	0.014A	0.014A	2
HC-05 *Bluetooth Serial Module* [61]	3.3V	0.040A	0.025A	1
AMT-203 *Encoder* [57]	5V	0.010A	0.008A	2

Using the above characterization, we selected a battery (Tenergy 3800mAh NimH) that would allow us to exceed 4.5 hrs of continuous use at average power requirement. This allowed us to meet our battery life goal of 4 hours, roughly equivalent to the length of three physical therapy sessions. Even if the device was operating at full power continuously (highly unlikely), the device would be able to sustain a full therapy session before a battery needs to be replaced. The battery specifications can be seen in **Table 10** below.

Table 10: Battery Life Summary

	Peak Power	Average Power
Total Wattage	34.2W	6.7W
Battery Life (29.6Wh)	1.5 hrs	4.5 hrs

Electronics Enclosure

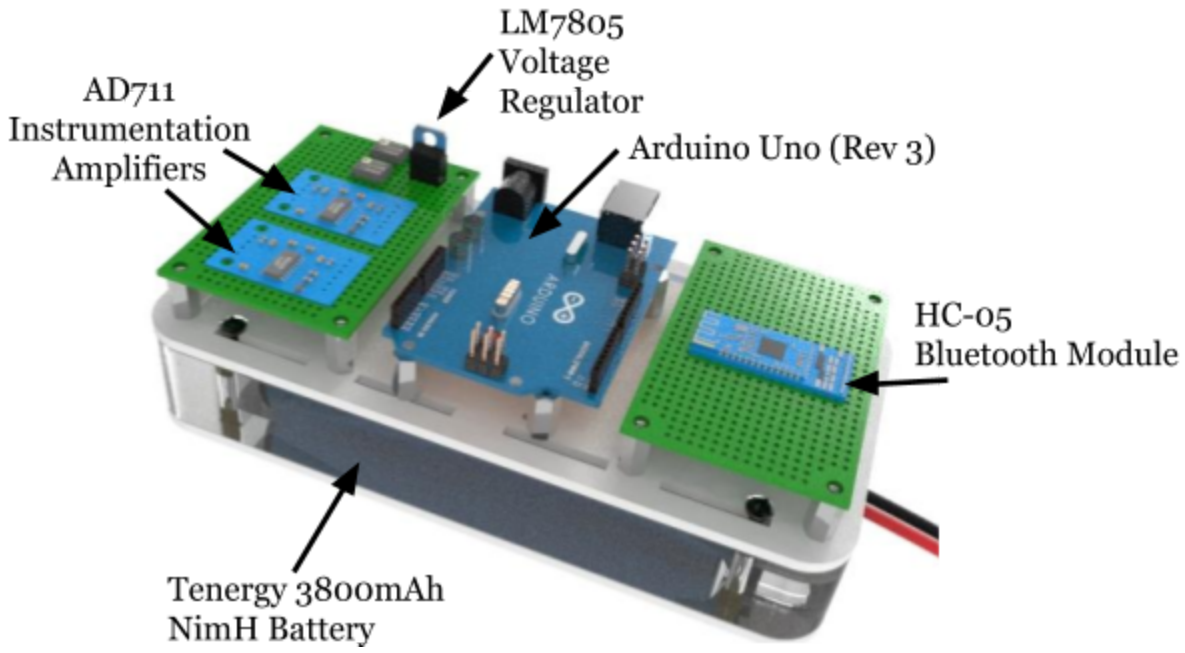


Figure 25: Electronics Enclosure Schematic (pictured without top and side covers)

We designed and fabricated a wearable enclosure to house primary electronic components such as our battery, microcontroller, instrumentation amplifiers, and bluetooth transceiver module. This enclosure can be shown without top and side covers in **Figure 25** and with top and side covers in **Figure 26**. This enclosure attaches onto the rear waistband of the hip bracing portion of our device. The enclosure was designed to allow organized wire routing to the knee and hip resistive sub-assemblies. It also allows easy access for replacing the battery and reprogramming the Arduino.

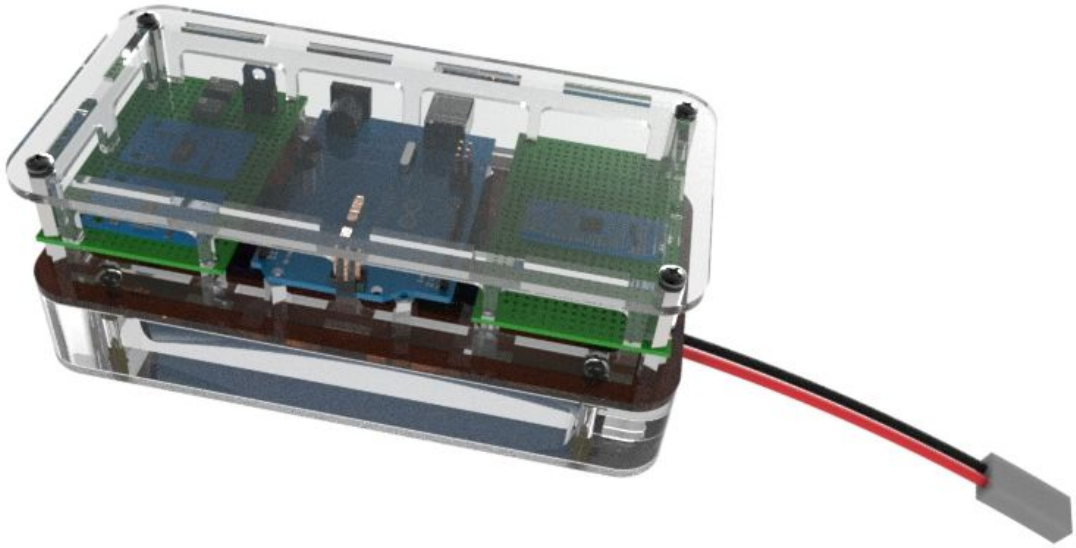


Figure 26: Full Electronics Enclosure

User Interface

A user interface is integral to our system as it provides a visual representation of our system which is paramount for intuition on how our system works. Additionally, a user interface provides physiotherapists with a user friendly way to interact with and customize RehabiliGait.

The entire Graphical User Interface (GUI) was mainly developed in MATLAB, although some Java code was added to enhance the aesthetics of the GUI. It worked independently of the arduino code, which was responsible for sensing and correcting deviations from the optimal gait; however, it did communicate with the microcontroller using a bluetooth module. In order to ensure the user friendliness of our system the GUI had the following features:

- Username and password login to protect the medical records of patients.
- Main menu where physiotherapist could review records of patients or start a new session.
- A live simulation page where the patient's gait is simulated against a reference gait and the corresponding torques and angular measurements at each joint are shown over time.
- A dead simulation page where a physiotherapist can relieve the simulation of past sessions.

Brace

Brace Selection

Team Walk This Way consulted our physical therapist advisors, Dr. Lauri Bishop and Dr. Carol Walmsley, and determined that comfort was critical to their patients complying to a new wearable rehabilitative device. We further established that the bracing portion of our device need to be light (< 3kgs) and easy to put on. We identified several existing ACL and hip-abduction braces that met these requirements, and could be easily integrated with our resistive subsystem.

Figure 27: a) T-Scope knee (left) [62] **b)** and hip (right) braces [63]

We ultimately selected the Breg T-Scope® Premier Post-OP Knee Brace and the Breg T-Scope Hip Brace (**Figure 27a** and **Figure 27b**) as they consistently performed well in customer reviews, and came recommended by faculty advisor, Dr Paul Stegall. These braces performed well in comfort-focused reviews, notably using neoprene and foam padding for skin-friendly patient contact. The braces were within our desired goal weight, as they weigh 1 kg each. The braces are easy to put on as, they have straps that can be secured using velcro attachments. Both braces also feature adjustable telescoping joints that allow patients that range from 5’ to 6’4 to be accomodated, which exceeds the 90th percentile range of heights for our chronic stage stroke patient population.

Adjustment & Customization

We have to make modifications to the purchased hip and knee braces to interface with Rehabiligait's resistive subsystem. Each joint on the brace consists of an upper and a lower segment connected by a plastic rotary joint (see **Figure 28**). The plastic rotary joint is removed and interfacing holes are drilled into each lower segment. Finally, the hip lower segment and the outer knee upper segment are connected. Refer to **Table 11** for details.

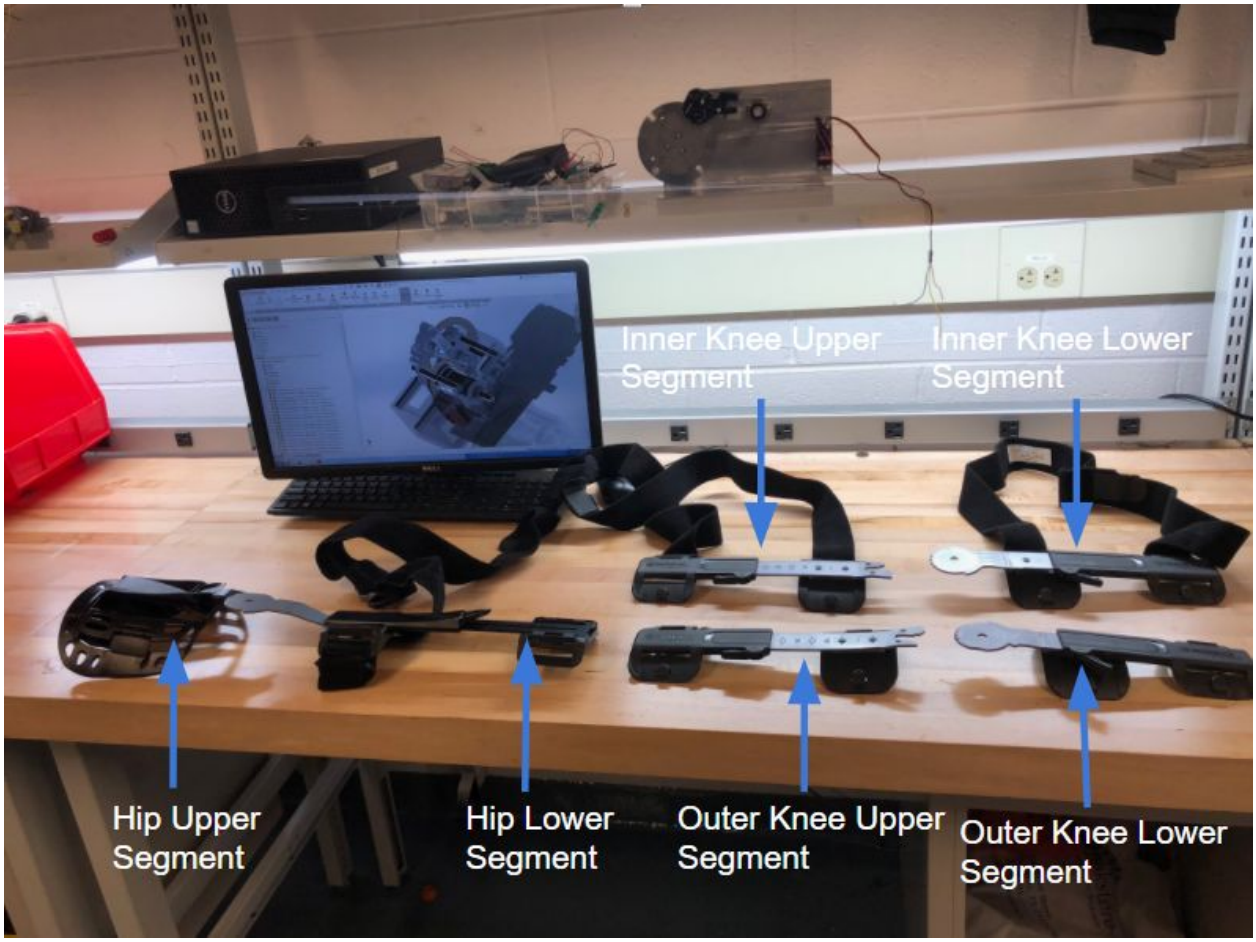
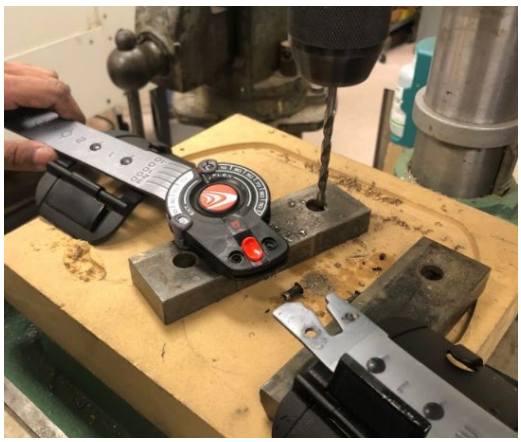
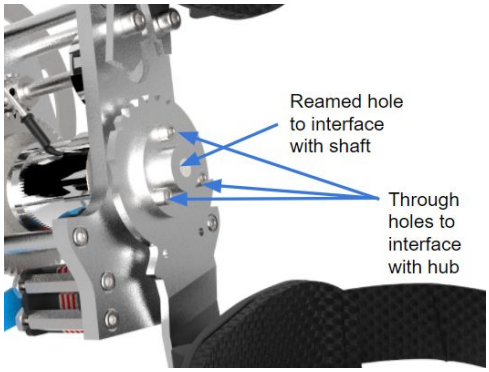


Figure 28: Hip and Knee Brace Components

Table 11: Brace Modifications

Modifications	Process
Plastic rotary joint removal	Rivets holding the rotary joint are drilled out. 
Connection hole on lower segments	Necessary interfacing holes are drilled on the mill. 
Hip lower segment and outer knee upper segment connection	Two braces are connected together using four screws and nuts. 

Safety Measures

Safety Inherent in System Design

As a resistive exoskeleton, Rehabiligait offers some considerable inherent safety advantages over its assistive exoskeleton counterparts. The friction brake is capable of applying a maximum of 12 Nm. Assistive exoskeletons of the same nature must apply enough torque to carry the patient through their gait. As existing data on a typical gait shows, this can require 30 Nm at the knee, and 50 Nm at the hip [7]. The maximum torque occurs when the servo is at its maximum length and the spring is fully stretched. If the system suddenly shuts down due to a lack of power or a malfunction, the spring automatically pulls the servo back to zero applied torque.

Acrylic Casings & Disc Covers

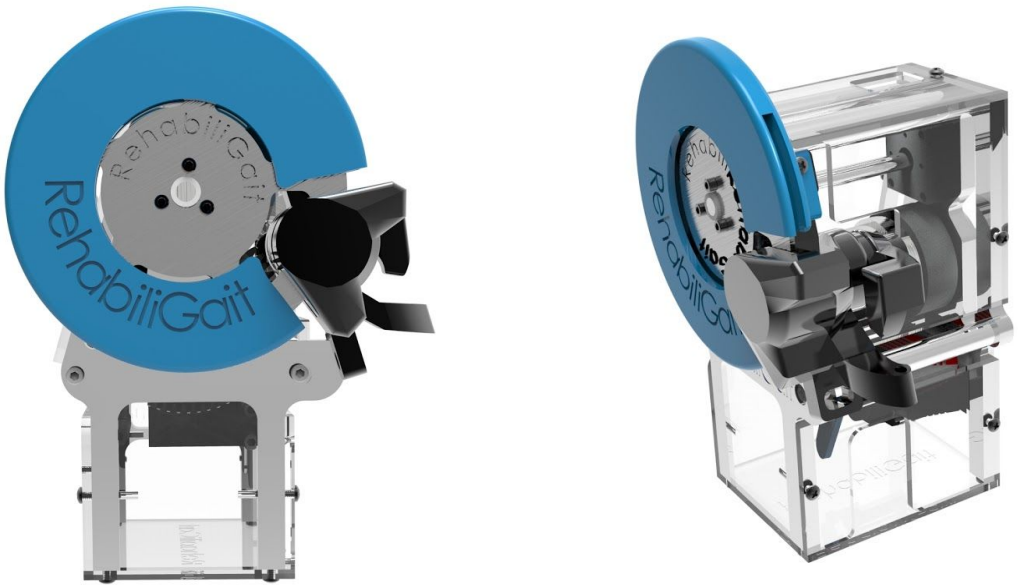


Figure 29: a) Rotor Cover, front **b)** Rotor Cover, isometric view

We calculated that the brake rotor would spin at 100 rpm during typical use, with a maximum expected velocity of 200 rpm. In order to safeguard users from injury, we enclosed the brake rotor with an ABS cover and all other mechanical components with an transparent acrylic housing (see **Figure 29a** and **29b**). The acrylic housing was designed to allow the servo horn to actuate the brake caliper without risk of it striking

the user. It also features a channel to direct wiring for the servo, encoder, and torque sensor upstream to the microcontroller without tangling with rotating power transmission components.

Machining and Construction

Machining and construction of Rehabiligait is broken down into four subassemblies. Subassembly one is the gear train consisting of the torque sensor and the 60 tooth spur gear. Subassembly two is the disc rotor consisting of the 15 tooth spur gear and disc hub. Subassembly three is the back plate assembly consisting of the servo and encoder. Subassembly four is the front plate assembly which holds the brake in place. Subassembly five consist of hubs and plates needed to attach the resistive subsystem to the leg brace. Please refer to **Table 12 and Appendix A8** for subassembly breakdowns and part specifications.

Table 12. Subassembly Breakdown

Assembly	Part	Drawing Number
Assembly 1 (Dwg # A1)	Gear(60T)	P1-2
	Gear Hub	P1-3
	Gear Sleeve	P1-4
	Gear Hub Shaft	P1-5
	Gear Sleeve Shaft	P1-6
Assembly 2 (Dwg # A2)	Shaft	P2-2
	Disc Hub	P2-3
	Disc	P2-4
Assembly 3 (Dwg # A3)	Front Plate	P3-1
	Support Shaft	P3-4
	Encoder Bracket	P3-7
	Stepped Support Shaft	P3-11
Assembly 4 (Dwg # A4)	Back Plate	P4-1
Assembly 5 (Dwg # A5)	Pin Plate	P5-1
	Side Plate	P5-2

	Brake Hub	P5-3
Test Bench Assembly (Dwg # F1)	Bearing Mount	F1
	Motor Mount	F2
	Hub	F3
	D-Shaft	F4
	Brake Mount	F5
	Base	F6
	Wall	F7
	Groove Bearing Support	F8
	Hub Connection	F9
	Test Disc	F10

Final System Embodiment and Function

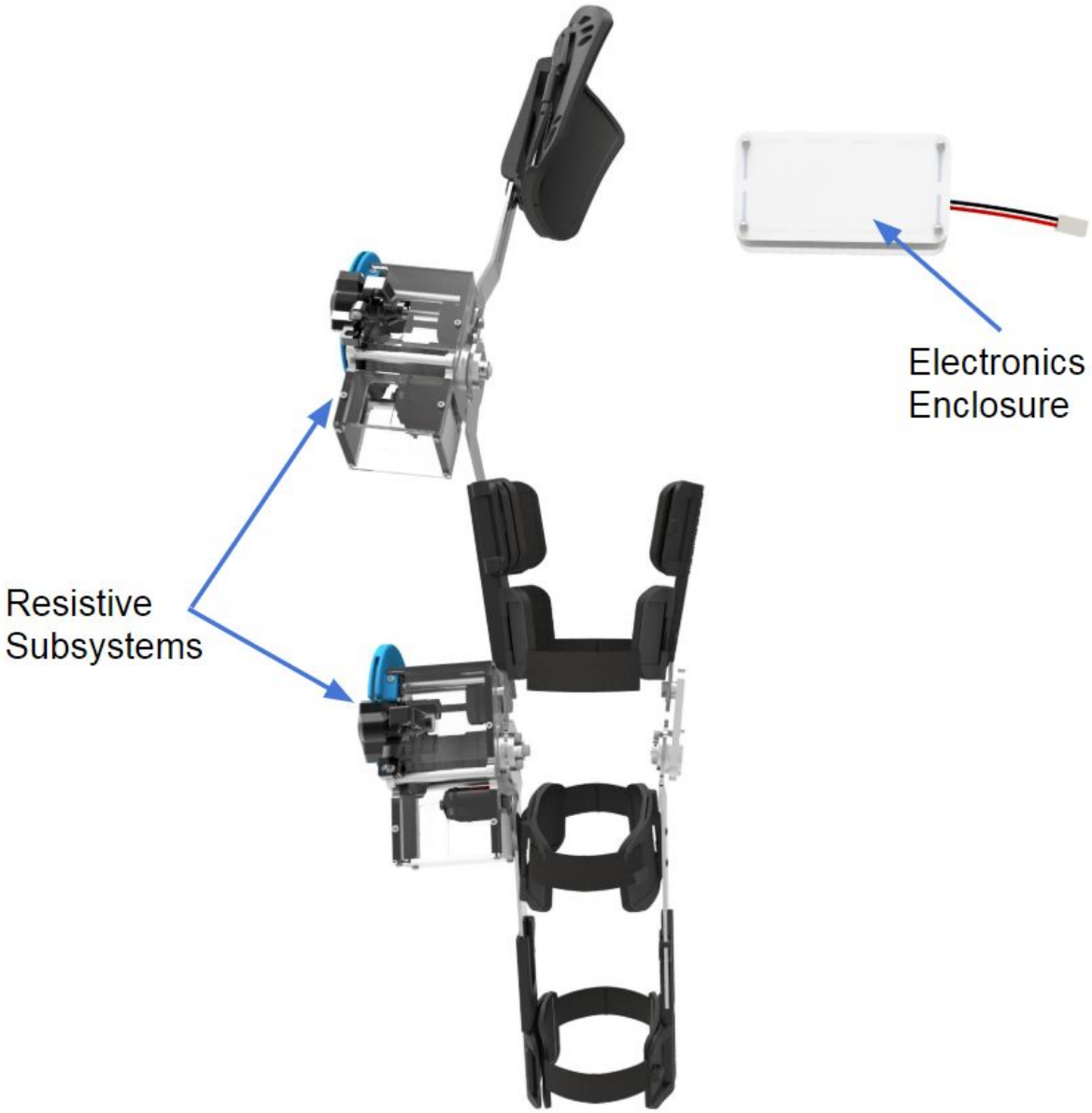


Figure 30: RehabailiGait Full System Embodiment

Figure 30 depicts Rehabiligait with all subsystems assembled. There is one resistive subsystem on each joint, and each resistive subsystem is connected to and powered by electronics enclosure strapped to patient's waist. When deviation from ideal gait path is detected, each resistive subsystem will apply the appropriate amount of resistive torque to sway patients back to their ideal gait path.

Validation and Testing

Torque Sensor Calibration and Testing

Torque sensor calibration was performed to ensure that the controls feedback would provide proper correction and to guarantee the user could accurately view torque applied. Torque sensor calibration was performed by applying added weight to the joint at a lever arm of 0.2m and applying enough friction in the brake to prevent it from slipping. **Figure 31** below shows an example of the output of the torque sensor calibration. First, 10 Nm was applied in the negative direction. This was followed by manually applying significant torque in the opposite direction to test hysteresis. When this torque was removed, the torque returned to zero, and upon reapplication of the 10 Nm, the sensor correctly reported the applied torque.

