Design of Dynamically Controlled Desktop Fiber Accumulator with Tension Feedback as Part of Smart Manufacturing Educational Kit

by

Shirley Suet-Ning Lu

S.B. in Mechanical Engineering
Massachusetts Institute of Technology, 2018

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

May 2020

Signature of Author: _____
Department of Mechanical Engineering
May 8, 2020

Certified by: _____
Brian W. Anthony
Principal Research Scientist, Department of Mechanical Engineering
Thesis Supervisor

Accepted by: _____
Nicolas Hadjiconstantinou
Professor of Mechanical Engineering
Graduate Officer

Design of Dynamically Controlled Desktop Fiber Accumulator with Tension Feedback as Part of Smart Manufacturing Educational Kit

By

Shirley Suet-Ning Lu

Submitted to the Department of Mechanical Engineering
On May 8, 2020 in Partial Fulfillment of the
Requirements for the Degree of

Master of Science in Mechanical Engineering

Abstract

In the age of automation and computation, smart manufacturing is being adopted by manufacturing industries because its data-rich approach enables production of higher-quality goods at reduced costs. However, there is an associated learning curve for operators and manufacturing professionals to transition from the use of simply automated machines, since the 1970s, to the smart, interconnected machines today. Therefore, the objective of this research is to develop a desktop fiber accumulator with a dynamic controller that monitors fiber tension as part of a smart manufacturing educational kit.

A Winding Loop Material Accumulator, WiLMA, is designed and tested as a machine whose main purpose is to be an effective buffer for the difference in speed between the machine upstream outputting fiber at a constant rate and the machine downstream intaking fiber at a variable rate without damaging or over-tensioning the fiber. To test the versatility of WiLMA, various operational profiles are used for the machines upstream and downstream. The upstream machine maintains a constant velocity profile, varying the ratio of the upstream machine's velocity to the downstream machine's velocity, from 1.25x to 3.0x. The downstream, variable-rate machine assumed two types of profiles: constant intake velocity, both low (10 mm/s) and high speed (40 mm/s); square wave profile for intake velocity, low (10 mm/s) and high speed (30 mm/s) as well as varying frequencies (0.1 Hz, 1.0 Hz).

The force applied to the fiber by a spring-loaded tensioner acts as a proxy for the fiber tension. This tensioning force is measured, and its standard deviation and error are used to evaluate WiLMA's performance. Further testing with different fiber materials, frequencies, and speeds are necessary to validate the versatility of WiLMA. However, with the force applied to the fiber nominally set at 0.75 lbs, all trials yielded less than 0.07 lbs of average standard deviation in this tensioning force. The results suggest that WiLMA performs well as an accumulator, distinguished from industry accumulators by its ability to respond to changing conditions, while not damaging the fiber. The feedback control system and data logging allow WiLMA to be used as an educational platform for smart manufacturing.

Thesis Supervisor: Brian W. Anthony
Title: Principal Research Scientist, Department of Mechanical Engineering

Acknowledgments

First and foremost, I would like to thank my family. Thank you to my sisters, Michelle and Catherine, for always having my back. Thank you to my parents for their support and confidence in me. I am especially appreciative of my dad's meal plan while I was writing this thesis since I had to move home early due to the COVID-19 pandemic.

I would also like to thank Dr. Brian Anthony, my advisor, for his advice and support on both my research direction and my career path forward.

Thank you to my labmates: David Donghyun Kim for your design reviews, Sangwoon Kim for your ROS setup help, Yasmin Chavez for controls explanations and company in navigating grad school and job search, etc., Alex Benjamin for your help troubleshooting my code and teaching me "kill %", Lauren Chai for your encouragement and support, and others in the Device Realization Lab for making grad school an enjoyable experience.

Thank you to my friends: Sherri Green for hanging out on the weekends so I don't become a workaholic, Lindsay Epstein for buddying me in makerspaces and classes as well as research struggles, and Jana Saadi for your encouragement and conversations.

Lastly, thank you to all my mentors at MIT the last six years as I conclude my MIT experience with this master's thesis, in particular, Bill Cormier, Dr. Danny Braunstein, Dr. Barbara Hughey, Steve Habarek, Jimmy Dudley, and Tasker Smith for always greeting me with encouraging smiles and life advice when I was anxious about research progress or my job search and wandered into Pappalardo Lab to loiter.

Table of Contents

List of Figures

List of Tables

1. Introduction

With the last surge of automation in the manufacturing space, known as the third Industrial Revolution or Industry 3.0 in the 1970s, use of programmable logic controllers, robots, etc. were integrated into manufacturing and production, greatly increasing the scale and efficiency of modern manufacturing. However, as modern manufacturing techniques and systems evolve in complexity, methods of monitoring and controlling the automated systems have also advanced, hence the onset of what industry analysts call Industry 4.0.

Industry 4.0 refers to the "rise of new digital industrial technology...that makes it possible to gather and analyze data across machines, enabling faster, more flexible, and more efficient processes to produce higher-quality goods at reduced costs" [1]. Smart manufacturing is a manifestation of Industry 4.0. A smart manufacturing system consists of smart machines, storage systems, and production facilities that are data-rich due to numerous sensors and are able to use that data to make decisions independently, i.e. without a human operator. For example, if a production error is causing faulty parts, in the traditional system, these parts would not be identified as non-conforming until the quality control stage, at which point, more faulty parts have been produced. Smart manufacturing allows real-time monitoring and decision making on the production line, minimizing wasted resources by detecting and resolving production faults earlier on.

Examples of such industries that would benefit from smart manufacturing are yarn/textiles and optical fibers since the product's functionality is highly dependent on its quality. To investigate smart manufacturing for use in fiber or filament production, a mockup of a desktop fiber production line consisting of three machines is designed, tested, and detailed in this paper. The fiber production system consists of a fiber extrusion device (FrED) [2], an accumulator, and a fabric machine. FrED produces a fiber while the fabric machine knits that fiber. Mockups of FrED and the fabric machine are designed and used in this experiment in place of the actual machines as those are still under development. This paper will focus on the design, build, and testing of the accumulator named WiLMA - Winding Loop Material Accumulator. WiLMA is required for continuous production of knitted fibers because FrED is ideally producing fiber at a constant rate while the fabric machine (in development at the time of this document) will likely have a discrete or time-varying profile. As such, a machine, WiLMA, is required between FrED and the fabric

machine to act as an accumulator or buffer to allow FrED to continuously produce fiber while allowing freedom in the design of the fabric machine and its intake profile.

To mimic a smart manufacturing environment, there is communication between the three machines as well as data feedback. For WiLMA, the parameter of interest is tension. Good tension control is significant for the yarn/textiles industry as well as optical fibers as it directly impacts the fibers' functionality. Therefore, the spring force measurement, a proxy for tension, and the associated statistics, such as error and standard deviation, will act as a measure of the effectiveness of the accumulator. In particular, low standard deviation in the tensioning force implies low variation in the tension of the fiber.