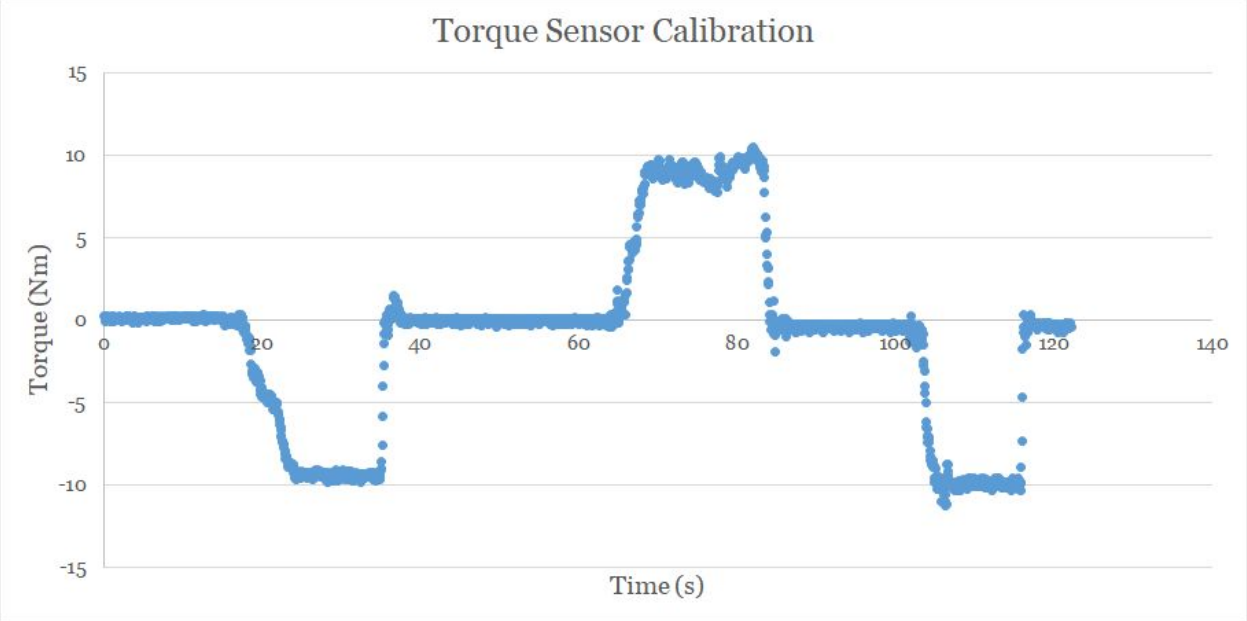


Figure 31: Torque Sensor Calibration Test

Torque Step Response Test

To validate the system's ability to apply a desired torque, a torque step response test was designed. The goal of this test was to validate a 100 ms or less response time. This test also measured torque application consistency. On average, the system took between 250

and 350 ms to respond to given step inputs, which is longer than our target response time. The majority of this deficit can be attributed to latency of the servo stretching the spring to the desired length. This theory is supported by the fact that system response time increased as desired applied torque (and thus required servo angular displacement) increased across the step response tests.

The torque application consistency is highly dependent on the absolute angle of the rotor. Despite numerous attempts to perfect the machining process of the hub and disc, the disc is not perpendicular with respect to the shaft, meaning that as the joint moved, the disc moved in and out of plane by approximately 1 mm. When the friction brake is applying some torque, this variation leads to a significant increase in torque application proportional to how far out of plane the disc is. While the controls system was able to correct for small differences between modeling and actual torque output, it was insufficient to correct this variation. To test the rest of the system, we performed additional brake characterization and torque step response on an identical region of the disc. The tests were performed on approximately 160 degrees of the disc, which corresponds to a 40 degree range in joint angle given the gear ratio.

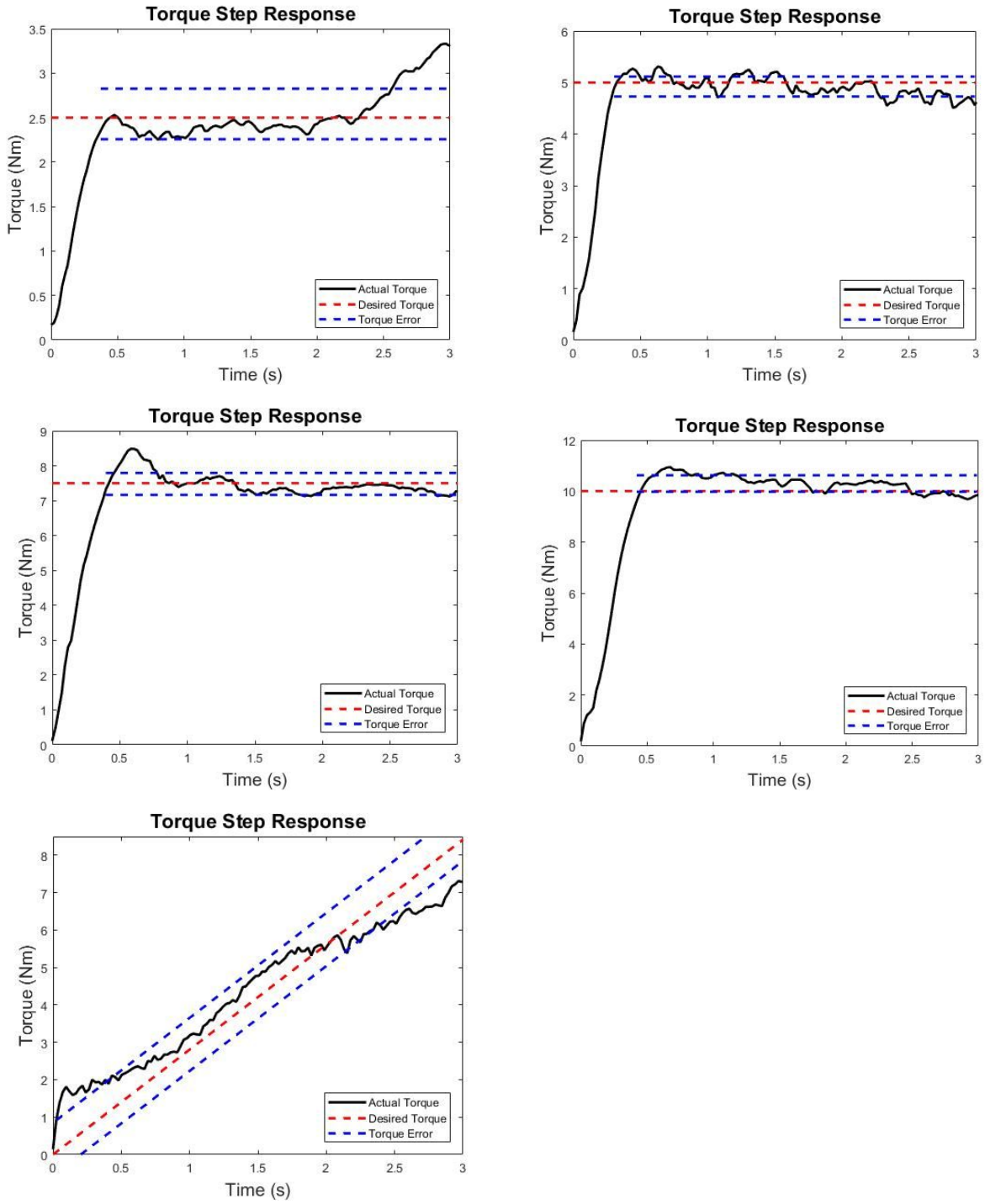


Figure 32: Torque Step Response Plots **a)** 2.5 Nm **b)** 5 Nm **c)** 7.5 Nm **d)** 10 Nm **e)** Linear Response

The torque step response plots can be seen in **Figure 32** above. The torque step response plot for 2.5 Nm application illustrates the effect that disc angle has on torque application. The torque application stays fairly consistent for the 2 s, but starts ramping

up as the disc leaves the range in which brake characterization was performed. This effect did not show up in later tests as it was simply easier to keep the rotor in the proper angle range under higher resistive torques. In each test, the steady state mean remained within 10% of the desired torque. The standard deviation for each test remained fairly constant at approximately 0.3 Nm.

Torque consistency and accuracy were not part of the original metrics for our device. Since the resistive torque is used only for haptic feedback to the user, a mean steady state error on the order of 10% is not a significant concern. An ideal metric would likely be on the order of 5% mean error, but the difference would not be easily noticeable by the user, as the most important effect is that resistive torque is increases with gait error. The standard deviation is more of a concern as changes in resistive torque at the same gait error could be misleading to the user. However, the user can only detect large variations in resistive torque anyway, so a resistive torque standard deviation of 0.2 Nm would be a good desired metric. While we did not originally establish torque consistency and accuracy metrics, the device is close to achieving the ideal metrics.

Weight Validation

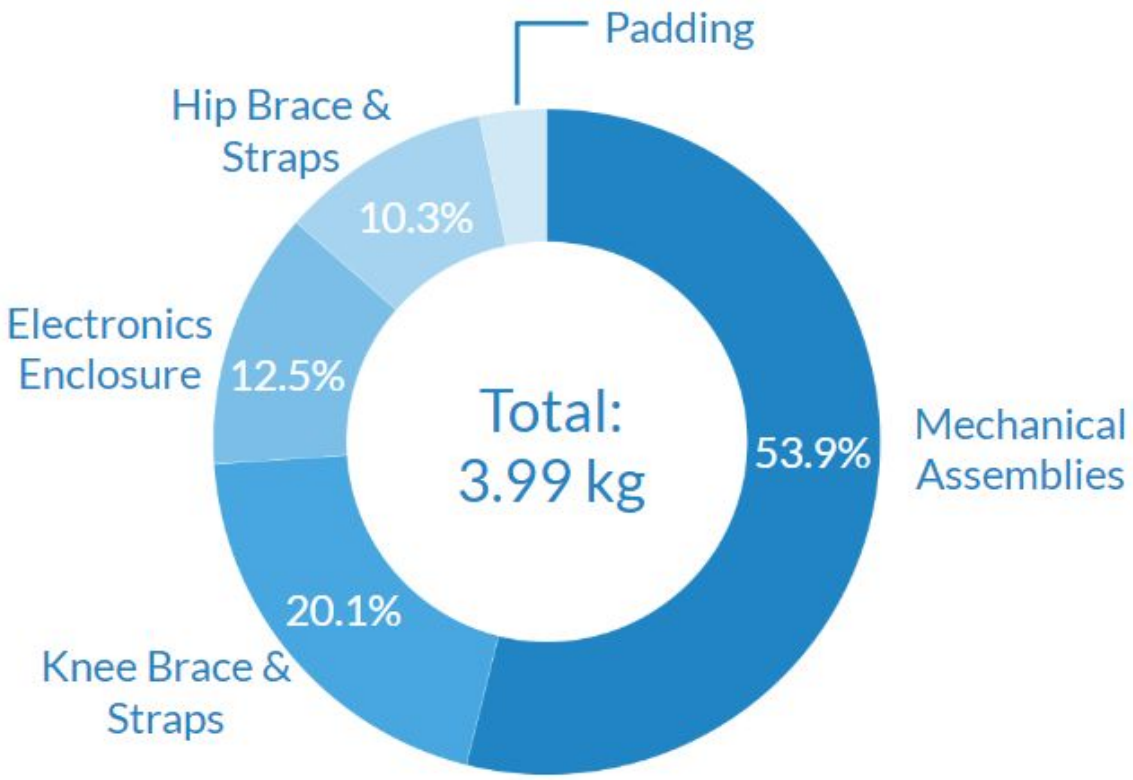


Figure 33: RehabiliGait Weight Distribution

To validate the target weight metric of RehabiliGait, each major component was weighed. This included the mechanical assemblies, the independent braces, the electronics enclosure, and additional added padding. The distribution of weight is shown in **Figure 33** above. The total weight of 4 kg was significantly under the target metric of 7 kg. If necessary, further weight optimization could be performed to reduce this total.

Battery Life Test

Team Walk This Way measured the drain rate of RehabiliGait's battery. We used a rechargeable Nickel-Metal Hydride 7.2 V 3800mAh Tenergy Battery [64]. In order to simulate standard use of the system, the system was programmed to run all computations and command the servo to maximum and minimum positions at 1 Hz. A voltage reading was taken at the beginning of the experiment and after 4.5 hrs of use. The voltage at the start of the experiment was 8.6 V. NiMH batteries can reach full charge at 115-130% of the reference voltage, so such a high reading is to be expected. At the end of the experiment the voltage read 7.6 V, and the system was still running with the servos visually moving at the same speed. This experiment validates the systems ability to last for the desired three physical therapy sessions of 90 minutes each.

Ergonomics Validation

RehabiliGait needs to be comfortable to wear during physical therapy sessions. An uncomfortable device can hamper the user's engagement, which leads to less effective rehabilitation. The time it takes a patient to put on and take off RehabiliGait is important to physical therapists (as previously discussed in the ergonomics section of the Background). Finally, for RehabiliGait to remain optimally useful throughout a physical therapy session, its location along the leg of the patient shouldn't shift over time.

Two members of Team Walk This Way ran six trials timing how long it took to put on and take off the device on another team member (see videos "**PuttingOn**" & "**TakingOff**" in the Media folder). **Table 13** shows the recorded times for these trials. The average time that it took to equip and take off the device was 31.2 and 10.2 seconds respectively. These low time values indicate RehabiliGait will fit seamlessly into physical therapy sessions and allow physical therapists more time to actually work on rehabilitation instead of setting up the device.

Table 13: Time to Equip and Detach Device

Trial	Equipping Time (s)	Detaching Time (s)
1	28	6
2	37	17
3	24	6
4	32	10
5	37	14
6	29	8

In addition to recording the above attachment and detachment times of RehabiliGait, Team Walk This Way observed whether or not the device slid down the leg of the user during use. Throughout all six trials, we found no evidence of slipping occuring. The user described the device as being "very tightly secured" around his waist and knees, specifically mentioning that the "velcro straps really helped fasten the hip and knee brace" well enough to prevent any misalignment of the device with respect to the knee and hip joints. Misalignment of the device with the joints is a serious problem because it affects the accuracy of the encoder, the effectiveness of the friction brake, and the angle joint space. All of these impact could all negatively harm the patient's gait training process.

Sound Testing

We ran a sound test to evaluate how loud the RehabiliGait device would be to the user. This is a necessary evaluation because the device's level of sound could irritate the patient and physical therapist which would distract them from their gait training process. We used a decibel meter application to record and approximate the amount of decibels emitted by the entire system (**Table 14** & **Figure 34**).

Table 14: Sound Testing

Average Sound (dB)	Maximum Sound (dB)
60	83

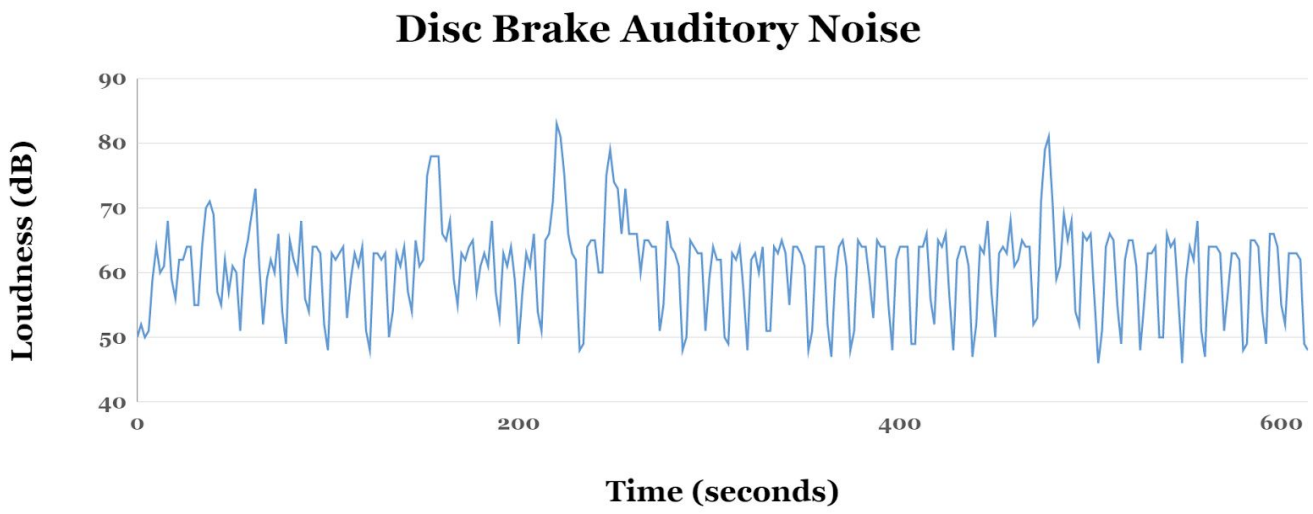


Figure 34: Loudness plotted over time

The average loudness recorded was 60 dB which is comparable to a typical conversation [32]. This value falls below our desired max loudness metric of 65 dB.

Brake Subsystem Structural Validation

For RehabiliGait to be considered robust, we needed to verify that no structural component will fail during use before torque testing starts. For the structural validation, we cantilevered weights on the end of each leg brace for 30 seconds as shown in **Figure 35**. Weights were increased incrementally to apply torque from 0-10 N-m. Set screws did come loose during testing, but we were able to remedy the issue using Loctite 648. No problems were identified with other structural components during testing.

Fig 35: Structural validation with 9 N-m

Brake Characterization

Brake characterization was performed to validate RehabiliGait's ability to apply a desired torque as a function of servo angle. In addition, brake characterization is needed to provide a first order estimate for the servo angle to apply a desired torque. A linear fit was deemed sufficient in successive torque step response tests as it proved as accurate as a third order polynomial that provided a tighter fit.

The brake characterization was performed using the friction brake test setup shown in **Figure 36**. The servo was actuated to a set angle, pulling the calipers closed. A motor

actuated the disc, and the reported steady state current was used to determine the resistive torque using a motor characterization.

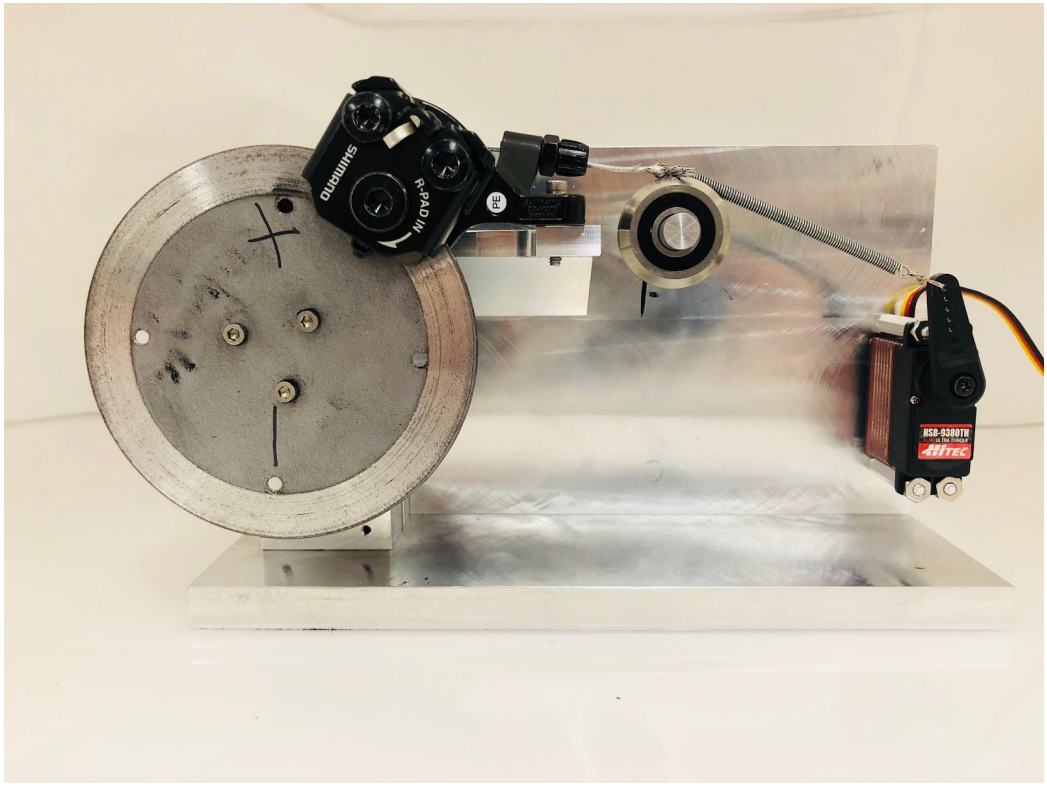


Figure 36: Friction Brake Test Setup

The results of the test can be found in **Figure 37** below. A servo angle range of 150 μs is sufficient to provide the full range of necessary torques given the dimensions and spring. The relationship between torque applied and servo angle is highly linear over this range. This characterization is used to provide a first order estimate for the necessary servo angle to provide the desired torque in Arduino's controls system. Error was negligible in this characterization.

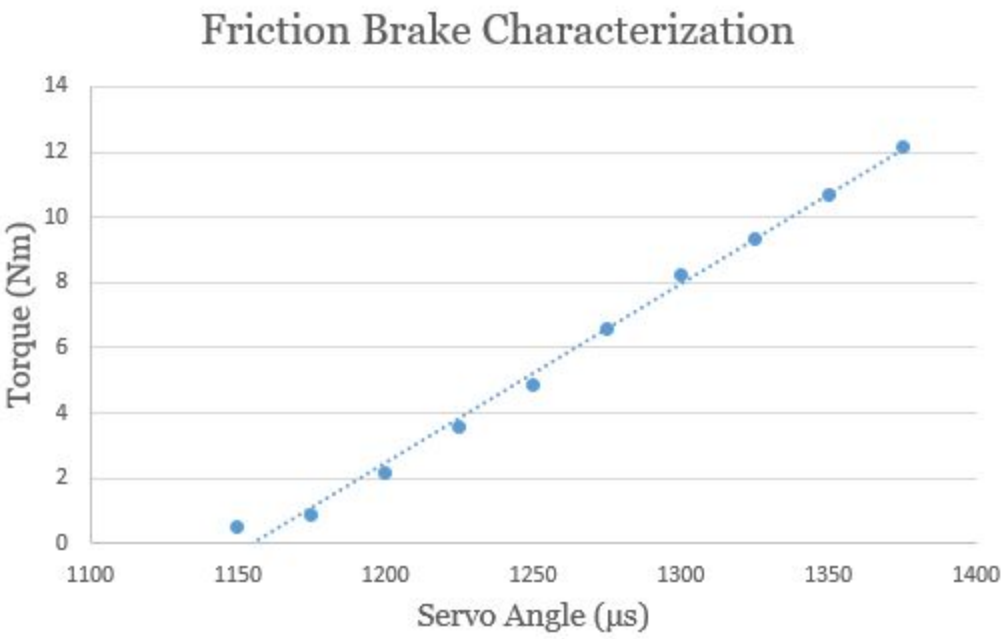


Figure 37: Friction Brake Characterization

Encoder Validation

To validate RehabiliGait's ability to accurately measure the angles of the hip and knee joints, an encoder test was designed. This test was meant to validate a data rate of at least 100 kHz.

The data rate was calculated empirically by directly measuring the time it took the arduino code to read the from the encoder. This was done using the inbuilt function micros. The measured average reading time was 500 kHz.

Wear Test

We didn't conduct a stand alone wear test due to time constraints. Rather, we logged all brake pad use during the characterization, testing, and validation of our project as presented in **Table 15**. At the end of our project, we assessed the brake pad wear as presented in **Table 16** and made a prediction about replacement frequency based.

Table 15: Brake Pad Use

Scenario	Torque Application	Time(hr)
Brake Characterization	0-10 N-m	22
Torque Validation	0-10 N-m	14
GUI Testing	0-10 N-m	5
Ergonomic Testing	5 N-m	1
		Total: 42

Table 16: Brake Pad Material Wear

	Original Brake Material Thickness	Brake Material Thickness after Use
Left Pad 1 (knee)	2.10 mm	2.08 mm
Right Pad 1 (knee)	2.09 mm	2.05 mm
Left Pad 2 (hip)	2.08 mm	2.07 mm
Right Pad 2 (hip)	2.09 mm	2.07 mm

From **Table 16** we can see that brake pads on the knee joint experience more wear than that of hip joint. This is expected because knee joint experiences greater degree of movement. Theoretically, brake pads do not have to be replaced until all brake material is depleted. However, brake cables were tensioned in such way that new brake pads do not contact the disc rotor upon installation. This is to ensure that the resistive subsystems do not apply any torque when the caliper is fully open. The caveat is that at the furthest reach of the servo arm, the caliper cannot apply enough normal force for 10 N-m when the thickness of brake material decreases by more than 1 mm. Using 1 mm

brake material thickness reduction as the cutoff, recommended replacement frequency for knee and hip joints are 4 months and 8 months respectively.

Full System Validation

The system was fully validated by testing on one of the team members. The team member was asked to perform 2 gaits: an incorrect gait (drop foot) and a correct gait. The user interface was then used to determine whether the correct angles were being measured and that the correct torques were being applied when the user deviated from the gait, as shown in videos **"incorrectgaitsimulation.mov"** and "**correctgaitsimulation.mov"**(videos can be found in the Video subfolder of the Media folder). The team member was then asked to gauge jerkiness of the torque applied at the various joints on a scale of 1 - 10 with 10 being very jerky. The hip was given a score of 6 and the knee a score of 10. The high jerkiness score can likely be attributed to the response time, and the higher knee score is due to the fact that the knee moves at higher angular velocities and would require quicker updates.

Discussion

Validation Results

Table 17: Validation Summary

Validation Test	Metric	Goal	Actual
Torque Sensor	Torque Reading	0, 10 Nm	0, 10 Nm
	Response Time	100 ms	300 ms
Torque Step Response	Mean error	5%	10%
	Standard Deviation	0.2 Nm	0.3 Nm
Weight	Total System Weight	7 kg	4 kg
Battery Life	System Runtime	4.5 hr	4.5 hr
Ergonomics	Put on, take off time	7, 5 minutes	31.2, 10.2 seconds
Sound	Sound Produced	65 dB	60.7 dB (83 dB max)
Brake Structural Validation	Apply Maximum Torque	10 Nm	10 Nm
Brake Characterization	Torque Range	0.5-10 Nm	0.5-10 Nm
Encoder	Data Rate	100 kHz	500 kHz
Wear	Brake Pad Life	1 month	4-8 months
Full System	Jerkiness Rating (knee, hip on a 0-10 scale)	4, 4	6, 10

The majority of validation tests proved that the system met its desired metrics, as shown in **Table 17**. The ergonomic test demonstrated that the device can be taken on and off

quicker than the desired metrics. The battery lasted more than 4 hours under average use. The device was shown to produce an average noise value of 60.7 dB, comparable to the 60 dB range that constitutes normal conversation. The friction brake was able to supply torques throughout the desired torque range, and this relationship was characterized. The torque sensor and encoder sensing components were calibrated and tested displaying sufficient accuracy and resolution. The wear test showed that brake pads need to be replaced on an order of months, which is reasonable for physical therapists. The torque step response test showed that the system was capable of following the desired torque, but experienced a response time larger than the desired metric.

Recommendations

The torque step response test showed that the system was capable of following the desired torque with a steady state standard deviation of 0.3 Nm, but has a response time 350 ms from 0-10 Nm. The response time is reduced for smaller fluctuations in desired torque dropping to 250 ms for a test from 0-5 Nm, but still remains larger than our desired metric of 100 ms. This test was also only performed over a subsection of the disc. In order to maintain torque consistency during normal operation, the system would need to be redesigned either eliminating the angle of the disc relative to the shaft, or accommodating it.

To sufficiently reduce system response time and minimize torque inconsistencies, the friction brake subsystem would need to be redesigned. In order to reduce the response time, the spring-servo system would have to replaced with a DC motor that directly actuates the friction brake. The torque applied would be a function of the current applied to the DC motor, which can be controlled through PWM. This system would respond more quickly as it requires a much smaller input of potential energy to reach steady state. Torque inconsistency could be improved through the design of a custom friction brake. The current bike brake calipers currently operate by keeping one end static, and actuating the other other end of the caliper through the cable. This works properly if the caliper is completely aligned and the disc remains in the same position; however, as the disc moves in and out of plane, the disc moves closer to one end of the calipers which increases the normal force. Redesigned calipers with ends that can move freely as a unit, but can be controlled in relative displacement, would be more resistant to disc displacements.

Budgets, Sponsorships, and Resources

Team Walk This Way came in under budget at $2171 of total expenditure over the course of this project. This was in part due to the team receiving $1100 worth of sponsorships from HiTec and Loadstar Sensors. The breakdown of the most expensive purchases of our project are displayed in **Figure 38** below.

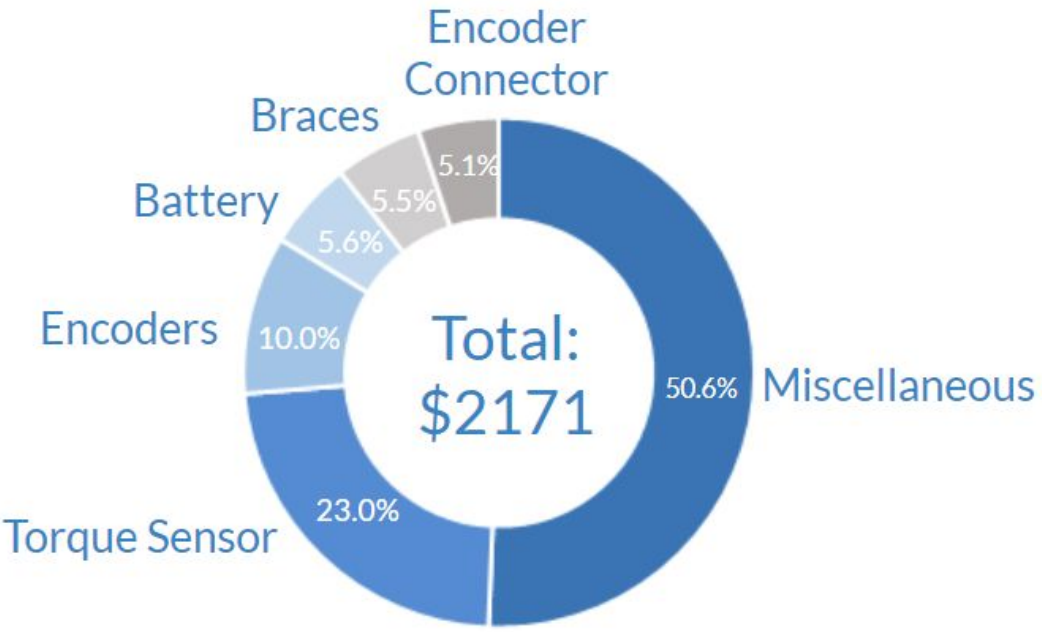


Figure 38: Budget Breakdown

Sponsorships

Team Walk This Way were fortunate to receive $600 worth of sponsorships from HiTec RCD, through a donation of three units of HiTec HSb9380TH servo motors. We also received a $500 sponsored discount on the purchase of two units of RS-T1 20 Nm reactionary torque sensors from LoadStar Sensors. Team Walk This Way is grateful for these donations and thanks HiTec and LoadStar for their generous contributions to our project.

Intellectual Property

Team Walk This Way will not be pursuing obtaining any intellectual property pertaining to any of the work done over the course of this project. The device and any data collected over the course of this project may be repurposed for existing research by Dr. Paul Stegall and Dr. Lauri Bishop. In addition, we hope that our project inspires future research into active learning rehabilitative technology.

References

1. M.W. Brault, et. al., “Prevalence and Most Common Causes of Disability Among Adults --- United States”, *Morbidity and Mortality Weekly Report*, vol. 58, no. 16, 2005, pp. 421-426.
2. E. J. Benjamin et. al, “Heart Disease and Stroke Statistics—2017 Update: A Report From the American Heart Association”, *Circulation*, vol. 135, no. 10, 2017, pp. 230.
3. C. A. Wamsley, R. Rai, and M. J. Johnson, “High-force Haptic Rehabilitation Robot and Motor Outcomes in Chronic Stroke”, *International Journal of Clinical Case Studies*, vol. 3, 2017, pp. 22-23.
4. A. S. Go, et. al., “Executive Summary: Heart Disease and Stroke Statistics--2013 Update: A Report From the American Heart Association,” *Circulation*, vol. 127, no. 1, Jan. 2013, pp. 143–152.
5. D. T. Wade, et. al., “Walking after Stroke,” *Scand J Rehab Med*, vol. 19, 1987, pp. 25–30.
6. M. Lotze, et. al., “Motor learning elicited by voluntary drive,” Brain, vol. 126, no. 4, Jan. 2003, pp. 866–872.
7. D. Winter, *Biomechanics and motor control of human movement*. 4th ed. Hoboken, N.J.: Wiley, 2009, pp. A.1-7.
8. E. M. Simonsick, et. al., “Measuring Fitness in Healthy Older Adults: The Health ABC Long Distance Corridor Walk”, *Journal of American Geriatrics Society*, vol. 49, 2001, pp. 1544-1548.
9. A. B. Newman, et. al., “Association of Long-Distance Corridor Walk Performance With Mortality, Cardiovascular Disease, Mobility Limitation, and Disability”, *Journal of the American Medical Association*, vol. 295, no. 17, May 2006, pp. 2018-2026.
10. S. L. Patterson, et. al., “Determinants of Walking Function After Stroke: Differences by Deficit Severity”, *Arch Phys Med Rehabil*, vol. 88, Jan. 2007, pp. 115-119.
11. Westmoreland Stroke Foundation, (2015). *Drop Foot*. [image] Available at: http://www.milkaclarkestrokefoundation.org/rehab-corner/category/foot-drop [Accessed 27 Apr. 2018].
12. D. Kerrigan, et. al., “Hip Hiking and Circumduction: Quantitative Definitions”, *American Journal of Physical Medicine & Rehabilitation*, vol. 79, no. 3, 2000, pp. 247-252.
13. P. Levangie and C. Norkin, (2017). *Hip Hiking Diagram*. [image] Available at: http://fadavispt.mhmedical.com/data/books/1862/levangiejoint_ch10_f021.png [Accessed 24 Oct. 2017].

14. M. Mumenthaler and H. Mattle (2004). *Neurology*. [image] Stuttgart: George Thieme Verlag. [Accessed 24 Oct. 2017].
15. R. A. States, E. Pappas, and Y. Salem, "Overground Physical Therapy Gait Training for Chronic Stroke Patients with Mobility Deficits", *Stroke,* vol. 40, no. 11, 2009, pp 1-48.
16. S. Lennon, D. Baxter, and A. Ashburn, "Physiotherapy based on the Bobath concept in stroke rehabilitation: a survey within the UK," *Disability and Rehabilitation*, vol. 23, no. 6, 2001, pp. 254–262.
17. B. J. Kollen, et. al., "The Effectiveness of the Bobath Concept in Stroke Rehabilitation: What is the Evidence?," *Stroke*, vol. 40, no. 4, Aug. 2008, pp. 89-97.
18. U. Bogataj, et. al., "The rehabilitation of gait in patients with hemiplegia: a comparison between conventional therapy and multichannel functional electrical stimulation therapy", *Physical Therapy*, vol. 75, no. 6, 1995, pp. 40+.
19. G. Colombo, et. al., "Treadmill training of paraplegic patients using a robotic orthosis", *J Rehabil Res Dev*, vol. 37, no. 6, 2000, pp. 700.
20. K. Y. Nam et. al., "Robot-Assisted Gait Training(Lokomat) Improves Walking Function and Activity in People with Spinal Cord Injury: a Systematic Review", *Journal of Neuroengineering and Rehabilitation*, vol. 14, no. 24, 2017, pp. 1-13.
21. Lokomat - Hocoma, 2017. [Online]. Available: https://www.hocoma.com/solutions/lokomat/. [Accessed: 09-May-2017].
22. Lokomat, *Lokomat Product Presentation*. [Online image] Available from: https://i.ytimg.com/vi/cXyeXT--Kbs/maxresdefault.jpg [Accessed 27 Oct. 2017].
23. Exoskeleton Report, *Esko GT*. [Online image] Available from: http://exoskeletonreport.com/product/ekso-gt/ [Accessed 28 Oct. 2017].
24. E. Strickland, "Demo: The Ekso GT Robotic Exoskeleton for Paraplegics and Stroke Patients," *IEEE Spectrum: Technology, Engineering, and Science News*, Sep. 2016. [Online]. Available: https://spectrum.ieee.org/the-human-os/biomedical/bionics/paraplegic-man-walks-in-ekso-robotic-exoskeleton-to-demo-its-killer-app. [Accessed: 28-Apr-2018].
25. M. Cunningham and S. C. Cramer, "Reward Improves Long-Term Retention of a Motor Memory through Induction of Offline Memory Gains," *MD Conference Express*, vol. 13, no. 1, Jan. 2013, pp. 35–35.
26. J. F. Israel, et. al., "Metabolic Costs and Muscle Activity Patterns During Robotic- and Therapist-Assisted Treadmill Walking in Individuals With Incomplete Spinal Cord Injury," *Physical Therapy*, vol. 86, no. 11, Jan. 2006, pp. 1466–1478.
27. C. J. Hasson, J. Manczurowsky, and S.-C. Yen, "A reinforcement learning approach to gait training improves retention," *Frontiers in Human Neuroscience*, vol. 9, art. 459, 2015, pp. 1-9.

28. P. Stegall, D. Zanotto, and S. K. Agrawal, “Variable Damping Force Tunnel for Gait Training Using ALEX III”, *IEEE Robotics and Automation Letters*, vol. 2, no. 3, 2017.
29. A. Ng and A. Chan, “Finger Response Times to Visual Auditory and Tactile Modality Stimuli”, *Lecture Notes in Engineering and Computer Science*. vol. 2196, 2012, pp. 1449-1454.
30. “Medicare Limits on Therapy Services,” *Medicare* , Jan. 2017. [Online]. Available: https://www.medicare.gov/pubs/pdf/10988-Medicare-Limits-Therapy-Services.pdf. [Accessed: 28-Apr-2018].
31. U.S. Census Bureau, “National Health and Nutrition Examination Survey,” *Health and Nutrition*, vol. 135, 2011.
32. Cohrssen, Barbara, “Now Hear This! Noise Hazards on the Job,” *Tradeswomen: San Francisco*, vol. 2, no. 1, Apr. 1982, pp. 32-37.
33. “Physical Medicine Devices,” *eCFR — Code of Federal Regulations*, Nov. 2017. [Online]. Available: https://www.ecfr.gov/cgi-bin/text-idx?SID=8ecd02461fef6ae1ab6d21061f18d501&mc=true&node=se21.8.890_13450&rgn=div8. [Accessed: 04-Nov-2017].
34. NFPA, “Chapter 10: Electrical Equipment,” *Healthcare Facilities Code*, 2018, pp. 90-93.
35. “Substances for Use Only as Components of Articles Intended for Repeated Use,” *eCFR — Code of Federal Regulations*, 2017 . [online] Available at: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfCFR/CFRSearch.cfm?fr=177.2600 [Accessed 29 Apr. 2018].
36. “Chapter 3: Building Planning,” *2015 International Residential Code*, 2017. [online] Available at: https://codes.iccsafe.org/public/document/toc/553/ [Accessed 5 Nov. 2017].
37. MR Damper, “RD-8041-1 MR Damper (Long Stroke),” *Lord Mr Store*. [Online]. Available: http://www.lordmrstore.com/lord-mr-products/rd-8041-1-mr-damper-long-stroke. [Accessed: 07-Apr-2018].
38. Magnetic Particle Brake, “Warner Electric Magnetic Particle Brake for Labeling Machine,” *Warner Electric*. [Online]. Available: http://www.altramotion.com/newsroom/2016/02/ap-magnetic-particle-brake-for-labeling-machine. [Accessed: 07-Apr-2018].
39. Back-drivable Motor, “DC Motor 12V High Speed Torque Multi-Purpose Motor PCB Drill RC Cars Airplane,” *eBay*. [Online]. Available: https://www.ebay.in/itm/DC-Motor-12V-High-Speed-Torque-Multi-Purpose-Motor-PCB-Drill-RC-Cars-Airplane-/121934381521. [Accessed: 07-Apr-2018].

40. Hydraulic Friction Brake, “Shimano M640 Zee Hydraulic Disc Brakes,” *Vital MTB*. [Online]. Available: https://www.vitalmtb.com/product/guide/Hydraulic-Disc-Brakes,11/Shimano/M640-Zee,12169. [Accessed: 08-Apr-2018].
41. Cable Friction Brake, “Avid,” *Competitive Cyclist*. [Online]. Available: https://www.competitivecyclist.com/avid-road-bb7-disc-brake. [Accessed: 08-Apr-2018].
42. “RD-8041-1 MR Damper (Long Stroke),” *Lord Mr Store*. [Online]. Available: http://www.lordmrstore.com/lord-mr-products/rd-8041-1-mr-damper-long-stroke. [Accessed: 07-Apr-2018].
43. “RD-8191 TFD Device - Cross Drilled Shaft 5 Nm,” *Lord Mr Store*. [Online]. Available: http://www.lordmrstore.com/lord-mr-products/rd-8191-5-nm-tfd-device-cross-drilled-shaft. [Accessed: 06-Apr-2018].
44. “Magnetic Particle Clutches and Brakes | Warner Electric,” *Warner Electric*. [Online]. Available: http://www.warnerelectric.com/products/torque-control-products/magnetic-particle/magnetic-particle-clutches-and-brakes. [Accessed: 06-Apr-2018].
45. “MAGPOWR SOFSTEP® Magnetic Particle Clutches and Brakes | Maxcess Americas,” *Maxcess International*. [Online]. Available: http://www.maxcessintl.com/magnetic-particle-clutches
46. “60W 1800 or 3200 RPM 90mm Industrial DC Motor 12V or 24V Straight Shaft,” *Motion Dynamics*. [Online]. Available: https://www.motiondynamics.com.au/g.p.g-60w-90mm-brushed-dc-motor-12v-or-24v-straight-shaft.html. [Accessed: 07-Apr-2018].
47. F. Imaduddin, S. A. Mazlan, and H. Zamzuri, “A design and modelling review of rotary magnetorheological damper,” *Materials & Design*, vol. 51, 2013, pp. 575–591.
48. “Magnetic Particle Clutches and Brakes | Warner Electric,” *Warner Electric*. [Online]. Available: http://www.warnerelectric.com/products/torque-control-products/magnetic-particle/magnetic-particle-clutches-and-brakes. [Accessed: 28-Oct-2017].
49. A. H. Gosline and V. Hayward. “Eddy current brakes for haptic interfaces: design, identification, and control”. *IEEE/ ASME Trans. Mechatron*. vol. 13, 2008, pp. 669–677.
50. K. Nice, “How Disc Brakes Work,” *Disc Brake Basics - How Disc Brakes Work | HowStuffWorks*, 21-Aug-2000. [Online]. Available: https://auto.howstuffworks.com/auto-parts/brakes/brake-types/disc-brake1.htm. [Accessed: 28-Oct-2017].

51. "Disc Friction," *Adaptive Map - Disc Friction*. [Online]. Available: http://adaptivemap.ma.psu.edu/websites/friction/disc_friction/discfriction.html. [Accessed: 28-Oct-2017].
52. Horsforth School, "Hydraulic Brake Diagram," *A question of pressures*, 2015. [Online] Available at: http://www.sciwebhop.net/sci_web/science/ks3/worksheets/word/9L%5C./9Lb3_files/image009.jpg [Accessed 7 Feb. 2015].
53. "AFTERPARTZ NV-5 G3/ HS1 Bike Disc Brake Kit Front and Rear 160mm Caliper Rotor BB5 BB7 BB-5 BB-7," *Amazon.com*, 2018. [Online]. Available: https://www.amazon.com/AFTERPARTZ-Brake-Front-Rotor-handle/dp/B00JVLBGFC/ref=sr_1_1?ie=UTF8&qid=1525099584&sr=8-1&keywords=afterpartz+bike+brake
54. University of Chicago, "Chapter 17: Materials Expansion Coefficients," *Laser and Optics User's Manual*, 2002, pp. 1-12.
55. M. Ashby, "Materials and Process Selection Charts," *CES 2010 EDUPACK*, Jan. 2010, pp. 1-40.
56. "HSB-9380TH Ultra Torque, Brushless, Titanium Gear Servo," *HiTec Multiplex*. [Online]. Available at: http://hitecrcd.com/products/servos/digital-brushless-servos/hsb-9380th-ultra-torque-brushless-titanium-gear-servo/product
57. "Modular Absolute Encoder," *CUI Inc*, May 2015, pp. 1-10.
58. "RST1 Reaction Torque Sensor," *Loadstar Sensors*, 2014, pp. 1-2.
59. "Precision, Low Cost, High Speed, BiFET Op Amp," *Analog Devices*, 2002, pp. 1-16.
60. "ATmega328/P," *Atmel*, 2016, pp. 215-223.
61. "HC Serial Bluetooth Products User Instructional Manual," *Guangzhou HC Information Technology Co., Ltd*, pp. 1-16.
62. "T Scope® Premier Post-Op Knee Brace – Breg, Inc.," Breg, Inc. [Online]. Available: https://www.breg.com/products/knee-bracing/post-op/t-scope-premier-post-op-knee-brace/. [Accessed: 28-Apr-2018].
63. "T Scope Hip Brace – Breg, Inc.," Breg, Inc. [Online]. Available: https://www.breg.com/products/hip-bracing/hip-bracing-post-op/t-scope-hip-brace/. [Accessed: 28-Apr-2018].
64. "Tenergy 7.2V 3800mAh High Power Flat NiMH Battery Packs for RC Cars & Sumo Robots," *Rechargeable Batteries | All-Battery.com*. [Online]. Available: http://www.all-battery.com/nimh-6S-11200.aspx. [Accessed: 30-Apr-2018].
65. Center for Devices and Radiological Health, "Medical Device User Fee Amendments (MDUFA) - FY 2018 MDUFA User Fees," U S Food and Drug Administration Home Page. [Online]. Available:

https://www.fda.gov/ForIndustry/UserFees/MedicalDeviceUserFee/ucm452519.htm. [Accessed: 30-Apr-2018].
66. "Facilities by State," Continuing Care Directory. [Online]. Available: http://www.carelookup.com/outpatient-rehab-directory/statelist.aspx. [Accessed: 30-Apr-2018].