2. Background

2.1 Professional Education Program for Smart Manufacturing

Some of the functional requirements for the design of WiLMA are driven by the fact that it is intended to be part of a professional education kit that teaches smart manufacturing. As defined by the National Institute of Standards and Technology (NIST), smart manufacturing systems are "fully-integrated, collaborative manufacturing systems that respond in real time to meet changing demands and conditions in the factory, in the supply network, and in customer needs" [3]. The smart manufacturing systems do so by using the abundance of data available, provided by the machines themselves as well as on-the-line inspection and sensors, for example.

With the onset of Industry 4.0, as described in the Introduction, existing manufacturing systems are being outfitted with more sensors to enable smart manufacturing. Switching to a smart manufacturing system could increase efficiency by predicting failures, streamlining processes, and reduce costs for labor, materials, energy, etc., effectively enabling faster product development [4]. With these benefits, more companies are looking to pivot to smart manufacturing, resulting in an increased demand for smart manufacturing courses to equip employees with the necessary technical knowledge.

One such course is offered by MIT Professional Education online. The course description may be found here [5]. The course teaches basic principles of smart manufacturing through hands-on implementation of a smart machine, FrED, which is described in the following section, Section 2.2: Overview of FrED. The program is designed for plant managers in manufacturing, design and

manufacturing engineers, data scientists, etc., to be able to interact with FrED, its operation, and data to learn about smart manufacturing [5].

Similarly, the accumulator is designed with the intention of potentially expanding this course to include a smart manufacturing system or production line, as opposed to a single smart machine. As such, being part of a professional education kit drives some of the design aspects of the accumulator, such as modularity, ease of manufacturing and scaling, and be "data-rich" to allow insight into the machine operation. These functional requirements are discussed further in Section 4: Functional Requirements for Accumulator Design.

2.2 Overview of FrED

2.2.1 Design and Operation

FrED refers to the fiber extrusion device designed by David Donghyun Kim as part of his doctoral research with MIT's Device Realization Lab. FrED is "a desktop fiber prototyping system [that] can be used to teach...students about feedback control systems and manufacturing" [2]. It is modeled after optical fiber draw towers, comprising four main components: extruder, cooling system, spool, and sensors; the components are labeled in the following CAD model (Fig. 1).

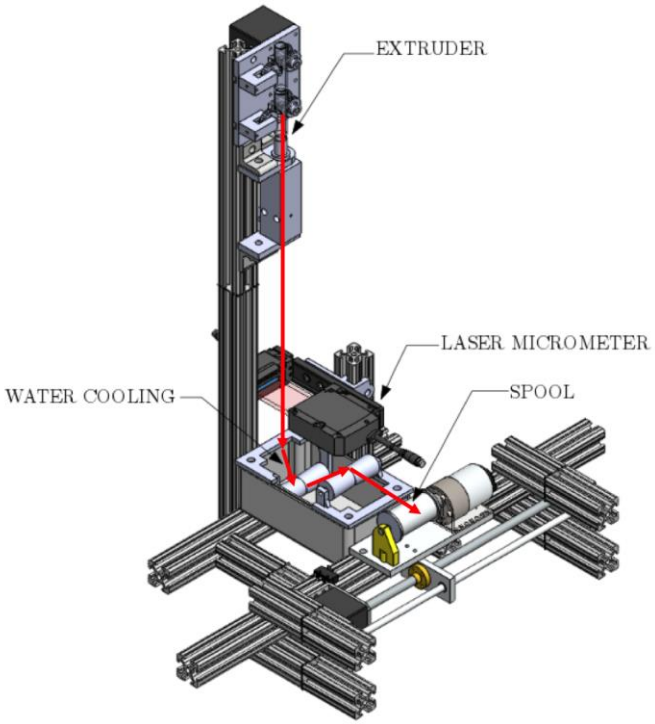


Figure 1: CAD Model of FrED, a desktop fiber production machine in which a fiber is drawn from the extruder, its diameter measured by the laser micrometer, cooled for stability, and spooled up [2]

The extruder component is the heating element through which the round stock material (generally called preform, typically plastic) is fed into and heated to its glass transition (melting) temperature. The stepper motor that feeds in the preform also forces the melted material through the significantly smaller outlet hole at the end of the extruder; this glassy material coming out of the extruder is then pulled, or drawn, to create a fiber with a diameter less than 0.4 mm. The diameter of the fiber can be controlled by adjusting the speed at which the material is drawn out of the extruder. The fiber then travels through the cooling system, which consists of idler pulleys

that guide the fiber through a water bath, cooling the fiber to solidify and stabilize it. Once cooled, the fiber is collected with the spool. The rotational speed of the spool controls the fiber diameter: the faster the spool rotates, the thinner the fiber. The speed of the spool is determined by a closed-loop control system. The feedback is given by the laser micrometer that measures the fiber diameter as it enters the cooling bath. Background information on feedback control may be found in Section 8.1.1: Closed Loop Control System.

2.2.2 Relevant Considerations

For David Donghyun Kim's doctoral thesis, the version of FrED outlined in the previous section has since evolved to address variation in fiber diameter due to the spool assembly design. Eccentric effects due to the misalignment in the spool belt drive manifested in the diameter data as a sinusoid. Identifying the trend in the diameter data helped determine the underlying cause in the machine, a prime example of smart manufacturing and using the data to solve process or machine problems. As such, this design of FrED, shown in Figure 1, is intentionally kept as part of the smart manufacturing educational kit.

2.3 Overview of Fabric Machine

The fabric machine is a desktop knitting machine currently in development by David Donghyun Kim as part of his doctoral research with MIT's Device Realization Lab. Ultimately, the purpose of the fabric machine is to take and knit the fiber produced by FrED to produce fabric with integrated functionality, for example, fabric that can act as a pressure sensor. The functionality may be achieved through knitting specific patterns and/or knitting fiber with specific properties, such as nitinol, a common super-elastic shape-memory alloy.

The following figure (Fig. 2) is a preliminary CAD of the fabric machine currently in development. The underlying operating principle of this desktop fabric machine design is similar to that of commercial knitting machines (Fig. 3) where the loop in the knit is formed by an individual needle, and a row is formed by the needles making loops sequentially. A video of a commercial knitting in operation may be found here [6]. Each time a needle is actuated, the fabric machine is taking in fiber to form the loop. Therefore, the rate at which the fabric machine actuates its needles is proportional to the effective linear velocity of the fabric machine's fiber intake. Once a row has been knitted, the stage must reset to start knitting a new row. The stage resetting in addition to each individual needle actuating reflects the stop-and-go nature for the fabric machine's

fiber intake behavior. At this stage in its development, the approximate intake velocity of the fabric machine is less than 5 mm/s.