Appendix

A1. Circuit Diagram

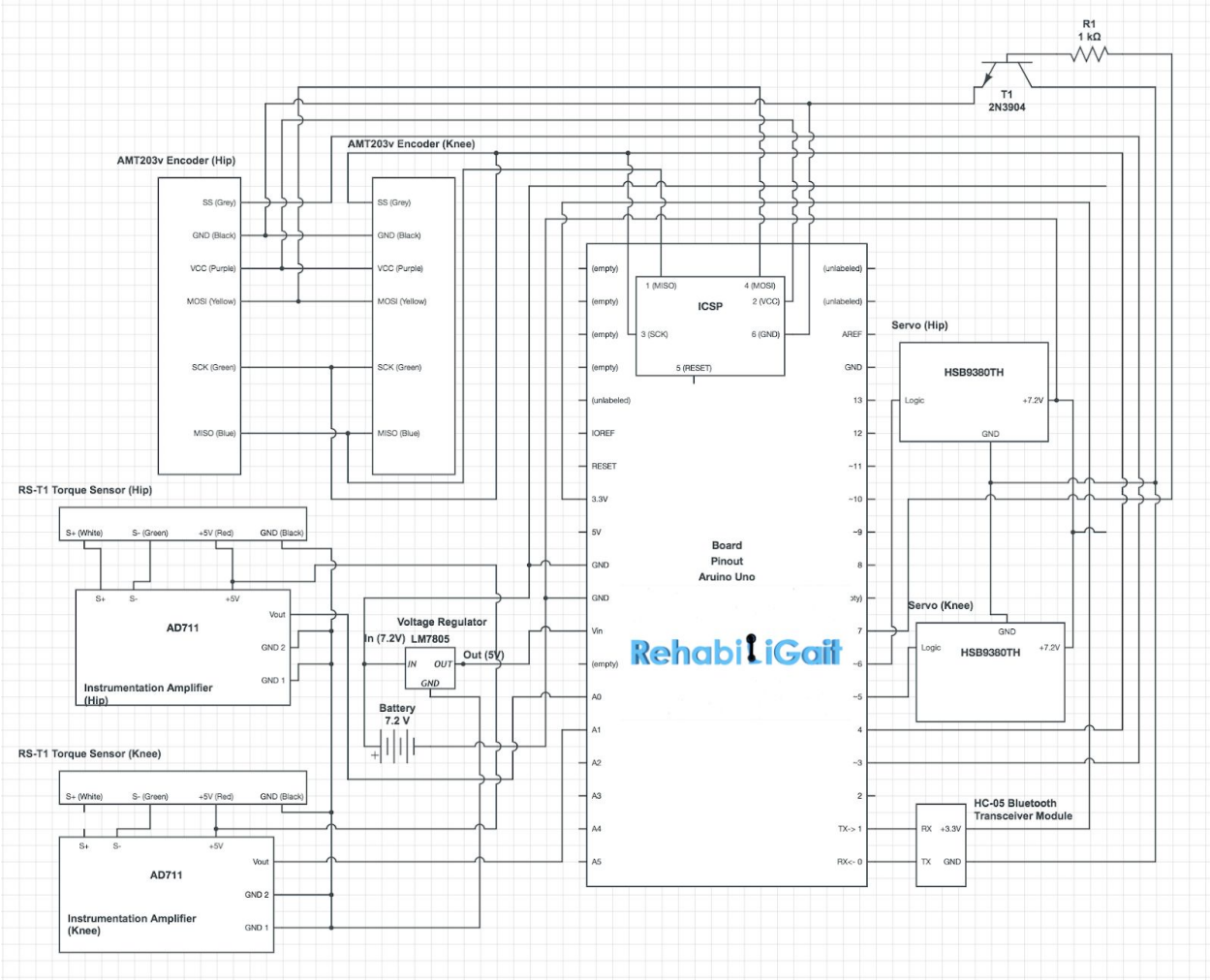


Figure 24: Full System Electronics Schematic

A2: Eddy Brake Inertial Calculation

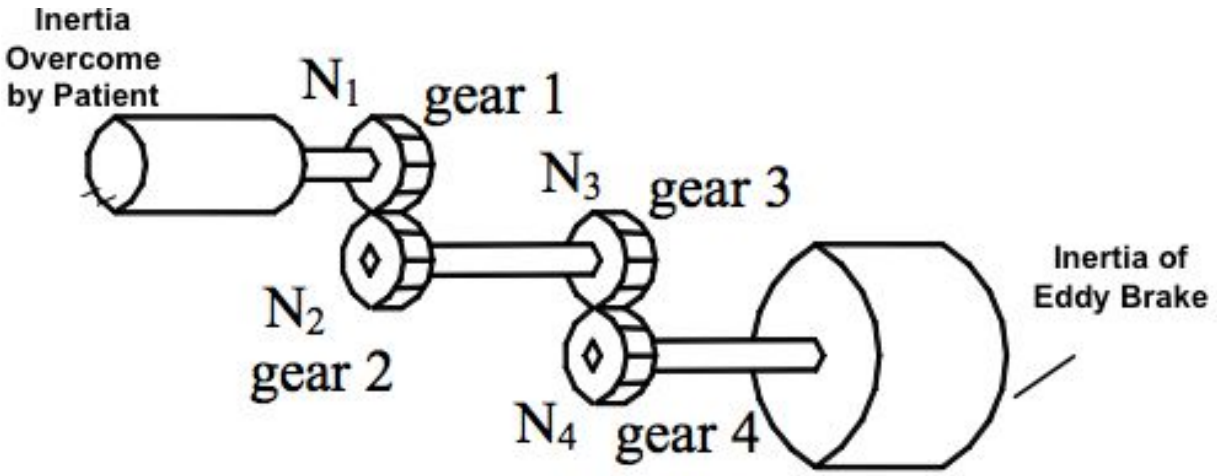


Figure XX: Eddy Current Brake Power Transmission

$$J_{patient} = J_{gear1} + (\frac{N_{gear1}}{N_{gear2}})^2 [J_{gear2} + J_{gear3} + (\frac{N_{gear3}}{N_{gear4}})^2 \{J_{gear4} + J_{Eddy\ Brake}\}]$$

$$J_{patient} \approx (\frac{N_{gear1}}{N_{gear2}} \frac{N_{gear3}}{N_{gear4}})^2 J_{Eddy\ Brake}$$ **(First Order Estimate)**

$$J_{Eddy\ Brake} = \frac{MR^2}{4}$$

$$Let\ (\frac{N_{gear1}}{N_{gear2}} \frac{N_{gear3}}{N_{gear4}})^2 = G$$

$M = \pi R^2 t \rho$
$R = 0.0051m$
$t = 0.0032m$
$\rho = 2700 \frac{kg}{m^3}$
$G = 64$

$$J_{patient} \approx \frac{G^2 MR^2}{4} = \frac{\pi t \rho G^2 R^4}{4} = 0.19 kgm^2$$

Torque Applied on Patient

$$\tau_{patient} = J_{patient} \times \alpha_{knee} = 5.5 Nm$$

$\alpha_{knee} \approx 30 \frac{rad}{s^2}$ [7]

A3. Project Budget (Refer to full project budget in appendix folder)

RehabiliGait Project Budget			
Item Name	**Price($)**	**Quantity**	**Total ($)**
Magnets	80.64	6	483.84
Encoder and Codewheel	218.58	2	437.16
Microcontroller Cable	12.74	1	12.74
Bluetooth	11.00	1	11
Guide Rail and Bear Carriage	23.22	3	69.66
Gearbox	67.95	1	67.95
Key	0.70	1	0.7
Grease	8.25	1	8.25
AD623ARZ	38.50	10	385
Degraw - 4 pcs 50kg Load Cells and HX711 Combo Pack Kit | Load cell amplifier ADC weight sensor for Load cell Arduino | Bathroom scale kit	14.99	1	14.99
10Pcs BF350 Resistance Strain Gauge Strainmeter Weighing Sensor High Precision W	16.06	10	160.6
Quality VXB Set of 8 RM2-2RS 3/8 V-Groove Guide Bearing Sealed Ball Bearings Vgroove W2X	24.95	1	24.95
Clutch Cable	17.90	2	35.8
24 pitch 20 pressure angle 60 teeth gear	34.50	1	34.5
24 pitch 20 pressure angle 15 teeth gear	15.38	1	15.38
Precision Flanged Ball Bearing	42.80	4	171.2
303 Stainless Steel, 1/4"" Diameter, 18"" Long	12.64	1	12.64
BREG T Scope Right Hip Brace Black Adjustable Velcro - Post Op Orthotics	80.00	1	80
Torque Sensor Invoice	500.00	2	1000

A4. BOM and Projected Cost Estimate (Refer to full version in appendix folder)

RehabiliGait Full BOM and First Order Production Cost Estimate							
Part Name	Mass (g)	Material	Production Process	Material Cost Per Kilogram	Cost Per Part (Production Scale)	QTY.	Total Cost
RehabiliGait							
Hip Brace Assembly						1	$0.00
Hip Shell	92.0	ABS	Injection Molded	$38.00	$3.50	1	$3.50
Hip Pad Holder	96.4	ABS	Injection Molded	$38.00	$3.66	1	$3.66
Hip Slider	90.0	ABS	Injection Molded	$38.00	$3.42	1	$3.42
Hip Upper Metal Frame	107.8	Stainless Steel 304	Stamped	$34.00	$3.67	1	$3.67
Hip Lower Metal Frame	130.5	Stainless Steel 304	Stamped	$34.00	$4.44	1	$4.44
Hip Shell Pad	1.4	Polyeurathane Foam	Catalog Component	$15.00	$0.02	1	$0.02
Thigh Pad	0.4	Polyeurathane Foam	Catalog Component	$15.00	$0.01	1	$0.01
Hip Brace Hub	8.5	6061 Aluminum	Machined	$26.00	$0.22	1	$0.22
6_32 0.5 Inch Machine Screw	1.0	Stainless Steel 304	Catalog Component	-	$0.03	6	$0.16
Hip Connector Plate	83.0	6061 Aluminum	Machined	$26.00	$2.16	1	$2.16
Hip Shaft	6.9	6061 Aluminum	Machined	$26.00	$0.18	1	$0.18
	617.9						$21.43
Knee Brace Assembly						1	
Upper Metal Frame	128.9	Stainless Steel 304	Stamped	$34.00	$4.38	2	$8.77
Lower Metal Frame	228.2	Stainless Steel 304	Stamped	$34.00	$7.76	2	$15.52
Lower Slider	66.4	ABS	Injection Molded	$38.00	$2.52	2	$5.05
Upper Thigh Pad Holder	59.2	ABS	Injection Molded	$38.00	$2.25	2	$4.50
Lower Thigh Pad Holder	21.0	ABS	Injection Molded	$38.00	$0.80	2	$1.60
Upper Slider	20.8	ABS	Injection Molded	$38.00	$0.79	2	$1.58
Rivet		-	Catalog Component	-	$0.01	8	$0.05
Thigh Pad	0.4	Polyeurathane Foam	Catalog Component	$15.00	$0.01	6	$0.04
Calf Pad	0.3	Polyeurathane Foam	Catalog Component	$15.00	$0.00	2	$0.01
Calf Strap	0.4	Nylon	Catalog Component	$3.00	$0.00	6	$0.01
Front Plate	71.6	6061 Aluminum	Machined	$26.00	$1.86	1	$1.86
Pin Plate	11.6	6061 Aluminum	Machined	$26.00	$0.30	1	$0.30
6_32 0.375 Inch Machine Screw	0.6	Stainless Steel 304	Catalog Component	-	$0.07	8	$0.56
Knee Brace Hub	8.5	6061 Aluminum	Machined	$26.00	$0.22	2	$0.44
	609.3						$40.27
Resisitive Subassembly						2	
Front Plate Assembly						1	
Front Plate	403.0	6061 Aluminum	Machined	$26.00	$10.48	1	$10.48
Support Shaft	13.9	6061 Aluminum	Machined	$26.00	$0.36	4	$1.44

A5. Code Flowchart

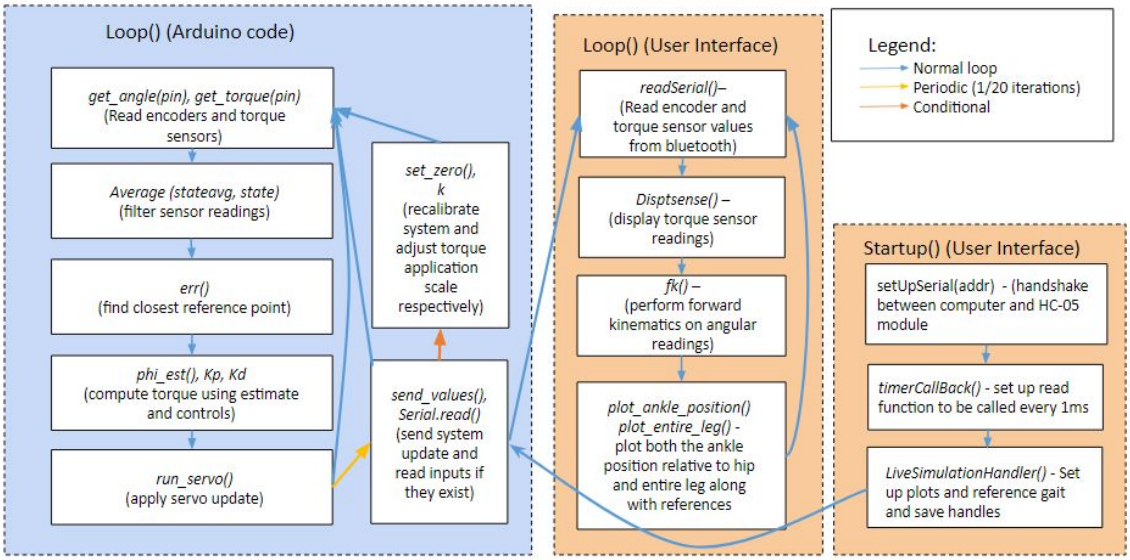


Figure 23: Code Structure Diagram

A6: Matlab

```
function varargout = DeadSimulationPage(varargin)
% DEADSIMULATIONPAGE MATLAB code for DeadSimulationPage.fig
%      DEADSIMULATIONPAGE, by itself, creates a new DEADSIMULATIONPAGE or raises the existing
%      singleton*.
%
%      H = DEADSIMULATIONPAGE returns the handle to a new DEADSIMULATIONPAGE or the handle to
%      the existing singleton*.
%
%      DEADSIMULATIONPAGE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DEADSIMULATIONPAGE.M with the given input arguments.
%
%      DEADSIMULATIONPAGE('Property','Value',...) creates a new DEADSIMULATIONPAGE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
```

```
%     applied to the GUI before DeadSimulationPage_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to DeadSimulationPage_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help DeadSimulationPage

% Last Modified by GUIDE v2.5 01-Apr-2018 22:04:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DeadSimulationPage_OpeningFcn, ...
                   'gui_OutputFcn',  @DeadSimulationPage_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before DeadSimulationPage is made visible.
function DeadSimulationPage_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to DeadSimulationPage (see VARARGIN)

% Choose default command line output for DeadSimulationPage
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes DeadSimulationPage wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Start up function.
function varargout = DeadSimulationPage_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Load Reference gait
handles.thickness
xh = xlsread('Winter_Appendix_data.xlsx','A1.Raw_Coordinate', 'E4:F109')/100;
xa = xlsread('Winter_Appendix_data.xlsx','A1.Raw_Coordinate', 'K4:L109')/100;
x = xa-xh;
handles.timer = timer('Name','dMyTimer',          ...
        'Period',0.1,                 ...
        'StartDelay',0,                   ...
        'TasksToExecute',inf,         ...
        'ExecutionMode','fixedSpacing', ...
        'TimerFcn',{@dtimerCallback,handles.figure1,hObject});
handles.switch = 1;
handles.t = 0;

% set up plot for ankle position
axes(handles.axes2);
handles.Pos = plot(x(:,1), x(:,2), '.r', NaN, NaN, '.b');
handles.time = 0;
legend('Reference', 'Actual');
title('Gait Characterization', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
```

```
xlabel('Ankle X Position (m)','FontSize',20,'Color','[0, 0.45, 0.74]');
ylabel('Ankle Y Position (m)','FontSize',20,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% set up plot for leg simulation
axes(handles.axes7);
handles.sim = plot(NaN, NaN, '-b', NaN, NaN, '.b', NaN, NaN, '-r', NaN, NaN, '.r');
xlim([-0.45, 0.55]);
ylim([-.9, 0.2]);
handles.Htext1 = text(handles.axes7, 0, 0, '');
handles.Htext2 = text(handles.axes7, 0, 0, '');
handles.Htext3 = text(handles.axes7, 0, 0, '');
axis equal
title('Leg Simulation', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('X Relative Position','FontSize',20,'Color','[0, 0.45, 0.74]');
ylabel('Y Relative Position','FontSize',20,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% set up plot for knee angle vs time
axes(handles.axes3);
handles.Hka = plot(NaN, NaN);
title('Knee Angle', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',20,'Color','[0, 0.45, 0.74]');
ylabel('Knee Angle (rad)','FontSize',20,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% set up plot for knee torque vs time
axes(handles.axes4);
handles.Hkt = plot(NaN, NaN);
title('Knee Torque', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',20,'Color','[0, 0.45, 0.74]');
ylabel('Knee Torque (Nm)','FontSize',20,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% set up plot for hip angle vs time
axes(handles.axes6);
handles.Hha = plot(NaN, NaN);
title('Hip Angle', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',20,'Color','[0, 0.45, 0.74]');
```

```
ylabel('Hip Angle (rad)','FontSize',20,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% set up plot for hip torque vs time
axes(handles.axes5);
handles.Hht = plot(NaN, NaN);
title('Hip Torque', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',20,'Color','[0, 0.45, 0.74]');
ylabel('Hip Torque (Nm)','FontSize',20,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% load stored files
p_id = getappdata(0, 'p_id');
ts = getappdata(0, 'ts');
v = [p_id, '-', ts, '.csv'];
handles.values = csvread([p_id, '-', ts, '.csv']);

guidata(hObject,handles);
start(handles.timer);
varargout{1} = handles.output;


% --- Not used.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Not used.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
```

```
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Not used.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = [get(handles.Pos(2, 1),'XData'), get(handles.Pos(2, 1),'YData'),
get(handles.Hka,'XData'), get(handles.Hka,'YData'), get(handles.Hkt,'XData'),
get(handles.Hkt,'YData')...
        , get(handles.Hha,'XData'), get(handles.Hha,'YData'), get(handles.Hht,'XData'),
get(handles.Hht,'YData')];
p_id = getappdata(0, 'p_id');
d = datetime('now');
d_str = datestr(d, 30);
filename = [p_id,'-', d_str, '.csv'];
csvwrite(filename, data);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
stop(handles.timer);
close;


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
```

```
% --- Otherwise, executes on mouse press in 5 pixel border or over pushbutton2.
function pushbutton2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Function that pauses and stops playback.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.switch == 1
        handles.switch = 0;
        stop(handles.timer);
        set(handles.pushbutton4, 'String', 'Continue');
else
        handles.switch = 1;
        start(handles.timer);
        set(handles.pushbutton4, 'String', 'Pause');
end


function [] = timerCallback(~,~,guiHandle,hObject)
```

Timer callback function for dead simulation

```
if ~isempty(guiHandle)
        % get the handles
        handles = guidata(guiHandle);
        if ~isempty(handles)
```

```
% get size of stored values
[m, n] = size(handles.values);
if (handles.time > n)
stop(handles.timer);
end
% get current index of stored values
tym = handles.time + 1;
% load values joint positions, joint angles and torque values
ax = handles.values([1,2,3,4,5,6], tym);
xAll = handles.values([7,8,9,10,11,12,13,14], tym);
xAllr = handles.values([15,16,17,18,19,20,21,22], tym);
  ka = handles.values(23, tym);
kt = handles.values(24, tym);
ha = handles.values(25, tym);
ht = handles.values(26, tym);
t = handles.values(27, tym);

% plot only 10 most recent ankle position
if (handles.curr > 10)
xpos = get(handles.Pos(2,1),'XData');
ypos = get(handles.Pos(2,1),'YData');
xdata = [xpos(end - 10:end), ax(1,2)];
ydata = [ypos(end - 10:end), ax(2,2)];
set(handles.Pos(2,1),'XData',xdata,'YData',ydata);
else
xpos = get(handles.Pos(2,1),'XData');
ypos = get(handles.Pos(2,1),'YData');
xdata = [xpos(end - handles.limit:end), ax(1,2)];
ydata = [ypos(end - handles.limit:end), ax(2,2)];
set(handles.Pos(2,1),'XData',xdata,'YData',ydata);
end

% plot only handles.limit recent values
if (handles.curr > handles.limit)

xdata = xAll([1 3 5 7]);
ydata = xAll([2 4 6 8]);
set(handles.sim(1,1),'XData',xdata,'YData',ydata);
xdata = xAll([1 3 5 7]);
```

```
ydata = xAll([2 4 6 8]);
set(handles.sim(2,1),'XData',xdata,'YData',ydata);

xdata = xAllr([1 3 5 7]);
ydata = xAllr([2 4 6 8]);
set(handles.sim(3,1),'XData',xdata,'YData',ydata);
xdata = xAllr([1 3 5 7]);
ydata = xAllr([2 4 6 8]);
set(handles.sim(4,1),'XData',xdata,'YData',ydata);


set(handles.Htext1, 'string', handles.HTstr1, 'position', xAll(1:2) + [0.03,0]);
set(handles.Htext2, 'string', handles.HTstr2, 'position', xAll(3:4)+ [0.03,0]);
set(handles.Htext3, 'string', handles.HTstr3, 'position', xAll(5:6));


xpos = get(handles.Hka,'XData');
ypos = get(handles.Hka,'YData');
xdata = [xpos(end - handles.limit:end) t];
ydata = [ypos(end - handles.limit:end) ka];
set(handles.Hka,'XData',xdata,'YData',ydata);


xpos = get(handles.Hkt,'XData');
ypos = get(handles.Hkt,'YData');
xdata = [xpos(end - handles.limit:end) t];
ydata = [ypos(end - handles.limit:end) kt];
set(handles.Hkt,'XData',xdata,'YData',ydata);


xpos = get(handles.Hha,'XData');
ypos = get(handles.Hha,'YData');
xdata = [xpos(end - handles.limit:end) t];
ydata = [ypos(end - handles.limit:end) ha];
set(handles.Hha,'XData',xdata,'YData',ydata);


xpos = get(handles.Hht,'XData');
ypos = get(handles.Hht,'YData');
```

```
xdata = [xpos(end - handles.limit:end) t];
ydata = [ypos(end - handles.limit:end) ht];
set(handles.Hht,'XData',xdata,'YData',ydata);
else
xdata = xAll([1 3 5 7]);
ydata = xAll([2 4 6 8]);
set(handles.sim(1,1),'XData',xdata,'YData',ydata);
xdata = xAll([1 3 5 7]);
ydata = xAll([2 4 6 8]);
set(handles.sim(2,1),'XData',xdata,'YData',ydata);


set(handles.Htext1, 'string', handles.HTstr1, 'position', xAll(1:2) + [0.03,0]);
set(handles.Htext2, 'string', handles.HTstr2, 'position', xAll(3:4)+ [0.03,0]);
set(handles.Htext3, 'string', handles.HTstr3, 'position', xAll(5:6));


xdata = [get(handles.Hka,'XData') t];
ydata = [get(handles.Hka,'YData') ka];
set(handles.Hka,'XData',xdata,'YData',ydata);

xdata = [get(handles.Hkt,'XData') t];
ydata = [get(handles.Hkt,'YData') kt];
set(handles.Hkt,'XData',xdata,'YData',ydata);

xdata = [get(handles.Hha,'XData') t];
ydata = [get(handles.Hha,'YData') ha];
set(handles.Hha,'XData',xdata,'YData',ydata);

xdata = [get(handles.Hht,'XData') t];
ydata = [get(handles.Hht,'YData') ht];
set(handles.Hht,'XData',xdata,'YData',ydata);
end

handles.time = tym;

% update angle and torque display values
 set(handles.text4, 'String', num2str(ka));
set(handles.text6, 'String', num2str(ht));
```

```
        set(handles.text7, 'String', num2str(kt));
        set(handles.text8, 'String', num2str(ha));
        guidata(hObject,handles);
        end
end

function [ir, e] = err(q, qr)
```

function that iterates through all the points in the reference gait set and picks the closest matching gait

Inputs q: joint positions, qr: reference gait inputs

```
err = inf;

% check for well formatted values
if isnan(q(1))
        ir = NaN;
        e = NaN;
        return
end

% Iterate through a set of values and find closes matching index
for k = 1:length(qr)
        temp = (q(1)-qr(k,1))^2 + (q(2) - qr(k,2))^2;
        if temp < err
        err = temp;
        ir = k;
        end
end

err = abs(q-qr(ir,:))-3;

for i = 1:length(err)
        if err(i) < 0
        err(i) = 0;
        end
end
```

```
e = 0.1*err.^2;
```

forward kinematics of exoskeleton

angle flexion is considered positive zero pose is thigh, shank perpendicular to ground, foot parallel to ground.
inputs - q = 3x1 joint angle input L = 3xl link lengths (hip-knee, knee-ankle, ankle-foot)
outputs - X = 2x3 knee, ankle, and end of foot positions

```
function X = fk(q)

L = [0.35; 0.38;0];

X = zeros(2,3);
X(:,1) = L(1)*[sin(q(1)); -cos(q(1))];
theta2 = q(1)-q(2);
X(:,2) = X(:,1) + L(2)*[sin(theta2); -cos(theta2)];
theta3 = theta2 + pi/2 + q(3);
X(:,3) = X(:,2) + L(3)*[sin(theta3); -cos(theta3)];
```

```
% q = 3x1 joint angle input (hip, knee, angle)
% L = 3xl link lengths (hip-knee, knee-ankle, ankle-foot)
%output x 4x2 hip, knee, angle and foot positions
function pos=fkNew(q)
L = [1;1;1];
pos=zeros(4,2);
posPoly=zeros(5,2);
A1=[cos(q(1)+pi/2) -sin(q(1)+pi/2) 0 -L(1)*cos(q(1)+pi/2);
        sin(q(1)+pi/2) cos(q(1)+pi/2) 0 -L(1)*sin(q(1)+pi/2);
        0 0 1 0;
        0 0 0 1];
A2=[cos(q(2)) -sin(q(2)) 0 -L(2)*cos(q(2));
        sin(q(2)) cos(q(2)) 0 -L(2)*sin(q(2));
        0 0 1 0;
        0 0 0 1];
```

```
A3=[cos(q(3)-pi/2) -sin(q(3)-pi/2) 0 L(3)*cos(q(3)-pi/2);
        sin(q(3)-pi/2) cos(q(3)-pi/2) 0 L(3)*sin(q(3)-pi/2);
        0 0 1 0;
        0 0 0 1];
knee=(A1*[0;0;0;1])';
angle=(A1*A2*[0;0;0;1])';
foot=(A1*A2*A3*[0;0;0;1])';
pos(2,:)=knee(1:2);
pos(3,:)=angle(1:2);
pos(4,:)=foot(1:2);
%grid on
q(2)=-q(2);
polyD=0.03175;
A1P=[cos(q(1)+pi/2) -sin(q(1)+pi/2) 0 -(L(1)-polyD/2)*cos(q(1)+pi/2);
        sin(q(1)+pi/2) cos(q(1)+pi/2) 0 -(L(1)-polyD/2)*sin(q(1)+pi/2);
        0 0 1 0;
        0 0 0 1;];
A2P=[cos(q(2)/2) -sin(q(2)/2) 0 -polyD*cos(q(2)/2);
        sin(q(2)/2) cos(q(2)/2) 0 -polyD*sin(q(2)/2);
        0 0 1 0;
        0 0 0 1;];
A3P=[cos(q(2)/2) -sin(q(2)/2) 0 -(L(2)-polyD/2)*cos(q(2)/2);
        sin(q(2)/2) cos(q(2)/2) 0 -(L(2)-polyD/2)*sin(q(2)/2);
        0 0 1 0;
        0 0 0 1;];
A4P=[cos(q(3)-pi/2) -sin(q(3)-pi/2) 0 L(3)*cos(q(3)-pi/2);
        sin(q(3)-pi/2) cos(q(3)-pi/2) 0 L(3)*sin(q(3)-pi/2);
        0 0 1 0;
        0 0 0 1;];
PolyPos2=(A1P*[0;0;0;1])';
PolyPos3=(A1P*A2P*[0;0;0;1])';
PolyPos4=(A1P*A2P*A3P*[0;0;0;1])';
PolyPos5=(A1P*A2P*A3P*A4P*[0;0;0;1])';
posPoly(2,:)=PolyPos2(1:2);
posPoly(3,:)=PolyPos3(1:2);
posPoly(4,:)=PolyPos4(1:2);
posPoly(5,:)=PolyPos5(1:2);
```

```
function varargout = LiveSimulationPage(varargin)
% LIVESIMULATIONPAGE MATLAB code for LiveSimulationPage.fig
%      LIVESIMULATIONPAGE, by itself, creates a new LIVESIMULATIONPAGE or raises the existing
%      singleton*.
%
%      H = LIVESIMULATIONPAGE returns the handle to a new LIVESIMULATIONPAGE or the handle to
%      the existing singleton*.
%
%      LIVESIMULATIONPAGE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LIVESIMULATIONPAGE.M with the given input arguments.
%
%      LIVESIMULATIONPAGE('Property','Value',...) creates a new LIVESIMULATIONPAGE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before LiveSimulationPage_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to LiveSimulationPage_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help LiveSimulationPage

% Last Modified by GUIDE v2.5 31-Mar-2018 18:44:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @LiveSimulationPage_OpeningFcn, ...
                   'gui_OutputFcn',  @LiveSimulationPage_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before LiveSimulationPage is made visible.
function LiveSimulationPage_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to LiveSimulationPage (see VARARGIN)

% Choose default command line output for LiveSimulationPage
handles.output = hObject;


% UIWAIT makes LiveSimulationPage wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = LiveSimulationPage_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Create handles that deal with simulation characteristics
handles.output = hObject;
handles.kas = [];
handles.has = [];
```

```
handles.kts = [];
handles.hts = [];
handles.xAlls = zeros(8, 1);
handles.axs = zeros(6, 1);
handles.ts = [];
handles.curr = 0;
handles.limit = 25;
handles.ref = deg2rad(qrl);
handles.errbound = 20;
temp = load('gong');
handles.sound = temp.y;
handles.Fs = temp.Fs;
handles.errthresh = 20;
handles.thickness = 4;
handles.errcounthip = 0;
handles.errcountknee = 0;
handles.errhip = 30;
handles.errknee = 30;


%Load up reference gait

xh = xlsread('Winter_Appendix_data.xlsx','A1.Raw_Coordinate', 'E4:F109')/100;
xa = xlsread('Winter_Appendix_data.xlsx','A1.Raw_Coordinate', 'K4:L109')/100;
x = xa-xh;
handles.timer = timer('Name','MyTimer',          ...
   'Period',0.001,               ...
          'StartDelay',0,                  ...
          'TasksToExecute',inf,         ...
          'ExecutionMode','fixedSpacing', ...
          'TimerFcn',{@timerCallback,handles.figure1, hObject});
% or: jSlider.setPaintLabels(1);  % with both ticks and labels

% Set up serial connection
handles.arduino = setupSerial('/dev/tty.DSDTECHHC-05-DevB');

%Set up plot for for ankle position with reference gait
handles.t = 0;
axes(handles.axes2);
```

```
handles.Pos = plot(x(:,1), x(:,2), 'or', NaN, NaN, 'ob', 'markers', 8);
handles.time = 0;
legend('Reference', 'Actual');
title('Gait Characterization', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Ankle X Position (m)','FontSize',25,'Color','[0, 0.45, 0.74]');
ylabel('Ankle Y Position (m)','FontSize',25,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

%Set up plot of leg simulation with reference gait
axes(handles.axes7);
handles.sim = plot(NaN, NaN, '-b', NaN, NaN, '.b', NaN, NaN, '-r', NaN, NaN,
'.r','markers', 60);
xlim([-0.45, 0.55]);
ylim([-.9, 0.2]);
handles.Htext1 = text(handles.axes7, 0, 0, '');
handles.Htext2 = text(handles.axes7, 0, 0, '');
handles.Htext3 = text(handles.axes7, 0, 0, '');
axis equal
legend('Actual', '','Reference','');
title('Leg Simulation', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('X Relative to Hip Position','FontSize',25,'Color','[0, 0.45, 0.74]');
ylabel('Y Relative to Hip Position','FontSize',25,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);


% Set up plot of Knee angle vs time
axes(handles.axes3);
handles.Hka = plot(NaN, NaN);
%ylim([-2, 2]);
title('Knee Angle', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',25,'Color','[0, 0.45, 0.74]');
ylabel('Knee Angle (rad)','FontSize',25,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% Set up plot of Knee Torque vs time
axes(handles.axes4);
handles.Hkt = plot(NaN, NaN);
title('Knee Torque', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',25,'Color','[0, 0.45, 0.74]');
```

```
ylabel('Knee Torque (Nm)','FontSize',25,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% Set up plot of Hip angle vs time
axes(handles.axes6);
handles.Hha = plot(NaN, NaN);
title('Hip Angle', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',25,'Color','[0, 0.45, 0.74]');
ylabel('Hip Angle (rad)','FontSize',25,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% Set up plot og Hip torque vs time
axes(handles.axes5);
handles.Hht = plot(NaN, NaN);
title('Hip Torque', 'FontWeight','bold', 'FontSize',30, 'Color','[0, 0.45, 0.74]');
xlabel('Time (s)','FontSize',25,'Color','[0, 0.45, 0.74]');
ylabel('Hip Torque (Nm)','FontSize',25,'Color','[0, 0.45, 0.74]');
set(findall(gca, 'Type', 'Line'),'LineWidth',handles.thickness);

% Torque slider implelemntation
handles.jSlider = javax.swing.JSlider;
[handles.hObjava, handles.hslider] =javacomponent(handles.jSlider);
set(handles.hslider, 'Parent', handles.figure1);
set(handles.hslider,'units','normalized','position', [0.845,0.92,0.11,0.051]);
set(handles.jSlider,...
        'MajorTickSpacing',20,...
        'Value',50, ...
        'PaintTicks',true,...
        'PaintLabels',true...
        );
handles.hjSlider = handle(handles.jSlider, 'CallbackProperties');
set(handles.hjSlider, 'StateChangedCallback', {@slider1_Callback,handles.figure1});
guidata(hObject,handles);
setappdata(0, 'slider', handles.jSlider)
setappdata(0, 'arduino', handles.arduino)
start(handles.timer);
varargout{1} = handles.output;
guidata(hObject, handles);
```

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles, p)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Obtain value from slider and send to  arduino
slider = getappdata(0, 'slider');
arduino = getappdata(0, 'arduino');
value = slider.getValue;
fprintf(arduino, '%c', 'k');
fprintf(arduino, '%f', (value/100)*0.2);


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Send set zero signal to arduino.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

fprintf(handles.arduino, '%c', 'r');

% --- Save data to file.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
stop(handles.timer);
data = [handles.axs(1, :); handles.axs(2, :); handles.axs(3, :); handles.axs(4, :);...
        handles.axs(5, :); handles.axs(6, :); handles.xAlls(1, :); handles.xAlls(2, :);...
        handles.xAlls(3, :); handles.xAlls(4, :); handles.xAlls(5, :); handles.xAlls(6, :);...
        handles.xAlls(7, :); handles.xAlls(8, :);...
        handles.xAllrs(1, :); handles.xAllrs(2, :);...
        handles.xAllrs(3, :); handles.xAllrs(4, :); handles.xAllrs(5, :); handles.xAllrs(6,
:);...
        handles.xAllrs(7, :); handles.xAllrs(8, :);handles.kas; handles.kts; handles.has;
handles.hts....
        ; handles.ts];
p_id = getappdata(0, 'p_id');
d = datetime('now');
d_str = datestr(d, 30);
filename = [p_id,'-', d_str, '.csv'];
csvwrite(filename, data);
fclose(handles.arduino);
guidata(hObject,handles);
close all;


% --- Exit simulation page.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
stop(handles.timer);
fclose(handles.arduino);
guidata(hObject,handles);
close all;


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over pushbutton2.
function pushbutton2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)



function varargout = LoginPage(varargin)
% LOGINPAGE MATLAB code for LoginPage.fig
%      LOGINPAGE, by itself, creates a new LOGINPAGE or raises the existing
%      singleton*.
%
%      H = LOGINPAGE returns the handle to a new LOGINPAGE or the handle to
%      the existing singleton*.
%
%      LOGINPAGE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LOGINPAGE.M with the given input arguments.
%
%      LOGINPAGE('Property','Value',...) creates a new LOGINPAGE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before LoginPage_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to LoginPage_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help LoginPage

% Last Modified by GUIDE v2.5 05-Apr-2018 16:07:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @LoginPage_OpeningFcn, ...
                   'gui_OutputFcn',  @LoginPage_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before LoginPage is made visible.
function LoginPage_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to LoginPage (see VARARGIN)

% Choose default command line output for LoginPage
handles.output = hObject;
% This creates the 'background' axes
ha = axes('units','normalized','position',[0, 0 1 1]);
% Move the background axes to the bottom
I=imread('LoginPage.png');
hi = imagesc(I);
uistack(ha,'bottom');
set(ha,'handlevisibility','off','visible','off');
% Load in a background image and display it using the correct colors
% The image used below, is in the Image Processing Toolbox.  If you do not have
%access to this toolbox, you can use another image file instead.
% Turn the handlevisibility off so that we don't inadvertently plot into the axes again
% Also, make the axes invisible
set(ha,'handlevisibility','off', ...
        'visible','off')
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes LoginPage wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = LoginPage_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
% This creates the 'background' axes
ha = axes('units','normalized', ...
          'position',[0 0 1 1]);

% Move the background axes to the bottom
uistack(ha,'bottom');
% Load in a background image and display it using the correct colors
% The image used below, is in the Image Processing Toolbox.  If you do not have
%access to this toolbox, you can use another image file instead.
I=imread('LoginPage.png');
hi = imagesc(I);
colormap gray
% Turn the handlevisibility off so that we don't inadvertently plot into the axes again
% Also, make the axes invisible
set(ha,'handlevisibility','off', ...
          'visible','off');



axes(handles.axes1);
set(gca,'xtick',[]);
set(gca,'ytick',[]);

% Logo
% The image used below, is in the Image Processing Toolbox.  If you do not have
%access to this toolbox, you can use another image file instead.
I=imread('logo.png');
hi = imshow(I);
```

```
set(ha,'handlevisibility','off', ...
        'visible','off');
% Turn the handlevisibility off so that we don't inadvertently plot into the axes again
% Also, make the axes invisible
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
MainMenu



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
          set(hObject,'BackgroundColor','white');
end


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over pushbutton1.
function pushbutton1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end

function varargout = MainMenu(varargin)
% MAINMENU MATLAB code for MainMenu.fig
%      MAINMENU, by itself, creates a new MAINMENU or raises the existing
%      singleton*.
%
%      H = MAINMENU returns the handle to a new MAINMENU or the handle to
%      the existing singleton*.
%
%      MAINMENU('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAINMENU.M with the given input arguments.
%
%      MAINMENU('Property','Value',...) creates a new MAINMENU or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MainMenu_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MainMenu_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help MainMenu

% Last Modified by GUIDE v2.5 04-Apr-2018 12:35:23

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MainMenu_OpeningFcn, ...
                   'gui_OutputFcn',  @MainMenu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MainMenu is made visible.
function MainMenu_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MainMenu (see VARARGIN)

% Choose default command line output for MainMenu
handles.output = hObject;
handles.output = hObject;
% This creates the 'background' axes
ha = axes('units','normalized','position',[0, 0 1 1]);
% Move the background axes to the bottom
I=imread('MainMenu.png');
```

```
hi = imagesc(I);
uistack(ha,'bottom');
set(ha,'handlevisibility','off','visible','off');


axes(handles.axes1);
set(gca,'xtick',[]);
set(gca,'ytick',[]);

% Logo
% The image used below, is in the Image Processing Toolbox.  If you do not have
%access to this toolbox, you can use another image file instead.
I=imread('logo.png');
hi = imshow(I);
set(ha,'handlevisibility','off', ...
        'visible','off');

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MainMenu wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MainMenu_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
p_id = get(handles.edit1,'String');
setappdata(0,'p_id',p_id);
LiveSimulationPage

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
direc = pwd;
files = dir(direc);
list = {};
p_id = get(handles.edit1,'String');
setappdata(0,'p_id',p_id);
for i = 1:length(files)
        f = files(i, 1);
        if strcmp(f.name, '.') || strcmp(f.name, '..')
        continue
        end
        datafiles = strsplit(f.name, '-');
        if (strcmp(datafiles{1}, p_id))
        value = datafiles{2};
        value = string(value(1:end-4));
        list{end+1} = value;
        end
end
set(handles.popupmenu1, 'String', list);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all;
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
          set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over popupmenu1.
function popupmenu1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
contents = get(handles.popupmenu1,'String');
ts = contents{get(handles.popupmenu1,'Value')};
setappdata(0,'ts',ts);
DeadSimulationPage;

function qr = qrl()

% joint angles (theta, omega, alpha: ankle, knee, hip)
dataq = xlsread('Winter_Appendix_data.xlsx','A4.RelJointAngularKinematics',
'D5:L110');
q = dataq(:,[7 4]); % degrees to radians (hip, knee, ankle)

r = 22:length(dataq)-15;
q = q(r,:);
q_new = q(1,:);
for i = 2:length(r)
```

```
        q_new = [q_new; (q(i-1,:)+q(i,:))/2; q(i,:)];
end
qr = q_new;


function[obj] = setupSerial(comPort)
% Initialize Serial object
delete(instrfindall);
obj = serial(comPort);
flushinput(obj);
set(obj,'DataBits',8);
set(obj,'StopBits',1);
set(obj,'BaudRate',9600);
set(obj,'Parity','none');
fopen(obj);
end
```

A7: Arduino code

```
#include <SPI.h>
#include "VarSpeedServo.h"
#include "MatrixMath.h"

// NOTES:
// Program will not run to completion if encoders are not attached properly.