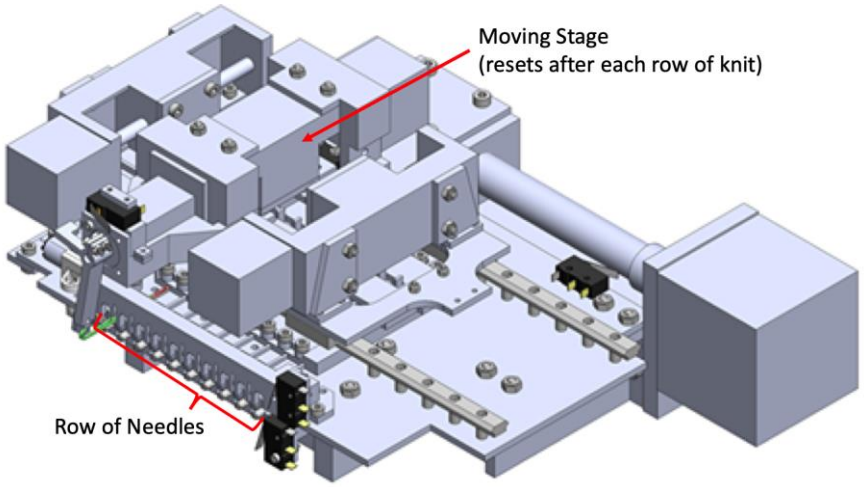


Figure 2: Preliminary CAD of the fabric machine (in development by David Donghyun Kim as part of his doctoral thesis with MIT's Device Realization Lab), modeled after commercial knitting machines

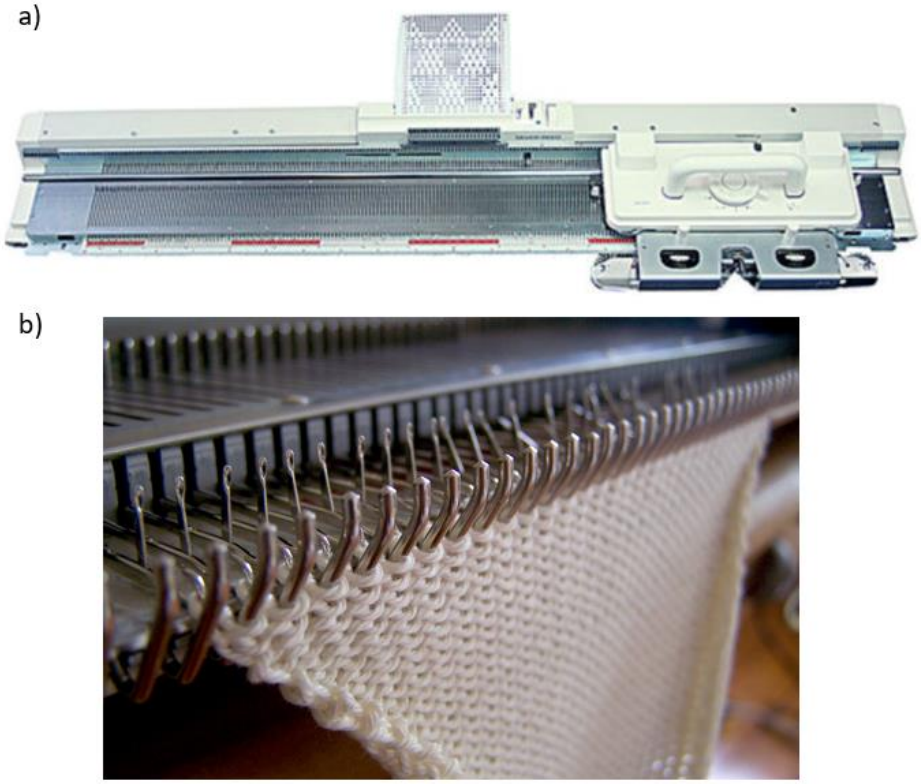


Figure 3: a) commercial knitting machine [7]; b) close-up of needles and knit on a commercial knitting machine [8]

2.4 Production System Overview

Figure 4 illustrates the high-level overview of how the production system operates with the flow of the material as well as the parameters being monitored.

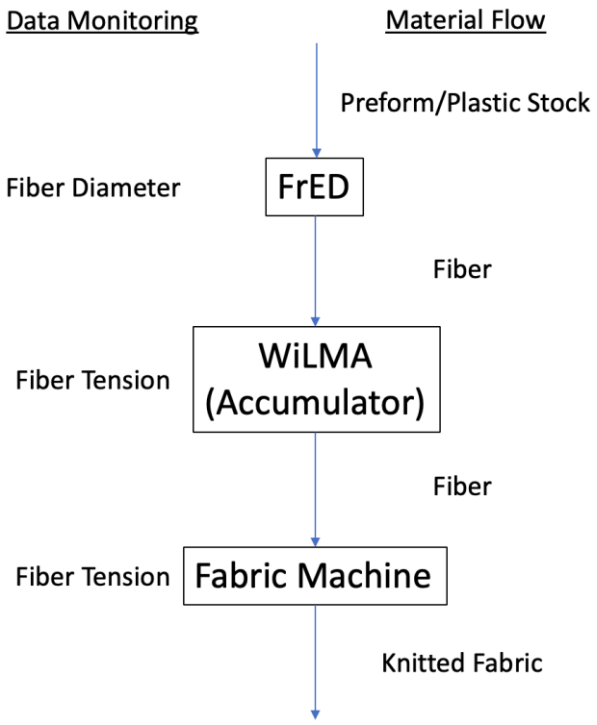


Figure 4: High-Level Production System Overview

3. Review of Relevant Industries

3.1 Yarns and Textiles

3.1.1 Yarn Tensioning Devices

According to the Fundamentals of Yarn Winding, the most common method to tension a running yarn is "to apply frictional retarding force to running yarn" [9]. This concept is illustrated in the following figure (Fig. 5). Yarn tensioning devices (tensioners) can be typically placed into four categories/types: additive, multiplicative, additive cum multiplicative, and self-compensating.

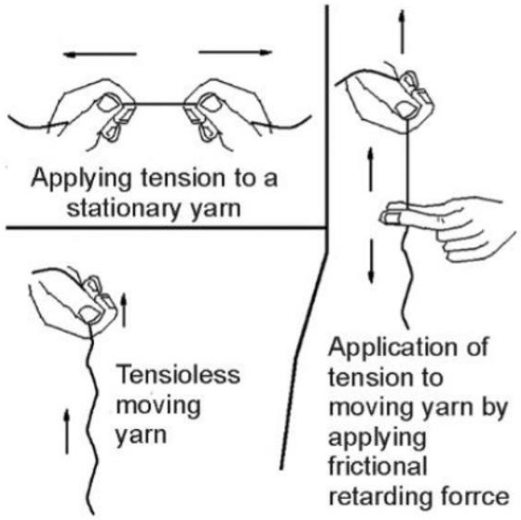


Figure 5: Tensioning yarn by applying frictional retarding force (Figure courtesy of Milind Koranne, Fundamentals of Yarn Winding [9])

Additive Tensioners

Examples of additive tensioners are ball tensioners and disc tensioners. These tensioners straightforwardly apply friction to the running yarn by effectively pinching the yarn between two surfaces, as seen in the following figure (Fig. 6). Additive tensioners apply a constant tension to the input tension, governed by this equation:

$$T_o = T_i + 2\mu W \quad \text{(Eq. 1)}$$

Where T_o is output tension, T_i is input tension, W is applied force by the tensioner and μ is the coefficient of friction between the yarn and the tensioner.

In general, compared to the other types of tensioners, it is easier to thread the yarn through an additive tensioning system. However, these tensioners are not typically used for yarn with twists as it would interfere with the twist and effectively untwist the yarn as it passes through [9].

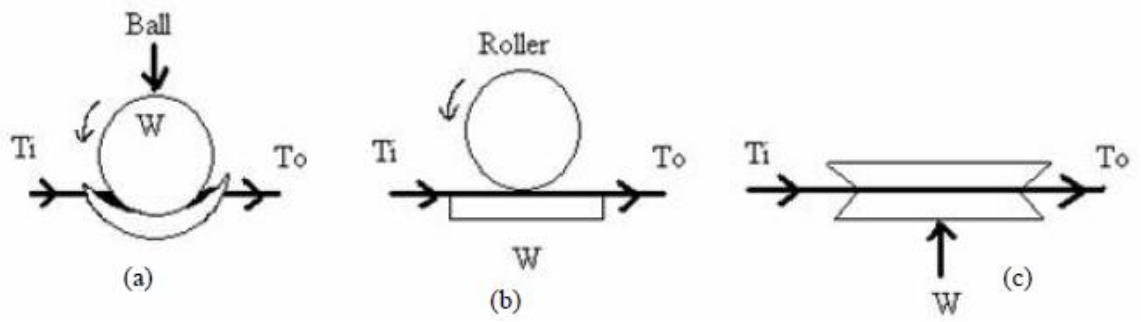


Figure 6: Additive tensioners: a) ball tensioner b) roller tensioner c) disc tensioner (Figure courtesy of Vivek Prasad Shankam Narayana [10])

Multiplicative Tensioners

As the name suggests, multiplicative tensioners operate by multiplying the incoming tension. Tensioning systems in this category apply tension via posts that engage with yarn. An example is a gate-type tensioner. Gate-type tensioners consist of two gates, each with a set of polished steel rods, porcelain tubes or ceramic coated posts as seen in Figure 7 [9]. As the movable gate closes, the angle theta increases as the amount of yarn in contact with the posts effectively increases, increasing the friction in the system. Multiplicative tensioners can generally be modeled by the capstan equation:

$$T_o = T_i e^{\mu\theta} \qquad \text{(Eq. 2)}$$

Where T_o is output tension, T_i is input tension, μ is the coefficient of friction between the yarn and the tensioner, and θ is the angle wrapped by the yarn onto the tensioner.

Since the posts are all moved simultaneously, the change in frictional force per unit of movement by the gate is proportional to the number of posts, resulting in a sensitivity to tension fluctuations and amplification of the change in frictional force [9].

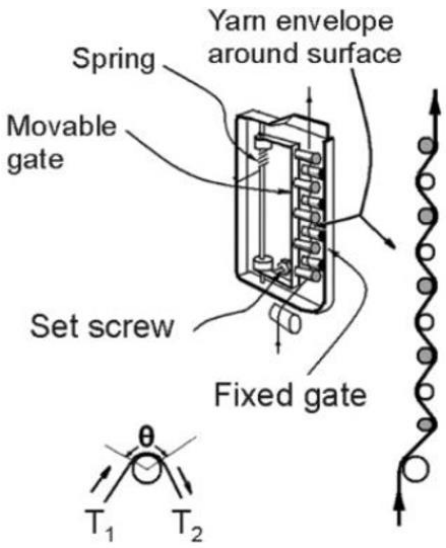


Figure 7: Gate-Type Tensioner (Figure courtesy of Milind Koranne, Fundamentals of Yarn Winding [9])

Additive cum Multiplicative Tensioners

Additive cum multiplicative tensions combine both additive and multiplicative elements. An example of this is shown in the following figure (Fig. 8). The additive component is contributed by the disc tensioners (Fig. 6c) that sandwich the fiber while the multiplicative component is contributed by the configuration of the tensioners, where each disc tensioner is analogous to the posts seen in Fig. 7.

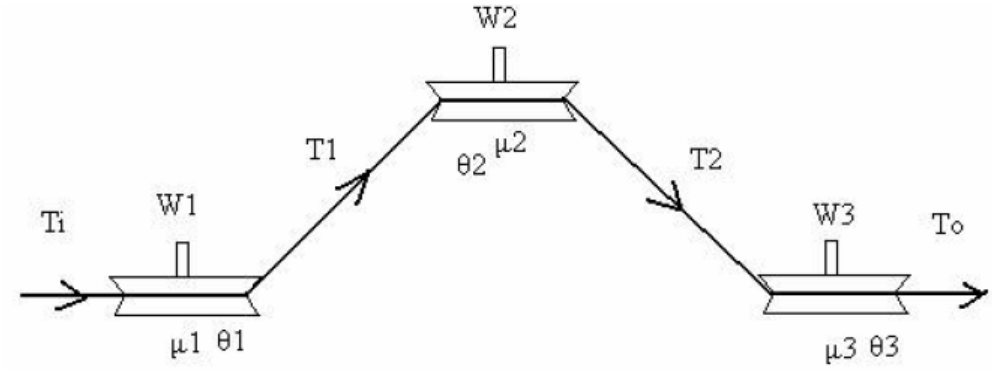


Figure 8: Typical additive cum multiplicative tensioner (Figure courtesy of Vivek Prasad Shankam Narayana [10])

Self-Compensating Tensioners

Self-compensating, or self-adjusting, tensioners often contain a spring to passively respond to changes in the tension. An example of this is a lever-type device, where the movable "gate" is spring-loaded, closing in when the tension is high and opening when the tension is low (Fig. 9).