// UPDATE
#define NREF 139
#define BETA 0.8 //0 = filter is off
#define T_OFFSET_K 1.87
#define T_SENSITIVITY_K 7.1
#define T_OFFSET_H 1.85
#define T_SENSITIVITY_H 6.667
#define KP 40
#define KD 0.0015
#define ERR_RANGE 3 // how far off from reference is "correct" in degrees
```

```
#define PHIK_M 35
#define PHIK_B 1900
#define PHIH_M 35
#define PHIH_B 1900
#define MIN_PHI 1700
#define MAX_PHI 2250

// PINS
#define KNEE_ANGLE 3 //Chip or Slave select
#define HIP_ANGLE 4 //Chip or Slave select
#define KNEE_TORQUE A0
#define HIP_TORQUE A1
#define SERVO_KNEE 5
#define SERVO_HIP 6
#define ZERO_RESET 7

// Other
#define SPEED 160
#define CONV_D2R M_PI/180
#define BEGIN_SEND -10000.0
#define MAX_TORQUE 10
#define WRAP_THRESH 45
#define WRAP 90

VarSpeedServo ServoH;
VarSpeedServo ServoK;

uint16_t ABSposition = 0;
uint16_t ABSposition_last = 0;
uint8_t temp[2];
float k = 0.05; // can be updates

// State:
// 0: knee angle
// 1: hip angle
// 2: knee torque
// 3: hip torque
float state[4];
float stateavg[4];
```

```
// reference gait ordered {x1, y1, x2, y2, x3, y3, ...}
float Ref[2*NREF] = {6.7,18.3,4.7,17.75,2.7,17.2,1.3,16.7,-0.1,16.2,-0.95,15.7,-1.8,
15.2,-2.05,14.75,-2.3,14.3,-2.05,13.95,-1.8,13.6,-1.2,13.2,-0.6,12.8,0.25,12.5,1.1,
12.2,2.05,11.95,3,11.7,4.05,11.55,5.1,11.4,6.25,11.3,7.4,11.2,8.6,11.25,9.8,11.3,10.95,
11.4,12.1,11.5,13.05,11.55,14,11.6,14.7,11.5,15.4,11.4,15.8,11.1,16.2,10.8,16.25,10.3,
16.3,9.8,16.1,9.2,15.9,8.6,15.55,7.9,15.2,7.2,14.7,6.5,14.2,5.8,13.7,5.15,13.2,4.5,
12.75,4,12.3,3.5,11.85,3.1,11.4,2.7,11.05,2.35,10.7,2,10.4,1.75,10.1,1.5,9.75,1.3,9.4,
1.1,9.1,0.95,8.8,0.8,8.5,0.6,8.2,0.4,7.9,0.2,7.6,0,7.3,-0.25,7,-0.5,6.7,-0.85,6.4,-1.2,
6.15,-1.5,5.9,-1.8,5.65,-2.15,5.4,-2.5,5.3,-2.8,5.2,-3.1,5.15,-3.4,5.1,-3.7,5.3,-3.95,
5.5,-4.2,5.85,-4.4,6.2,-4.6,6.75,-4.75,7.3,-4.9,8.1,-5.05,8.9,-5.2,9.9,-5.4,10.9,-5.6,
12.1,-5.75,13.3,-5.9,14.75,-6,16.2,-6.1,17.9,-6.15,19.6,-6.2,21.55,-6.2,23.5,-6.2,25.65,
-6.1,27.8,-6,30.1,-5.85,32.4,-5.7,34.9,-5.35,37.4,-5,39.95,-4.5,42.5,-4,45.05,-3.25,47.6,
-2.5,50,-1.5,52.4,-0.5,54.55,0.65,56.7,1.8,58.55,3.15,60.4,4.5,61.9,5.9,63.4,7.3,64.4,
8.75,65.4,10.2,65.95,11.55,66.5,12.9,66.55,14.1,66.6,15.3,66.15,16.3,65.7,17.3,64.8,18.1,
63.9,18.9,62.6,19.5,61.3,20.1,59.65,20.5,58,20.9,56.1,21.25,54.2,21.6,52.05,21.85,49.9,
22.1,47.55,22.25,45.2,22.4,42.6,22.5,40,22.6,37.25,22.55,34.5,22.5,31.55,22.35,28.6,22.
2,
25.6,21.9,22.6,21.6,19.55,21.15,16.5,20.7,13.7,20.15,10.9,19.6,8.35,18.95,5.8,18.3};
int i_err;
float qk_err;
float qh_err;
float qk_off;
float qh_off;

float t_err[4]; //knee, hip, knee', hip'
float ttemp;
float phik = MIN_PHI;
float phih = MIN_PHI;
float tk_des = 0;;
float th_des = 0;;

float deg = 0.00;
int wrap_k = 0;
int wrap_h = 0;
float ka;
float ha;

unsigned long time;
```

```
unsigned long temptime;
float T;
float To = 0;

unsigned long count = 0;
char incomingByte;

void setup()
{
 //For testing
 pinMode(LED_BUILTIN, OUTPUT);

 // encoder set up
 pinMode(KNEE_ANGLE,OUTPUT);//Slave Select
 digitalWrite(KNEE_ANGLE,HIGH);
 pinMode(HIP_ANGLE,OUTPUT);//Slave Select
 digitalWrite(HIP_ANGLE,HIGH);
 pinMode(ZERO_RESET,OUTPUT);//Slave Select
 digitalWrite(ZERO_RESET,HIGH);
 SPI.begin();
 SPI.setBitOrder(MSBFIRST);
 SPI.setDataMode(SPI_MODE0);
 SPI.setClockDivider(SPI_CLOCK_DIV16);

 //torque sensor set up
 pinMode(HIP_TORQUE, INPUT);
 digitalWrite(HIP_TORQUE, LOW);
 pinMode(KNEE_TORQUE, INPUT);
 digitalWrite(KNEE_TORQUE, LOW);

 // general set up
 Serial.begin(9600);

 Serial.flush();
 delay(2000);
 SPI.end();

 ServoH.attach(SERVO_HIP);
 ServoK.attach(SERVO_KNEE);
```

```
 for (int i = 0; i < 4; i++) {
  state[i] = 0;
  stateavg[i] = 0;
  t_err[i] = 0;
 }

 time = millis();
 T = time;

 phik = 0;
 phih = 0;
}

// Helper function for get_angle
uint8_t SPI_T (uint8_t msg, int joint)   //Repetive SPI transmit sequence
{
  uint8_t msg_temp = 0;  //vairable to hold recieved data
  digitalWrite(joint,LOW);    //select spi device
  msg_temp = SPI.transfer(msg);   //send and recieve
  digitalWrite(joint,HIGH);   //deselect spi device
  return(msg_temp);     //return recieved byte
}

// Return angle measurement from encoder. 0 < output < 90
float get_angle(int joint) {
 uint8_t recieved = 0xA5;   //just a temp vairable
  ABSposition = 0;   //reset position vairable

  SPI.begin();   //start transmition
  digitalWrite(joint,LOW);

  SPI_T(0x10, joint);   //issue read command
  recieved = SPI_T(0x00, joint);   //issue NOP to check if encoder is ready to send

  while (recieved != 0x10)   //loop while encoder is not ready to send
  {
   recieved = SPI_T(0x00, joint);
```

```
    //Serial.println(joint); //cleck again if encoder is still workin
    delayMicroseconds(20);    //wait a bit
  }

  temp[0] = SPI_T(0x00, joint);    //Recieve MSB
  temp[1] = SPI_T(0x00, joint);    // recieve LSB

  digitalWrite(joint,HIGH);  //just to make sure
  SPI.end();    //end transmition

  temp[0] &=~ 0xF0;    //mask out the first 4 bits

  ABSposition = temp[0] << 8;    //shift MSB to correct ABSposition in ABSposition
message
  ABSposition += temp[1];    // add LSB to ABSposition message to complete message

  if (ABSposition != ABSposition_last)    //if nothing has changed dont wast time
sending position
  {
    ABSposition_last = ABSposition;    //set last position to current position
    deg = ABSposition;
    deg = deg * 0.08789;    // aprox 360/4096
    return deg;
  }

}

// digitally recalibrate torque. Reset encoders to set new zero at current location
void setzero() {
 for (int i = 0; i < 2; i++) {
 SPI.begin();
 uint8_t signal = SPI_T(0x70, i + 3);    // set zero at set up
 while (signal != 0x80) {
   signal = SPI_T(0x00, i + 3);
   }
 SPI.end();    //end transmition

 }
 digitalWrite(ZERO_RESET,LOW);
```

```
  delay(50);
  digitalWrite(ZERO_RESET,HIGH);

  qk_off = qk_off + stateavg[2];
  qh_off = qh_off + stateavg[3];
}

float get_torque(int joint, float off, float sens) {
  float adc_output = analogRead(joint);
  float voltage = (5.0/1024)*adc_output;

  // based on torque calibration
  float torque = (voltage-off)*sens;

  return torque;
}

void send_values(float* values, int len) {
  // -10,000.0 used to define beginning of packet
  Serial.println(BEGIN_SEND);
  for (int i = 0; i < len; i++) {
    Serial.println(values[i]);
  }
}

void average(float* avg, float* curr, int len) {
  for(int i = 0; i < len; i++) {
    avg[i] = BETA*avg[i]+(1-BETA)*curr[i];
  }
}

// Returns min position error index based on reference gait
int err() {
  // joint space implementation
  float err;
  int i_err;
  int skip = 2;

  // Check every (1+skip) values in reference. Valid as long as:
```

```
  // dot_product((x(i+1) - x(i))/||x(i+1) - x(i)||, (x(i+1+skip) - x(i))/||x(i+1+skip) - x(i)||)
~ 1
  // skip = 2 valid for standard reference gaits with NREF ~ 139
  float error = sq(Ref[0]-stateavg[0]) + sq(Ref[1]-stateavg[1]);
  for(int i = 1+skip; i < NREF; i+=1+skip) {
    err = sq(Ref[2*i]-stateavg[0]) + sq(Ref[2*i+1]-stateavg[1]);
    if(err < error) {
      i_err = i;
      error = err;
    }
  }

  // After correct region is found, compare locally
  for(int i = i_err-skip; i <= i_err+skip; i++) {
    int j = i % NREF;
    err = sq(Ref[2*j]-stateavg[0]) + sq(Ref[2*j+1]-stateavg[1]);
    if(err < error) {
      i_err = j;
      error = err;
    }
  }
  return i_err;
}

// estimate phi based on brake characterization
float phi_est(float t, float m, float b) {
  float phi_des = m*abs(t) + b;

  // characterization is only valid above MIN_PHI. Estimates above MAX_PHI
  // are corrected in run_servo to simplify edge cases with controls
  if (phi_des < MIN_PHI) {
    return MIN_PHI;
  }
  return phi_des;
}

// correct servos. Never sends command above or below MAX_PHI or MIN_PHI
void run_servo() {
  if (phik < MIN_PHI) {
```

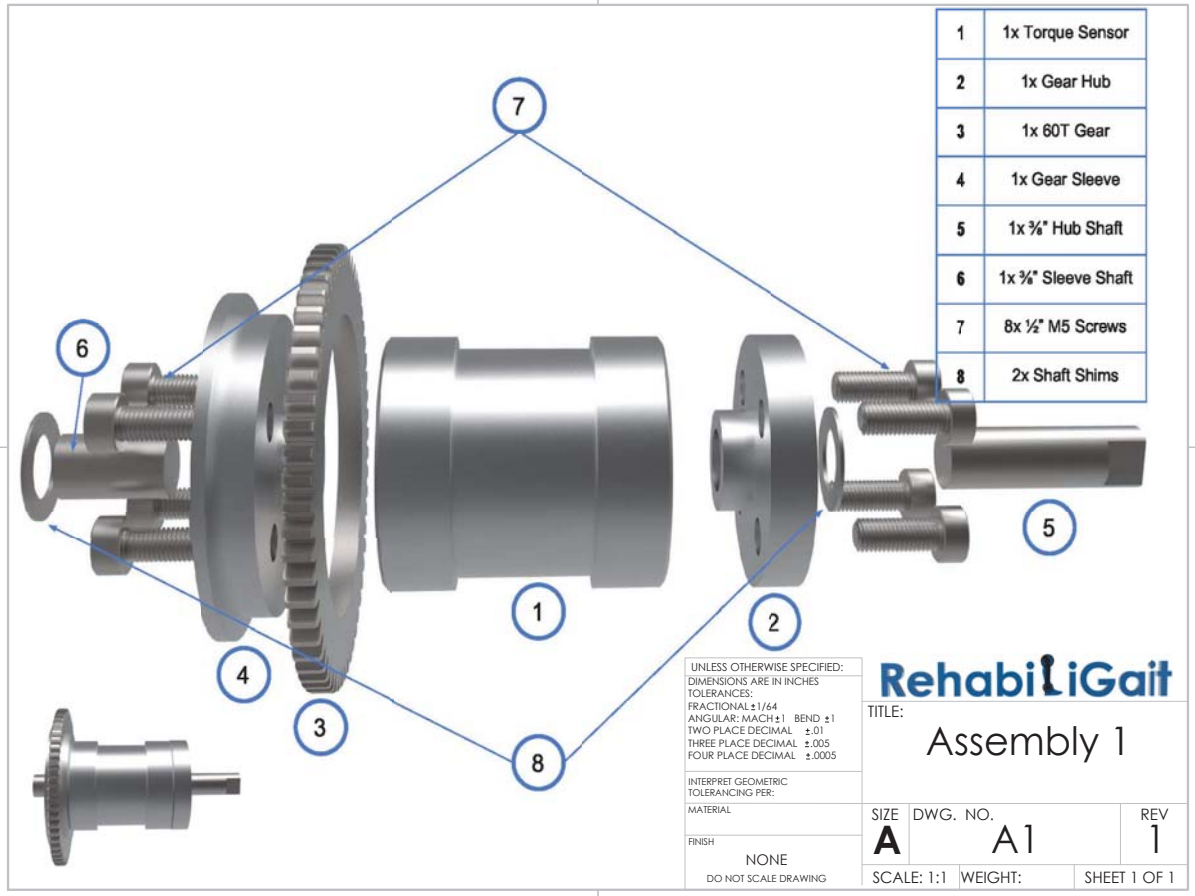
```
  ServoK.write(MIN_PHI,SPEED);
 } else if (phik > MAX_PHI) {
  ServoK.write(MAX_PHI,SPEED);
 } else {
  ServoK.write(phik, SPEED);
 }
 if (phih < MIN_PHI) {
  ServoH.write(MIN_PHI,SPEED);
 } else if (phih > MAX_PHI) {
  ServoH.write(MAX_PHI,SPEED);
 } else {
  ServoH.write(phih, SPEED);
 }
}

void loop()
{
    // store state in order to wrap angles
    ka = state[0];
    ha = state[1];

    // take measurementsM
    state[0] = (get_angle(KNEE_ANGLE)/4 + WRAP*wrap_k);
    state[2] = get_torque(KNEE_TORQUE, T_OFFSET_K,
T_SENSITIVITY_K)-qk_off;
    state[3] = get_torque(HIP_TORQUE, T_OFFSET_H, T_SENSITIVITY_H)-qh_off;
    state[1] = (get_angle(HIP_ANGLE)/4 + WRAP*wrap_h);

    // Wrap both knee and hip angles (hip and knee angles can only be directly
measured
    // 0-90 degrees. In order to see if wraparound occured I compare the angle change
    // to a threshold of 45 degrees. The code below ensures that output is total absolute
angle
    if (abs(ka-state[0]) > WRAP_THRESH) {
     if (ka > state[0]) {
      state[0] += WRAP;
      wrap_k +=1;
     }
     else {
```

```
  state[0] -= WRAP;
  wrap_k -= 1;
 }
}
if (abs(ha-state[1]) > WRAP_THRESH) {
 if (ha > state[1]) {
  state[1] += WRAP;
  wrap_h +=1;
 }
 else {
  state[1] -= WRAP;
  wrap_h -=1;
 }
}

// calculate T for derivatives
temptime = time;
time = micros(); // resets to 0 every ~40 min
// if micros() wraps, T doesn't update. Next iteration time>temptime and T updates
if (time > temptime) T = (time-temptime)/1000000.0;

average(stateavg, state, 4);

// compare to reference
i_err = err();

qk_err = abs(Ref[2*i_err]-stateavg[0]) - ERR_RANGE;
qh_err = abs(Ref[2*i_err+1]-stateavg[1]) - ERR_RANGE;
if (qk_err < 0) qk_err = 0;
if (qh_err < 0) qh_err = 0;

// controls
tk_des = k*sq(qk_err);
th_des = k*sq(qh_err);
if (tk_des > MAX_TORQUE) tk_des = MAX_TORQUE;
if (th_des > MAX_TORQUE) th_des = MAX_TORQUE;

ttemp = tk_des - stateavg[2];
t_err[2] = (t_err[0]-ttemp)/T;
```

```
t_err[0] = ttemp;
ttemp = th_des - stateavg[3];
t_err[3] = (t_err[1]-ttemp)/T;
t_err[1] = ttemp;

phik = phi_est(tk_des, PHIK_M, PHIK_B) + KP*t_err[0] + KD*t_err[2];
phih = phi_est(th_des, PHIK_M, PHIK_B) + KP*t_err[1] + KD*t_err[3];

// correct servos
run_servo();

// Every 20th iteration communicate with UI
if (count++ % 20 ==0) {
 send_values(stateavg, 4); // send tracked values

 // check if UI sending a command
 if (Serial.available() > 0) {
  delay(200);
       // read the incoming byte:
       incomingByte = Serial.read();

    // 'r' = reset: recalibrates encoders and torque sensors
    // Sets current torques and angles to zero through offset
    if (incomingByte == 'r') {
      setzero();
      ka = 0;
      ha = 0;
      wrap_k = 0;
      wrap_h = 0;
      state[0] = 0;
      state[1] = 0;
    }

    // Set torque application scalar
    if (incomingByte == 'k') {
      int ticker = 0;
      while (Serial.available() < 4 && ticker < 30) {
        continue;
        ticker++;
```