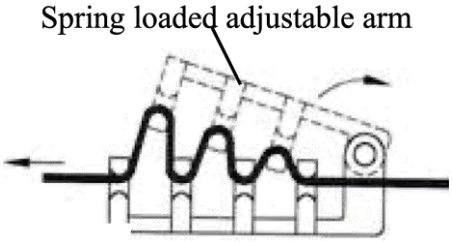


Figure 9: Lever tension device, an example of a self-compensating tensioner (Figure courtesy of Vivek Prasad Shankam Narayana [10])

3.1.2 Yarn Tension Regulation Methods

The two methods of interest to regulate yarn tension described in the Fundamentals of Yarn Winding [9] are through change in winding speed and application of a controllable yarn tensioner, such as those discussed in the previous section.

The method described for regulating tension through change in winding speed is reliant on the use of a temperature sensor that is installed on the yarn guide eyelet, the typically metal component that attaches to the machine and guides the yarn from its source to the machine. The temperature reading is expected to increase when the friction caused by the yarn on the eyelet increases. This friction and hence temperature act as an indirect measure for the tension in the yarn, providing feedback to regulate the winding speed.

Alternatively, popularly used in several winders available in industry [9], an electronic sensor as part of a tensioner (gate-type or disc) is installed in the yarn path to continuously measure the yarn tension through contact and deflection of the tensioner. The yarn tension measurement directly provides feedback for a closed control loop to control the tensioner.

More work is being done to outfit modern machines with "active computer-controlled tensioners with the possibility of programming yarn tension" [11]. Umirov et al. compares the control of a tensioner that has active disturbance rejection control (ADRC) against conventional proportional-derivative (PD) control. The experimental results showed that the ADRC achieved

better control of tension than the PD controller, in response to varying system parameters, such as yarn transport speed, or how fast the yarn is passing through the system.

Another such work in developing a yarn tension control system is the doctoral thesis of Ian Clifford Wright from the Loughborough University of Technology [12]. One of the compensator designs explored by Wright is a pneumatic passive compensator. The operation and testing of the operator can be seen in Figure 10. In short, the running yarn is wrapped around a disc; the position of that disc is controlled by the incoming air flow, effectively controlling the yarn tension.

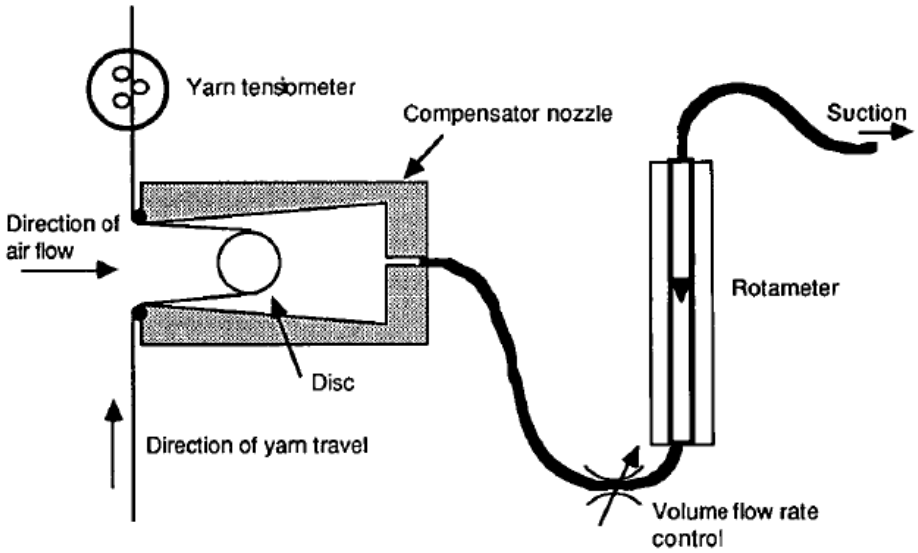


Figure 10: Pneumatic compensator designed for yarn tension control in the high-speed formation of conical packages of yarn (Figure courtesy of Ian Clifford Wright [12])

This novel pneumatic compensator was designed specifically for yarn tension control in the high-speed formation of conical packages of yarn. Note that this system may not be ideal for the fibers of interest in this paper as these fibers, drawn from acrylic and polycarbonate for example, are limited in their allowable bend radius before damaging the fibers. Also, based on preliminary experimentation with fiber drawn from acrylic and polycarbonate preforms, the fibers tend to have shape memory and are less compliant than yarn, hence would not necessarily maintain contact with the disc as tensions change in this pneumatic compensator system.

3.2 3D Printing Filament Spools

Another area of industry in which an accumulator is used is in spooling 3D printing filament. A video of the process may be found here [13]. A typical 3D printing filament production line can be seen in the following figure (Fig. 11). In short, the plastic pellets are melted, drawn, and cooled to produce the 3D printing filament. Once the filament is produced, it winds through the linear tension control accumulator (LTCA), composed of two idler pulleys (boxed in red in Fig. 11). When the spool being loaded is full, the operator must stop the spool intake of filament and replace the full spool with an empty one. In this time, the accumulator's pulleys drive apart to buffer the incoming filament that is being continuously produced by the upstream machine.

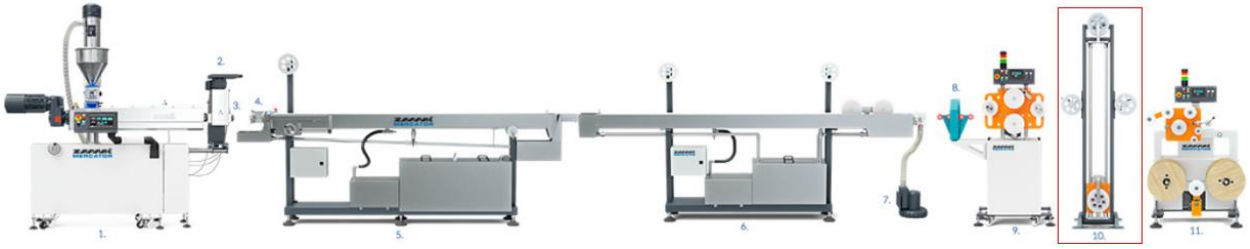


Figure 11: 3D printing filament production line; machines include (left to right): extruder, cooling stations, quality control (diameter inspection), accumulator, winding station [14]

Industrial linear tension control accumulators, such as the ones sold by Progressive Machine Company (Ormond Beach, FL) and REELEX Packaging (Patterson, NY), typically control the tension by manually "adjusting the number of counter weights to provide tension levels from as low as two oz. to as high as eight oz." [15]. The physical counterweights are calibrated and optimized for the production line in steady state. This method of managing tension is suitable for 3D printing filament, for example, because the production conditions and fiber parameters are not changing. However, using physical counterweights that must be manually adjusted is not ideal for dynamically changing systems.

4. Functional Requirements for Accumulator Design

As highlighted in Figure 4, the main functional requirement for the accumulator design is to mitigate the speed difference between FrED's fiber production and the fabric machine's fiber intake such that the production line (consisting of FrED, the accumulator, and the fabric machine) can operate continuously for an extended period of time. As explained in Section 2.2: Overview of FrED, the fiber is drawn after heating the extruder to melt the preform. Therefore, it is ideal to maintain a high extruder temperature and minimize temperature variation to decrease variation in the fiber diameter. Therefore, ideally, the accumulator should operate such that FrED does not need to stop fiber production in order to accommodate the fabric machine, assuming that the average speed of FrED's fiber production is faster than the fabric machine's fiber intake.

Additionally, to allow flexibility in the choice of the fiber material, the accumulator must not damage the fiber as it passes through the accumulator. Sensitivity to the fiber tension will allow fiber materials that provide additional, or embedded, functionality to be used in the production line as well. Examples of such materials include biocompatible materials for neural probes [16] and fibers with embedded filler materials, such as carbon nanotubes arranged in specific patterns [17]. Additionally, variation in yarn tension, for example, has shown to affect the physical properties of the produced yarn [10]. Therefore, the accumulator must have sensitive tension control to prevent over-tensioning the fiber or mechanically stressing the fiber in any way.

Driving the development of the accumulator and the associated assembly line, including FrED and the fabric machine, is the use of the machines for professional education. As part of an educational kit, the accumulator design must be designed for manufacturing, assembly and scaling. As such, the number of unique parts as well as the number of custom parts in the kit should be kept at a minimum. Consequently, the cost of the accumulator should also be minimized when possible.

In summary, the functional requirements for the accumulator design are:

- Effectively buffer the speed difference between FrED (the machine upstream) and the fabric machine (the machine downstream) such that the production line can operate continuously for an extended period of time
- Does not damage the fiber (no twisting, over-tensioning, stressing, etc.)
- Easy to manufacture, assemble, and scale

5. Design Concepts

Several design concepts were explored in the initial brainstorming stage through sketch modeling, CAD modeling, or a literature review of existing systems. The potential concepts are outlined briefly in the subsequent sections, followed by a comparison of the design concepts' advantages and disadvantages.

5.1 Tapered Pulley Design

The tapered pulley design resembles a cone without the tip, which acts as an idler pulley. A rough CAD model of the design as well as a sketch model can be seen in Fig. 12. The design concept was intended to address the functional requirement of effectively buffering the speed difference, i.e. difference in rates of material into and out of the system, by creating a stationary spool with effectively variable diameter. The design intention was to minimize the number of moving parts in the system, decreasing the machine's footprint and complexity, hence its likelihood of failure.

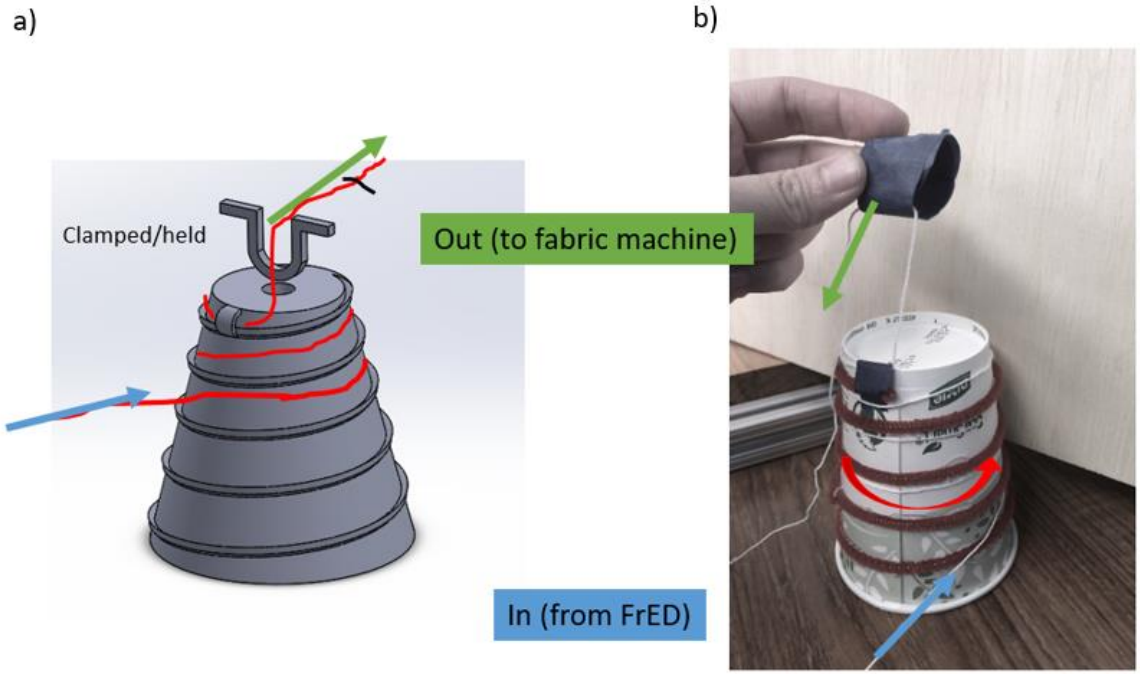


Figure 12: a) Preliminary CAD model of the tapered pulley design; b) sketch model of the tapered pulley design

In principle, this tapered pulley accumulator fills up with fiber along its length as FrED begins fiber production while the fabric machine is idle. Once the accumulator/buffer is sufficiently full,

the fabric machine would begin its operation, pulling fiber out of the tapered pulley accumulator. Assuming the rate of the fiber leaving the tapered pulley buffer is greater than the rate of the fiber entering the buffer, eventually, the buffer is emptied. The design vision was for the buffer to be sized such that the time at which it becomes empty coincides with the time the fabric machine begins a new row of knit, providing extra time to FrED to repopulate the buffer. With the fabric machine stopped, in this design, a motor would be necessary to drive the pulley to take up the fiber as FrED continues adding fiber to the buffer.

Note that at the time of this design consideration, the fabric machine was assumed to have stop-and-go operation but have an overall fiber intake rate that was faster than FrED's fiber production rate. This assumption has since changed with further development of the fabric machine. The fabric machine will still have a stop-and-go profile but will likely have an overall fiber intake rate that is slower than FrED's fiber production rate.

The tapered shape of the pulley was intended to guide the fiber to the location where the fabric machine pulls the fiber out of the buffer. Ideally, this location is fixed to simplify the connecting interface between the accumulator and the fabric machine. A sketch model was built to test the behavior of the fiber given the fixed connection point and rotating buffer. The sketch model showed the fiber twisting as it exited the buffer through the fixed holding point. The twisting was particularly obvious in the scenario where the fabric machine has stopped taking from the buffer and the buffer is spinning to take in the fiber produced by FrED.