```
          }
          if (Serial.available() >= 4) {
            k = Serial.parseFloat();
          }
        }
      }
    }
}
```

A8. Engineering Drawings



1	1x Torque Sensor
2	1x Gear Hub
3	1x 60T Gear
4	1x Gear Sleeve
5	1x ⅜" Hub Shaft
6	1x ⅜" Sleeve Shaft
7	8x ½" M5 Screws
8	2x Shaft Shims

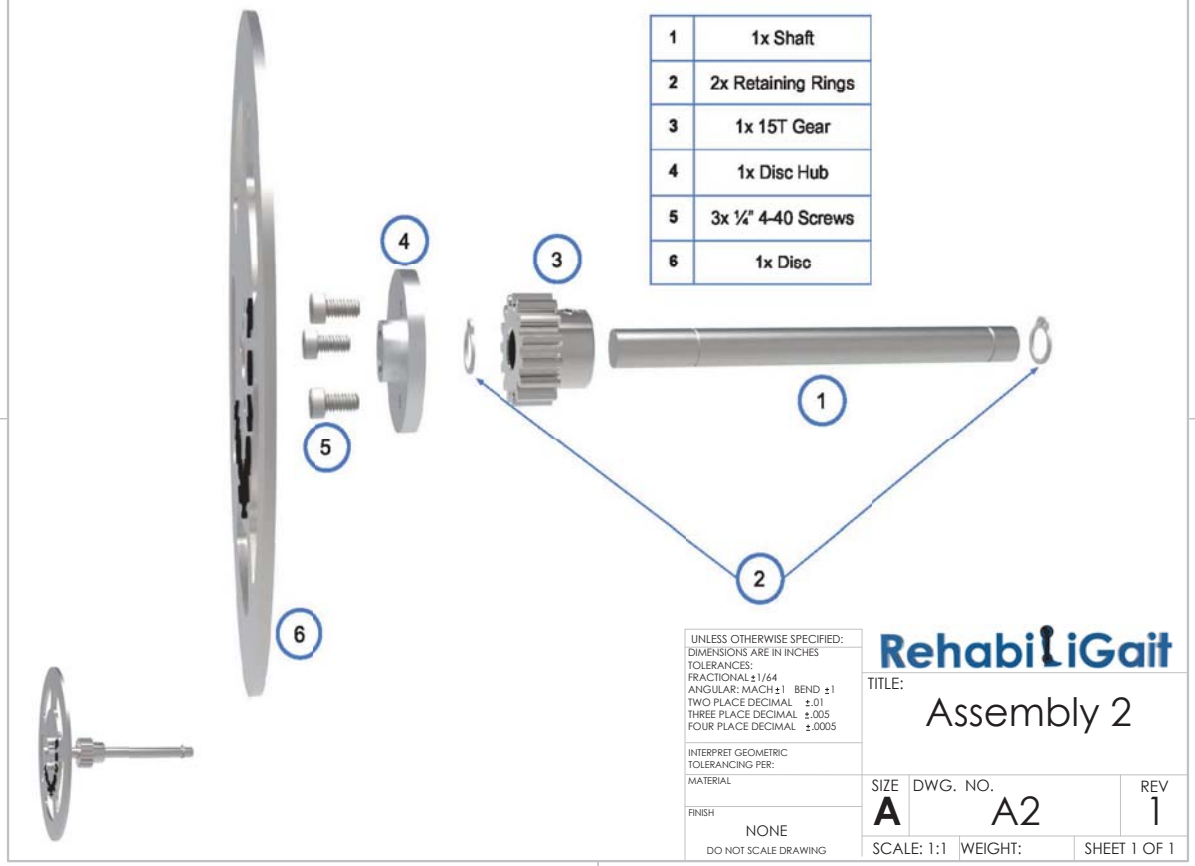
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NONE
DO NOT SCALE DRAWING

RehabiLiGait

TITLE: Assembly 1

SIZE	DWG. NO.	REV
A	A1	1

SCALE: 1:1 | WEIGHT: | SHEET 1 OF 1



1	1x Shaft
2	2x Retaining Rings
3	1x 15T Gear
4	1x Disc Hub
5	3x ¼" 4-40 Screws
6	1x Disc

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NONE
DO NOT SCALE DRAWING

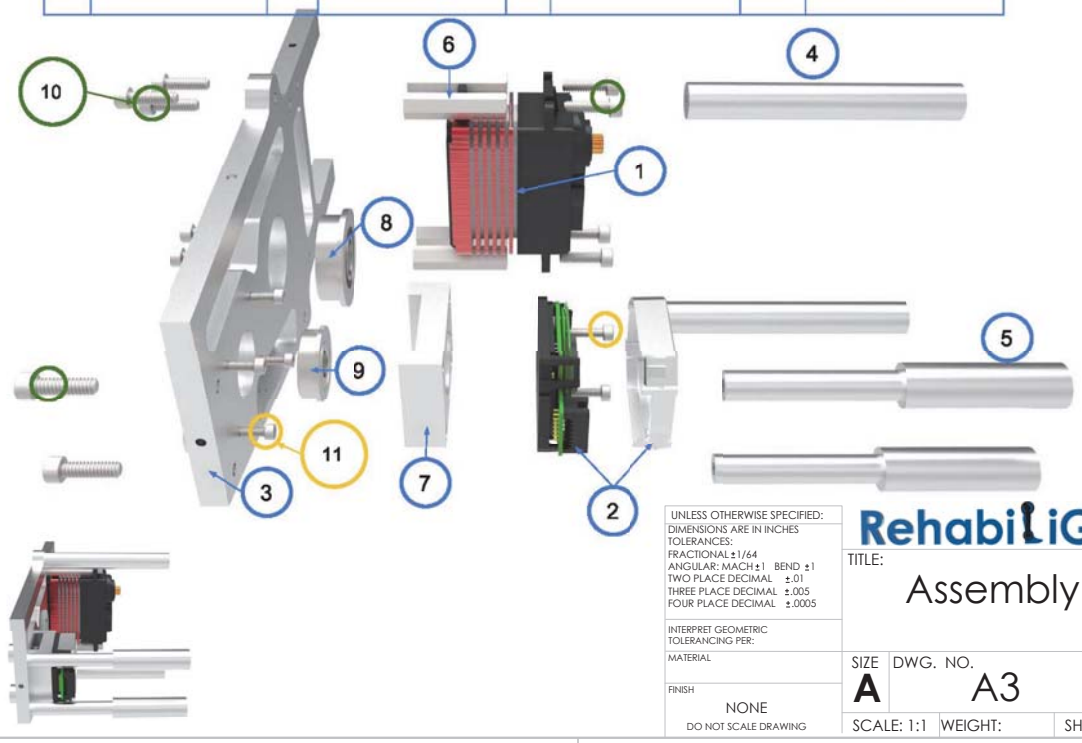
RehabiLiGait

TITLE: Assembly 2

SIZE	DWG. NO.	REV
A	A2	1

SCALE: 1:1 | WEIGHT: | SHEET 1 OF 1

1	1x Servo	4	2x Support Shafts	7	1x Encoder Bracket	10	12x ½" 6-32 Screws
2	1x Encoder	5	2x Stepped Shafts	8	1x ⅜"Bearing	11	6x ⅜" 2-56 Screws
3	1x Back Plate	6	4x 1" 6-32 Standoffs	9	1x ¼"Bearing		



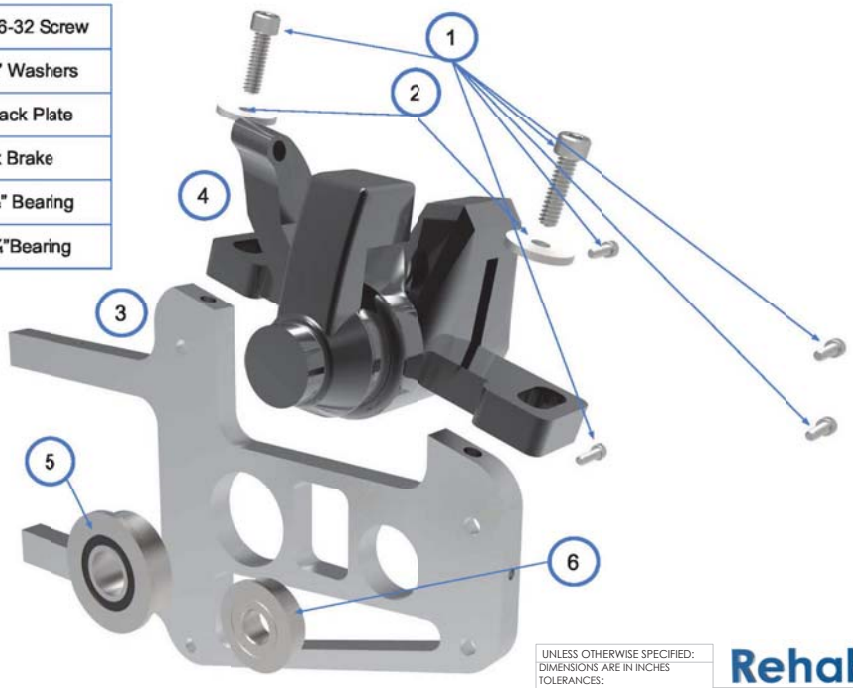
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NONE
DO NOT SCALE DRAWING

RehabiLiGait

TITLE: Assembly 3

SIZE	DWG. NO.	REV
A	A3	1
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

1	6x ¾" 6-32 Screw
2	2x ½" Washers
3	1x Back Plate
4	1x Brake
5	1x ⅜" Bearing
6	1x ¼"Bearing



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NONE
DO NOT SCALE DRAWING

RehabiLiGait

TITLE: Assembly 4

SIZE	DWG. NO.	REV
A	A4	1
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

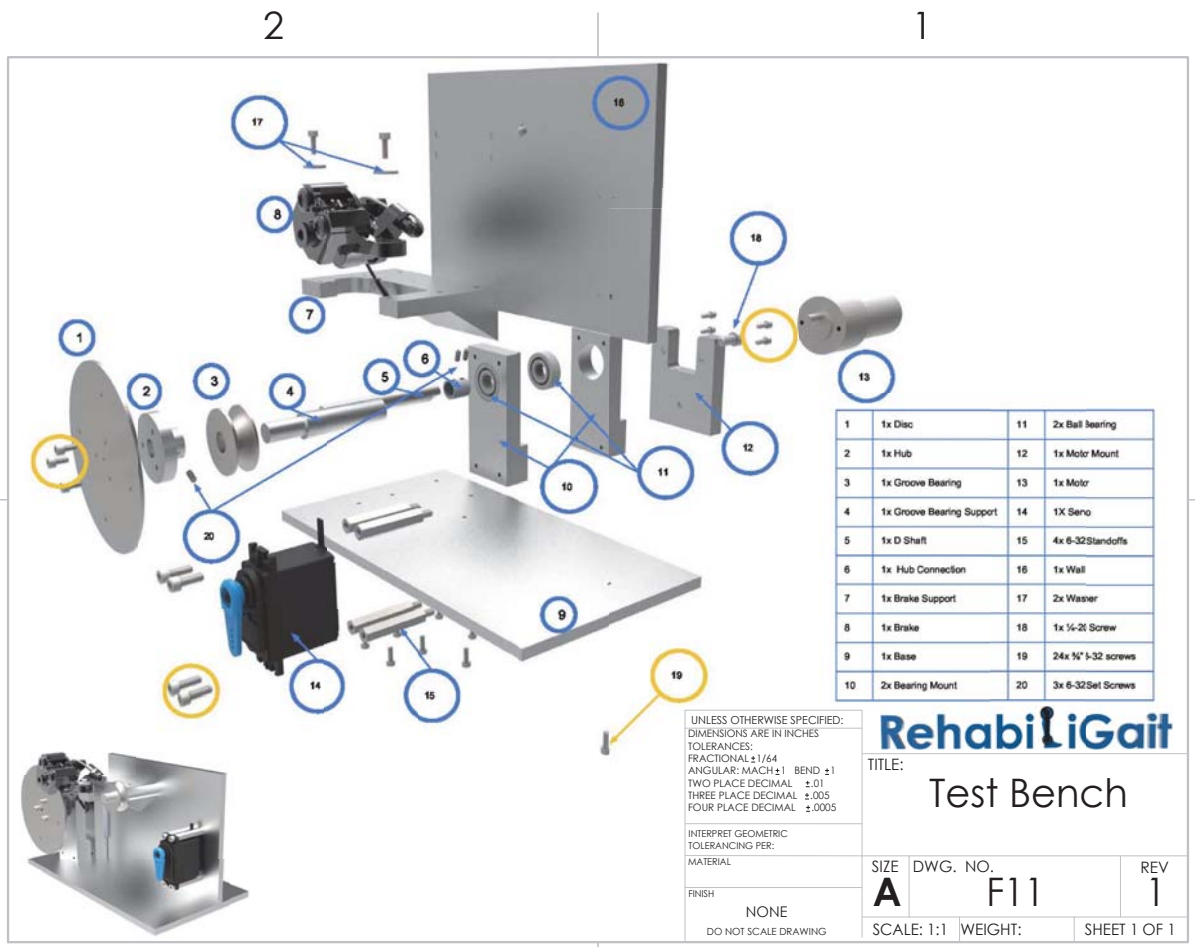
1	1x Resistive Subsystem (Assembly 1, 2, 3 & 4)
2	1x Pin Plate
3	1x Ball Bearing
4	1x Side Plate
5	2x Brake Hub
6	2x 6-32 Set Screw
7	4x 10-32 Screws
8	6x 6-32 Screws

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NONE
DO NOT SCALE DRAWING

RehabiLiGait

TITLE: Assembly 5

SIZE	DWG. NO.	REV
A	A5	1
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1



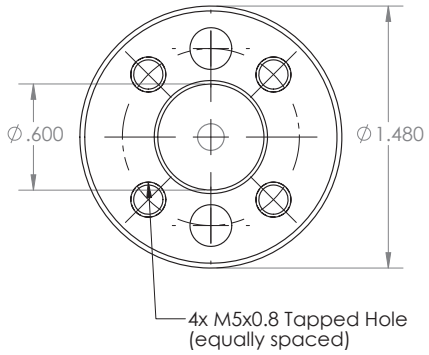
1	1x Disc	11	2x Ball bearing
2	1x Hub	12	1x Motor Mount
3	1x Groove Bearing	13	1x Motor
4	1x Groove Bearing Support	14	1X Servo
5	1x D Shaft	15	4x 6-32Standoffs
6	1x Hub Connection	16	1x Wall
7	1x Brake Support	17	2x Washer
8	1x Brake	18	1x ¼-20 Screw
9	1x Base	19	24x ¾" 8-32 screws
10	2x Bearing Mount	20	3x 6-32Set Screws

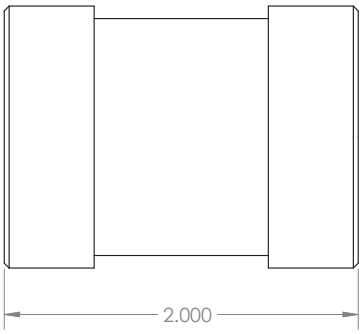
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NONE
DO NOT SCALE DRAWING

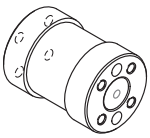
RehabiLiGait

TITLE: Test Bench

SIZE	DWG. NO.	REV
A	F11	1
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

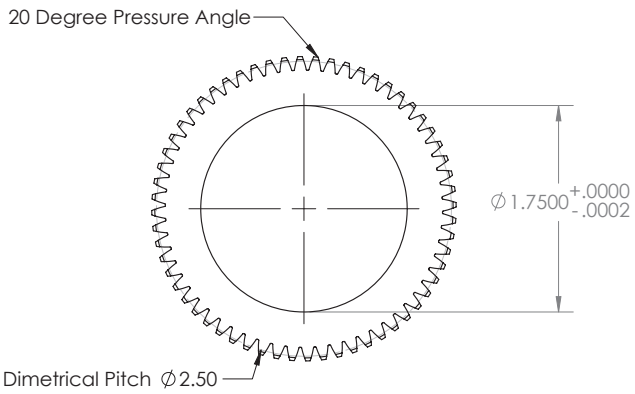


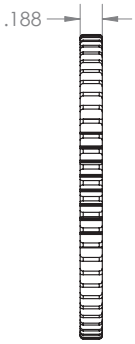


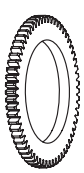


Loadstar SENSORS
Load Sensing Made Easy

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	TITLE: Torque Sensor
INTERPRET GEOMETRIC TOLERANCING PER:	
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. / REV
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:1 / WEIGHT: / SHEET 1 OF 1

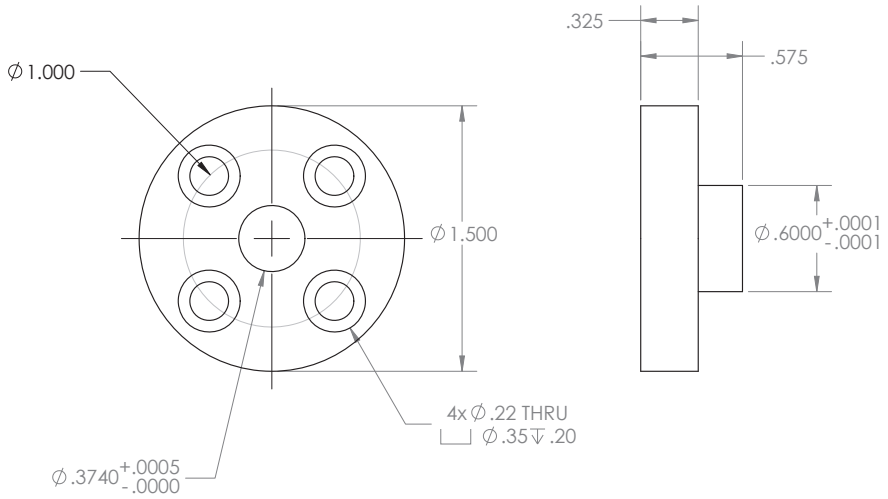


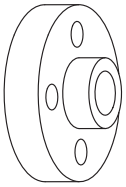




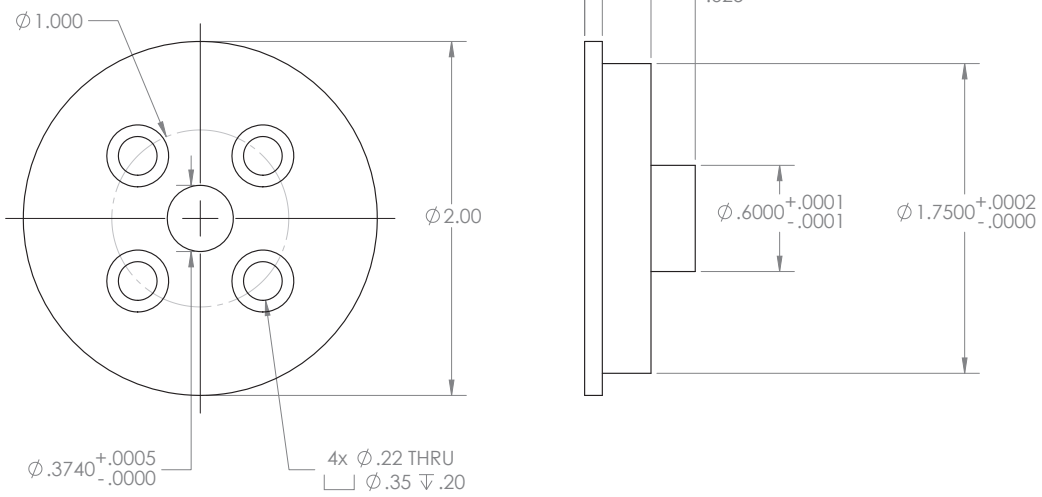
RehabiLiGait

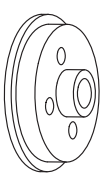
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	TITLE: Gear (60T)
INTERPRET GEOMETRIC TOLERANCING PER:	
MATERIAL 303 Stainless Steel	SIZE A / DWG. NO. P1-2 / REV 0
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:1 / WEIGHT: / SHEET 1 OF 1



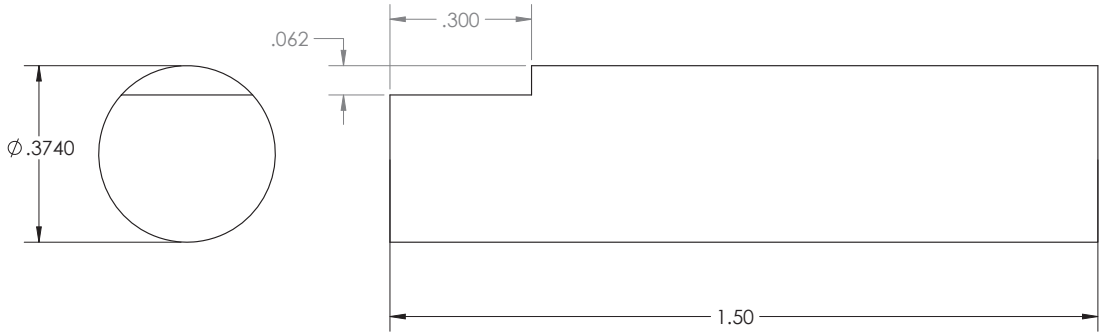


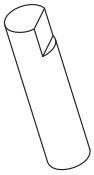
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait	
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Gear Hub	
MATERIAL 6061-T6 Aluminum	SIZE A	DWG. NO. P1-3 / REV 1
FINISH NONE	SCALE: 1:1	WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING		



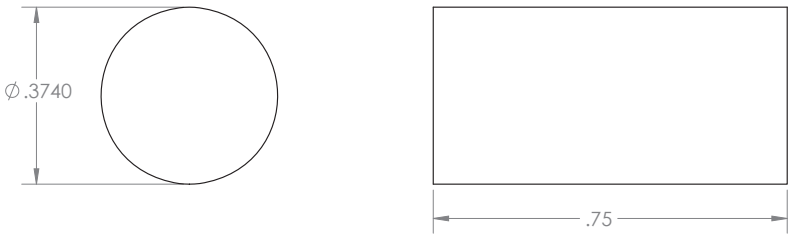


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait	
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Gear Sleeve	
MATERIAL 6061-T6 Aluminum	SIZE A	DWG. NO. P1-4 / REV 0
FINISH NONE	SCALE: 1:1	WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING		

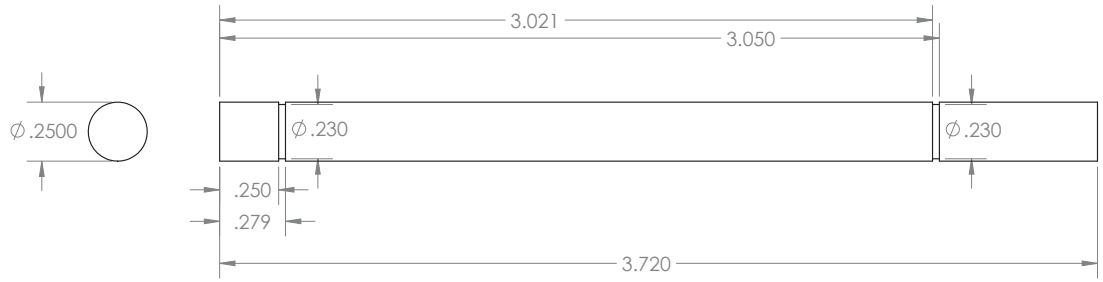




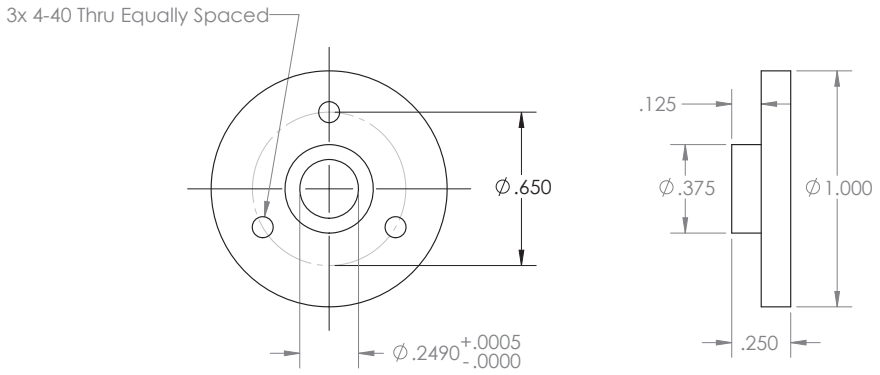
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Gear Hub Shaft
MATERIAL 303 Stainless Steel	SIZE A / DWG. NO. P1-5 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 2:1 / WEIGHT: / SHEET 1 OF 1

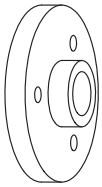


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Gear Sleeve Shaft
MATERIAL 303 Stainless Steel	SIZE A / DWG. NO. P1-6 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 4:1 / WEIGHT: / SHEET 1 OF 1

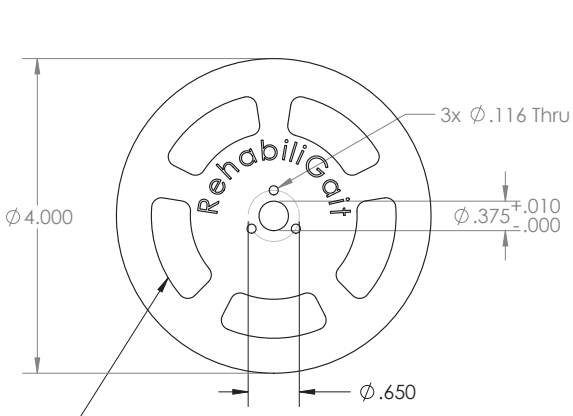


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005 INTERPRET GEOMETRIC TOLERANCING PER: MATERIAL: 303 Stainless Steel FINISH: NONE DO NOT SCALE DRAWING	RehabiLiGait		
	TITLE: Shaft		
	SIZE A	DWG. NO. P2-2	REV 1
	SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

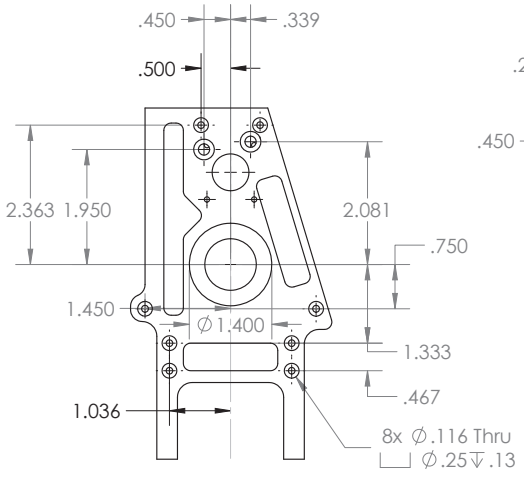


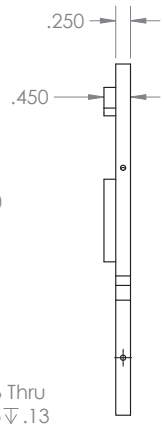


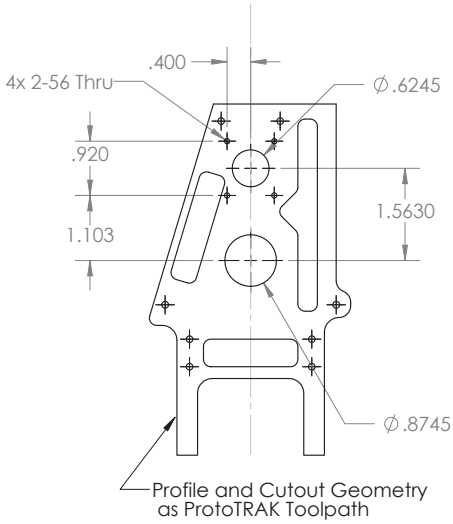
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005 INTERPRET GEOMETRIC TOLERANCING PER: MATERIAL: 6061-T6 Aluminum FINISH: NONE DO NOT SCALE DRAWING	RehabiLiGait		
	TITLE: Disc Hub		
	SIZE A	DWG. NO. P2-3	REV 1
	SCALE: 2:1	WEIGHT:	SHEET 1 OF 1

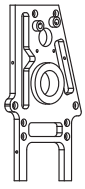


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait		
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Disc		
MATERIAL 304 Stainless Steel	SIZE A	DWG. NO. P2-4	REV 1
FINISH NONE	SCALE: 1:2	WEIGHT:	SHEET 1 OF 1