For preliminary consideration, this design is relatively simple and requires few parts, namely a tapered pulley, "threaded" to constrain the fiber along the length of the pulley, and a motor to drive the pulley when the fabric machine stops. However, the main drawback is that the fiber would be twisting as it comes out of the buffer and into the fabric machine. Addressing the twisting of the fiber, whether actively untwisting the fiber on the buffer's output or not twisting the fiber in the first place, would likely require a more complex design of the fabric machine intake mechanism and/or of the buffer/pulley itself. Therefore, other designs were explored.

5.2 Center-Pull Winder Design

Another design considered in the early stages was the winder design, a device used by yarn hobbyists that winds up yarn into a ball that allows the user to pull yarn from the center; essentially a device that makes center-pull yarn balls. A graphic of the device is pictured below (Fig. 13); a

video demonstration of a commercial yarn ball winder winding fiber may be found here (Appendix H). The video demonstrates the successful winding of the fiber as well as the fiber being extracted from the center of the fiber "ball". The fiber used in the demonstration video is PMMA fiber produced by FrED.

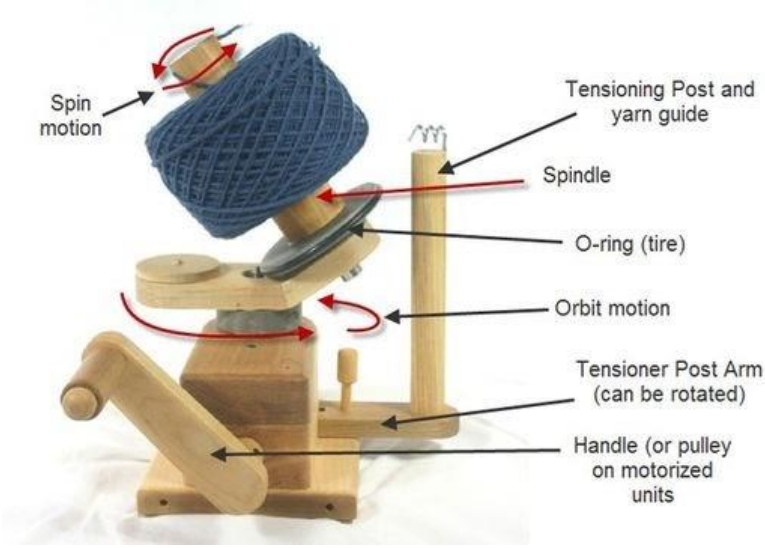


Figure 13: Diagram of Yarn Ball Winder Components (Courtesy of Heavy Duty Ball Winders by Nancy's Knit Knacks [18])

This design was explored because of its potential to be an infinite buffer since the fiber produced by FrED would continue to be wound around the outside of the ball while the fiber machine pulled from the center, effectively making it independent of the fabric machine's operational parameters.

Similar to the tapered pulley design, the fiber coming out of the center of the winder twists, resulting in some fiber deforming/kinking. The fiber produced by FrED is more susceptible than yarn to permanent kinking since the fiber material is less flexible. The combination of the fiber being lower friction and more resistant than yarn to bending makes it difficult for the fiber to stay wound on the spindle without sliding up and off the spindle. A video demonstrating the low friction fiber sliding off the spindle may be found here (Appendix H). A frame from this video is shown in the following figure (Fig. 14); there is a loop that forms in the fiber due to the fiber slipping off the spindle. The loop disappears as the spindle keeps spinning but leaves a kink in the fiber.

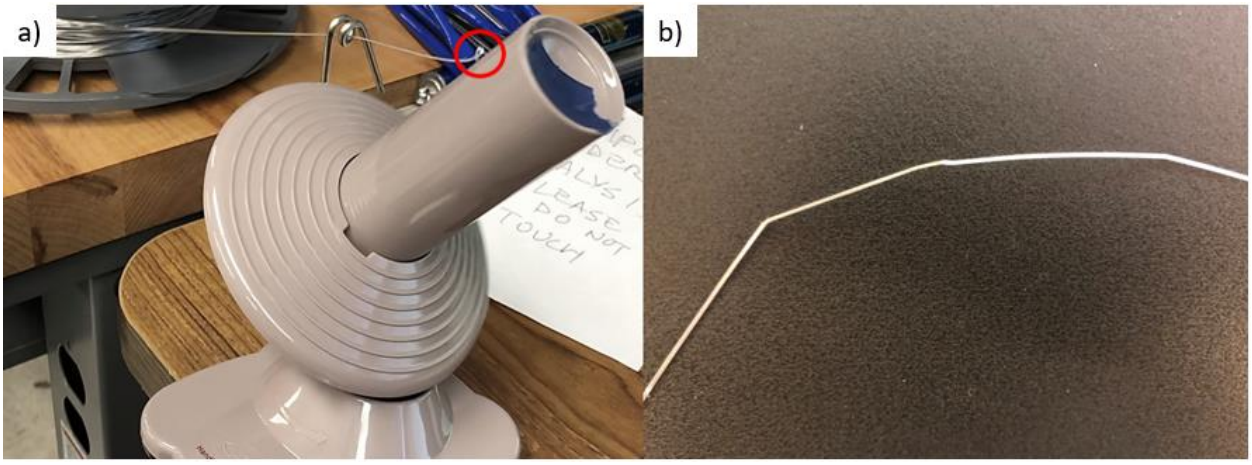


Figure 14: a) loop forming from fiber slipping off spindle of commercial yarn ball winder; b) permanent kinking of fiber due to loop formation

Since the fiber is rigid and low friction, it was hypothesized that the off-axis motion of the spindle is not necessary; winding the fiber around an upright spindle should still allow the end tucked inside to be pulled out by the fabric machine relatively easily.

However, the commercial winder is intended to wind yarn into center-pull balls for future use. In the context of the fiber production line, the act of the fabric machine pulling the fiber out from the center while the "ball" is still on the spool would result in the "hollowing" out the core of the ball. As a result, the ball begins to lose contact with the spindle in the center. If the winding speed is increased, the "tightness" of the newly wound fiber may bring the fiber "ball" back in contact with the spindle; however, that would result in effectively pinching the older fiber on the inside, soon to be pulled out by the fabric machine, between the freshly added fiber and the spindle, making it difficult for the fabric machine to pull the fiber out. Alternatively, if the fiber winding speed is maintained, another considered option was gradually "growing" the diameter of the spindle to meet the growing inner diameter of the fiber ball, effectively preloading against the inside of the fiber ball.