DO NOT SCALE DRAWING			

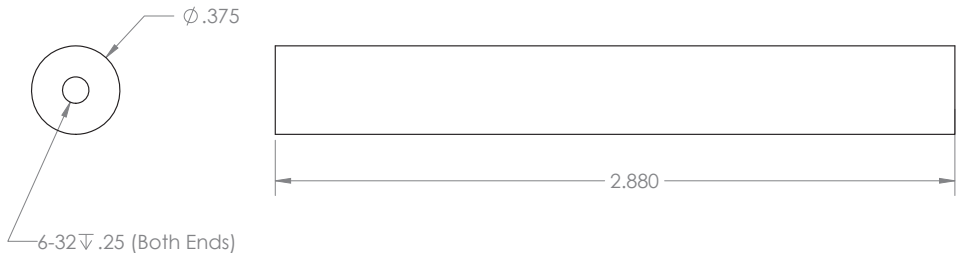




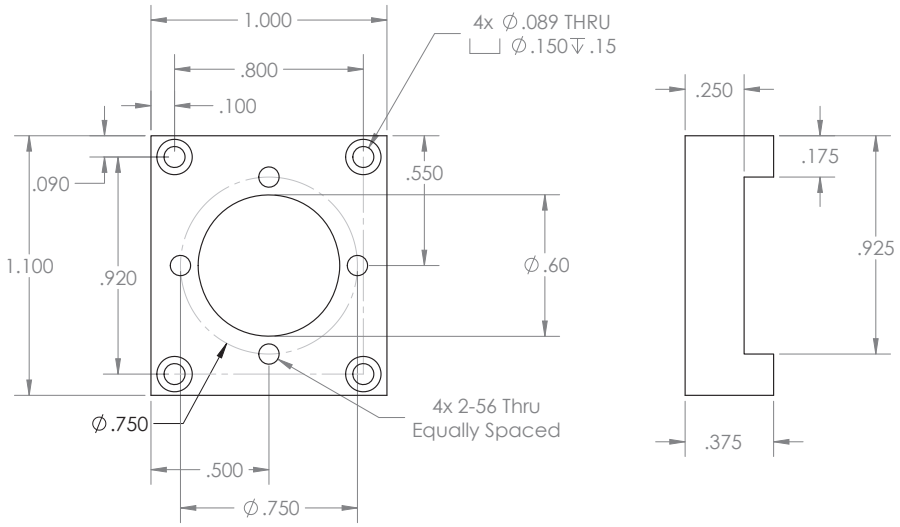


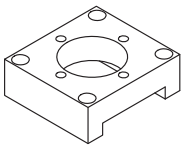


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait		
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Front Plate		
MATERIAL 6061-T6 Aluminum	SIZE A	DWG. NO. P3-1	REV 1
FINISH NONE	SCALE: 1:2	WEIGHT:	SHEET 1 OF 1
DO NOT SCALE DRAWING			

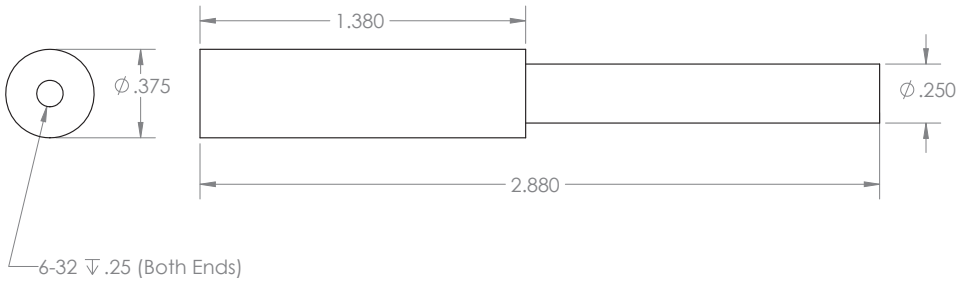


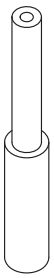
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Support Shaft
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. P3-4 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:1 / WEIGHT: / SHEET 1 OF 1



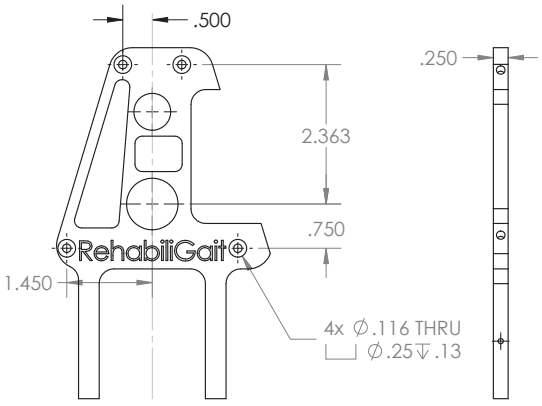


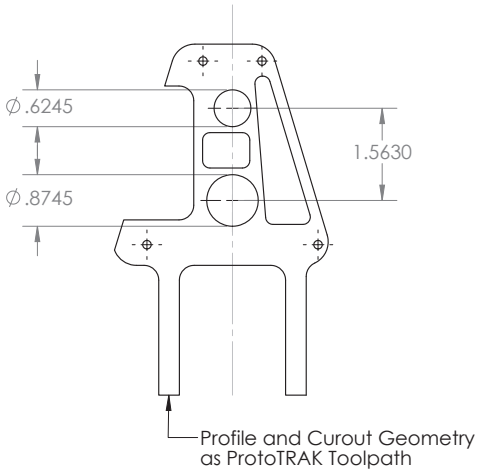
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Encoder Bracket
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. P3-7 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 2:1 / WEIGHT: / SHEET 1 OF 1



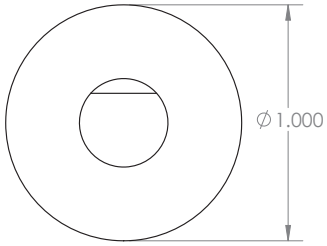


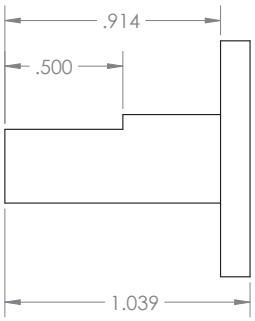
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Stepped Support Shaft
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. P3-11 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:1 / WEIGHT: / SHEET 1 OF 1

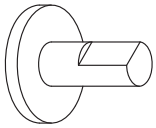




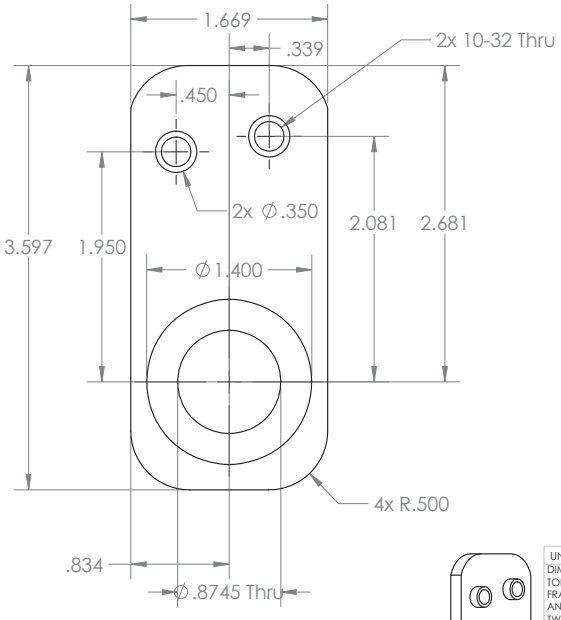
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Back Plate
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. P4-1 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:2 / WEIGHT: / SHEET 1 OF 1

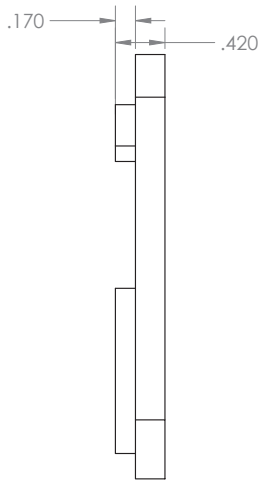


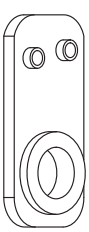




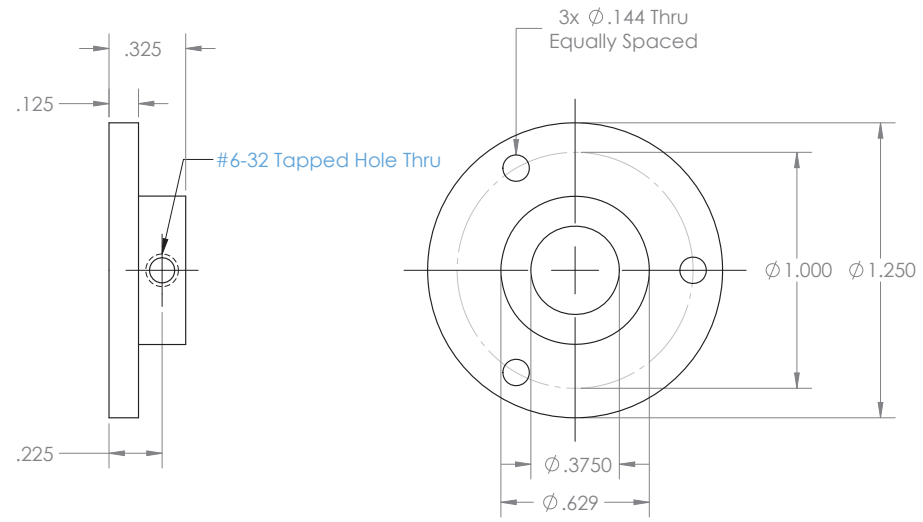
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Pin Plate
MATERIAL 303 Stainless Steel	SIZE A / DWG. NO. P5-1 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 2:1 / WEIGHT: / SHEET 1 OF 1

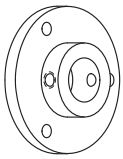






UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Side Plate
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. P5-2 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:1 / WEIGHT: / SHEET 1 OF 1





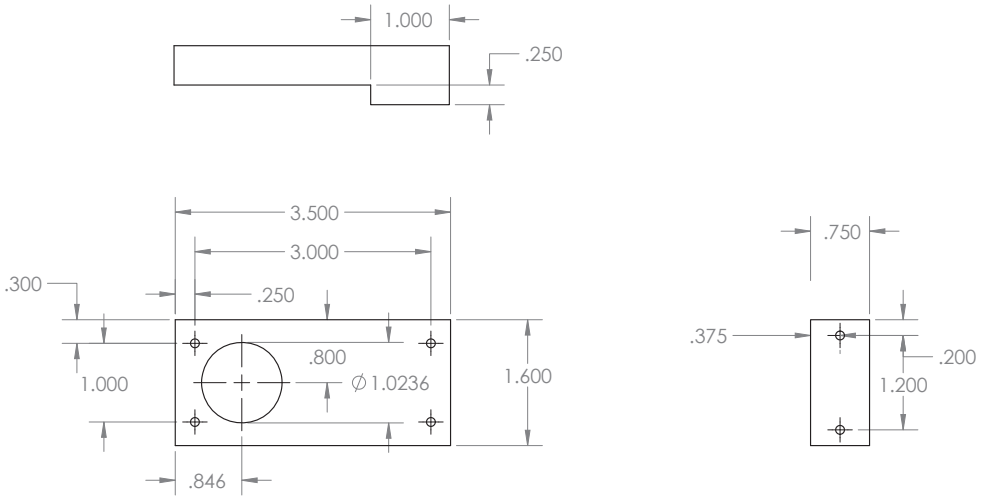
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL: 6061-T6 Aluminum
FINISH: NONE
DO NOT SCALE DRAWING

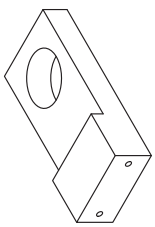
RehabiliGait

TITLE: Brake Hub

SIZE	DWG. NO.	REV
A	P5-3	1

SCALE: 2:1 | WEIGHT: | SHEET 1 OF 1





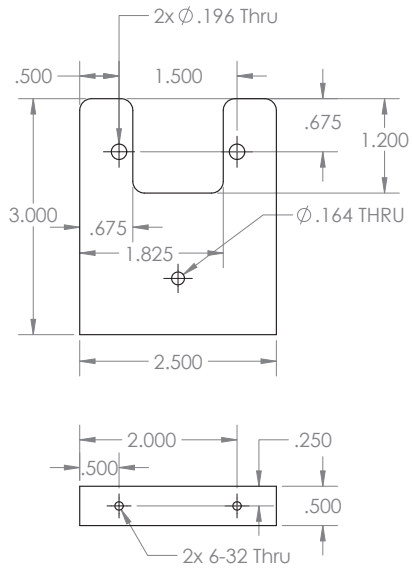
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL: 6061-T6 Aluminum
FINISH: NONE
DO NOT SCALE DRAWING

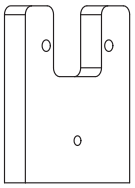
RehabiliGait

TITLE: Bearing Mount

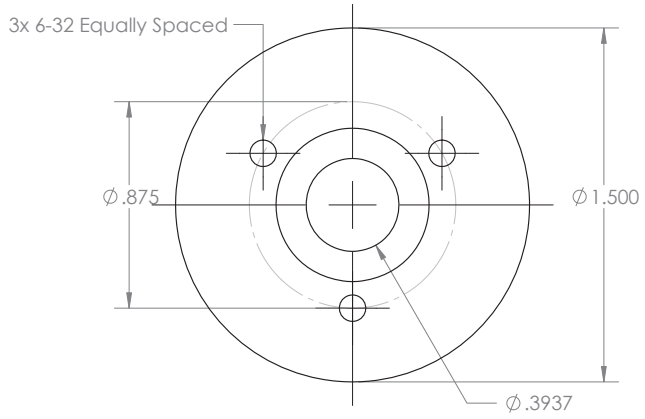
SIZE	DWG. NO.	REV
A	F1	1

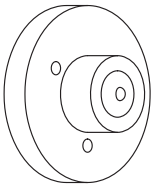
SCALE: 1:2 | WEIGHT: | SHEET 1 OF 1





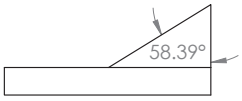
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Motor Mount
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. F2 / REV 1
FINISH NONE	SCALE: 1:2 / WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING	

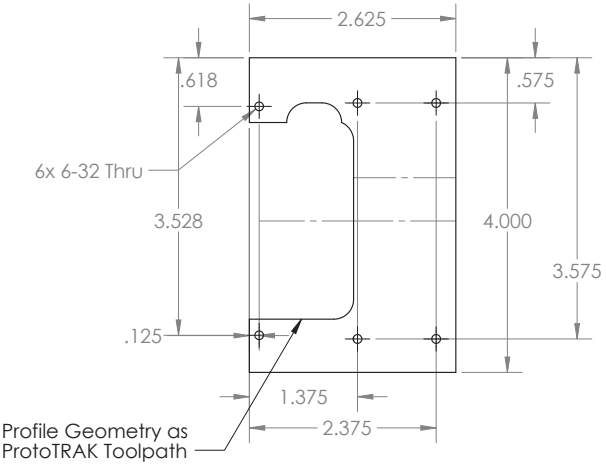


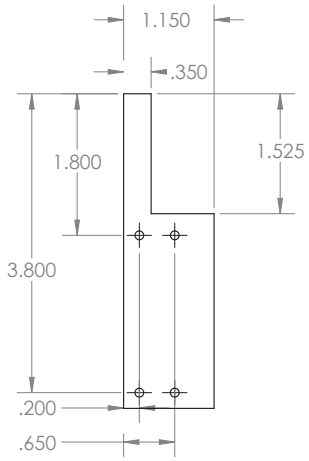


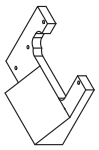
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Hub
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. F3 / REV 1
FINISH NONE	SCALE: 2:1 / WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING	

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: D-Shaft
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. F4 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:1 / WEIGHT: / SHEET 1 OF 1

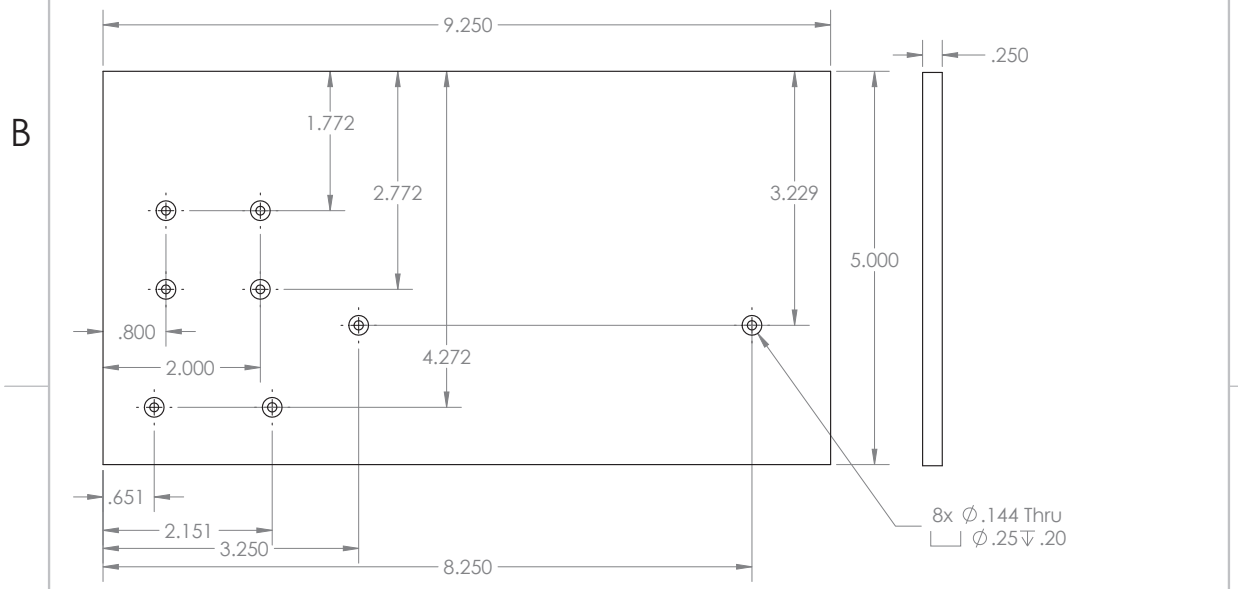




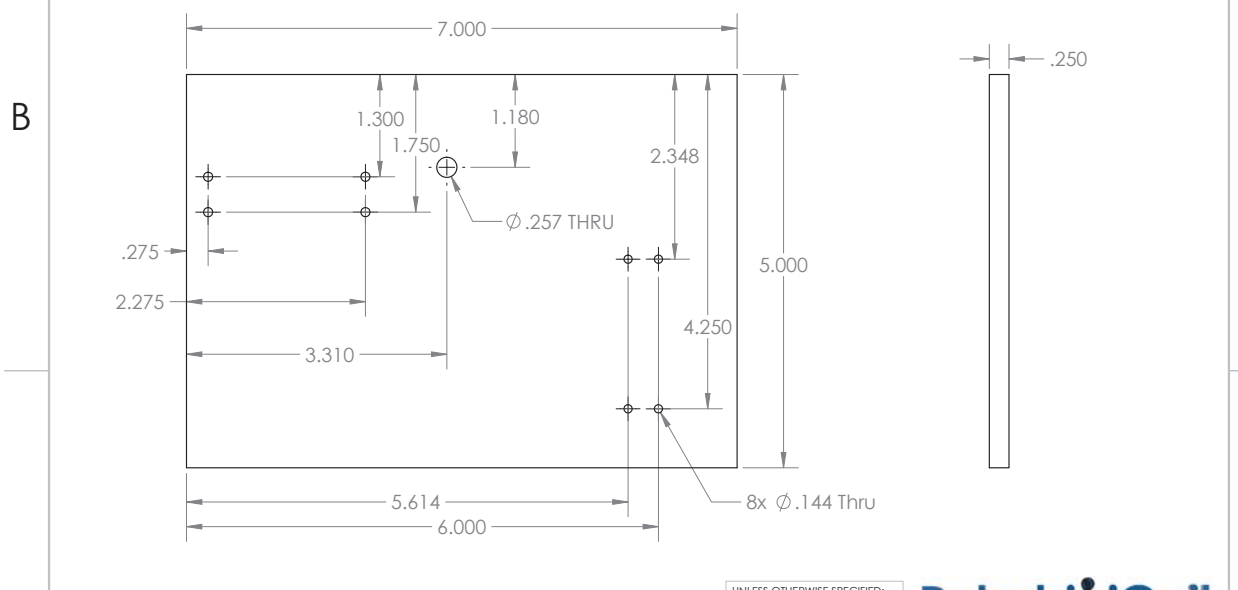




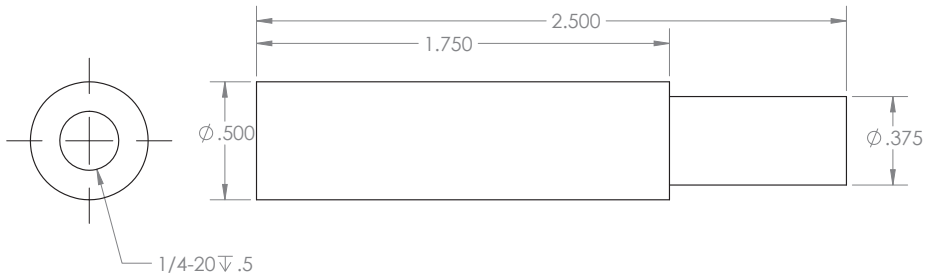
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiLiGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Brake Mount
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. F5 / REV 1
FINISH NONE	
DO NOT SCALE DRAWING	SCALE: 1:2 / WEIGHT: / SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Base
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. F6 / REV 1
FINISH NONE	SCALE: 1:4 / WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING	

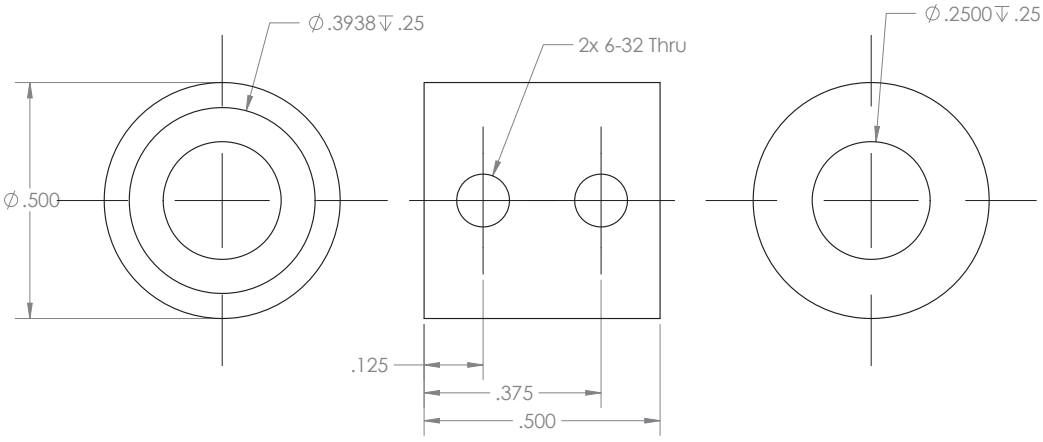


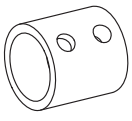
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	RehabiliGait
INTERPRET GEOMETRIC TOLERANCING PER:	TITLE: Wall
MATERIAL 6061-T6 Aluminum	SIZE A / DWG. NO. F7 / REV 1
FINISH NONE	SCALE: 1:2 / WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING	



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	TITLE: Groove Support	
INTERPRET GEOMETRIC TOLERANCING PER:		
MATERIAL 6061-T6 Aluminum	SIZE A	DWG. NO. F8 / REV 1
FINISH NONE	SCALE: 1:1	WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING		

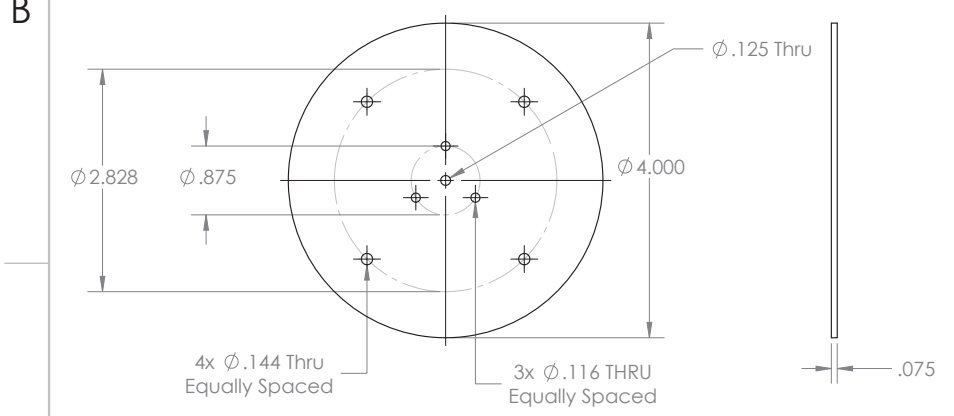
RehabiliGait

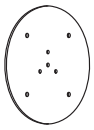




UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/64 ANGULAR: MACH ±1 BEND ±1 TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005 FOUR PLACE DECIMAL ±.0005	TITLE: Hub Connection	
INTERPRET GEOMETRIC TOLERANCING PER:		
MATERIAL 6061-T6 Aluminum	SIZE A	DWG. NO. F9 / REV 1
FINISH NONE	SCALE: 4:1	WEIGHT: / SHEET 1 OF 1
DO NOT SCALE DRAWING		

RehabiliGait





UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH ±1 BEND ±1
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
FOUR PLACE DECIMAL ±.0005

INTERPRET GEOMETRIC TOLERANCING PER:

MATERIAL: 303 Stainless Steel

FINISH: NONE

DO NOT SCALE DRAWING

RehabiLiGait

TITLE: Test Disc

SIZE	DWG. NO.	REV
A	F10	1

SCALE: 1:2 | WEIGHT: | SHEET 1 OF 1