This center-pull fiber ball idea was not further explored due to the fact that the fiber coming out of the center would be twisted and/or otherwise deformed in the process of removal by the fabric machine. Addressing this fiber deformation, as in the case of the tapered pulley design, as well as the need to preload the spindle against the inside of the fiber ball calls for additional components, and hence a more complex design – not ideal for an educational kit intended to scale.

5.3 Linear Tension Control Accumulator

As explained in the review of 3D print filament spool production (Section 3.2), another design concept considered was linear tension control accumulators (LTCA), which involves two idler pulleys or sheaves; in general, one pulley is stationary while the other is moving. The distance between the pulleys changes corresponding to the needed buffer size to accommodate the difference in the rates of incoming and outgoing filament/fiber.

To scale the industrial version of this LTCA (Fig. 15) down to a desktop accumulator, mechanical design factors to consider are stepper motor force, design of the linear rail to meet the functional requirement of design for manufacturing/assembly as part of an educational kit and accumulator size/footprint, dependent on the pulley/sheave size and height of the machine.

Figure 15: Example of an industrial linear tension control accumulator (Courtesy of REELEX Packaging [19])

5.4 Choosing a Design

5.4.1 Evaluation of Design Concepts Against Functional Requirements

Each functional requirement can be roughly translated into corresponding design features. For example, ease of manufacturing, assembly, and scaling maps to the complexity of the design, number of parts, number of motors, etc. The simplest design concept is the tapered pulley design as it requires a single motor that could directly drive the pulley with few additional parts. The most complex design is the center-pull winder design because it would require a system of gears for a motor to drive the spindle off-axis as well as additional parts and potentially additional motors to untwist or prevent twisting of the fiber coming out of the buffer. The linear tension control accumulator design may require more components but is relatively straightforward in its implementation.

In order to effectively act as a buffer between FrED and the fabric machine, the designed accumulator must be flexible in handling the discrepancy between the rate of the fiber going in and the rate of the fiber exiting the accumulator. The center-pull winder design is arguably the best in handling the material flow rate discrepancy because it essentially operates as a buffer with infinite capacity. Regarding the controls of the system, the winder simply needs to speed-match FrED's production speed; it can operate independent of the fabric machine parameters. The other two designs require information from both FrED and the fabric machine or another source of feedback in order to dynamically manipulate the limited buffer size.

Lastly, the designed accumulator must be able to manage the tension, through dynamic control as necessary, and prevent mechanically stressing the fiber in any way, such as twisting. This functional requirement is the most important to satisfy given the potential applications of the fiber production line to produce sensitive fibers, such as neural probes or fibers with integrated functionality via specific embedded carbon fiber arrangements (Section 4: Functional Requirements for Accumulator Design). After preliminary exploration of the design concepts, the center-pull winder design would require the most extensive design work to prevent twisting or to untwist the fiber on exiting the buffer as well as appropriate preload on the inside of the fiber ball to maintain constant tension throughout.

One of the functional requirements that has changed significantly as a result of the preliminary exploration of the design concepts is that the accumulator should be an infinite buffer that operates independent from the fabric machine parameters (because those parameters were unknown at the

time of brainstorming). This requirement has been changed in light of progress in the development of the fabric machine. Now, the functional requirement is that the accumulator should effectively buffer the speed difference between FrED (the machine upstream) and the fabric machine (the machine downstream) such that the production line can operate continuously for some extended period of time. The relaxation of this requirement from being an infinite buffer to a finite buffer is due to the fact that the progress on the design of the fabric machine suggests that the fabric machine will likely operate at a slower velocity than FrED and that FrED can be slowed without affecting the fiber diameter. In addition, without relaxing the requirement, the center-pull winder design would have been the only design considered that would satisfy this requirement. However, the center-pull winder design was determined to be worse in terms of tension management and not mechanically stressing the fiber. The functional requirement of not stressing or damaging the fiber in any way is prioritized over the buffer size requirement.

If a future version of the fabric machine or another such machine downstream of FrED operates faster than FrED, there must be direct communication between the buffer and FrED. The buffer size and amount must be coordinated with the operational speed of FrED such that the buffer is repopulated when the fabric machine is in the stopped phase of its stop-and-go operation. Anecdotally, this means the fabric machine is taking fiber out of the buffer faster than FrED is putting fiber in. The buffer will be sized according to FrED's fiber production speed so that the fabric machine does not run out of fiber to use during its fiber intake phase. Then, when the fabric machine pauses to begin a new row of knit, FrED must produce fiber at a rate such that the buffer can be repopulated to meet the demand of the fabric machine once it starts up again. Additionally, if necessary, the length of the fabric machine's downtime could be increased to accommodate the buffer repopulation.

5.4.2 Summary of Comparison of Design Concepts

To compare the design concepts, a Pugh chart is used (shown below, Table 1), listing important design features that reflect the functional requirements as decision criteria. The linear tension motion control design is chosen as the datum since its operating principle and physics are well-known from its implementation in industry. The Pugh chart is populated by the discussion in the previous section, 5.4.1 Evaluation of Design Concepts Against Functional Requirements. In comparing the designs, the linear tension motion control design emerges as the accumulator design that best balances the different functional requirements.

Table 1: Pugh chart to compare and choose accumulator design concept based on interpreted functional requirements.

	Flexibility in Handling $rate_{in} \neq rate_{out}$	Managing Tension	Minimal Twisting	Less Complex	Footprint
Linear Tension Control Accumulator	0	0	0	0	0
Tapered Pulley	0	-	-	+	0
Center-Pull Winder	++	--	--	-	+

6. WiLMA Design

6.1 Overview

To preface the following discussion on the design of WiLMA, the nomenclature used throughout the paper is as follows:

- WiLMA is used interchangeably with the term accumulator.
- WiLMA is composed of two main components: the buffer and the tensioner.
    - The buffer refers to the linear tension control portion of the accumulator. In other words, the buffer refers to the module containing the idler pulleys and the linear rail, as discussed in a previous section (Section 5.3: Linear Tension Control Accumulator).
    - The tensioner refers to the module whose purpose is to maintain constant tension throughout the accumulator. To do so, there is a tensioner on the input of the buffer. This is discussed in a future section (Section 6.5: Tensioning Module).

6.2 First-Order Analysis

The following sections outline the first-order analysis performed to loosely define the design space for the machine.

6.2.3 Linear Carriage Force

In essence, the buffer is made of a carriage on which the moving idler pulley is mounted, the linear rail guides that constrain the carriage, the lead screw, and the stepper motor that drives the lead screw. A diagram labeling the parts of a stepper motor setup may be seen in Fig. 16. The design parameter of interest in the buffer setup is the force required to move the carriage as this

informs the stepper motor torque specification and the maximum length of the idler pulley since the pulley would essentially be cantilevered off the carriage.

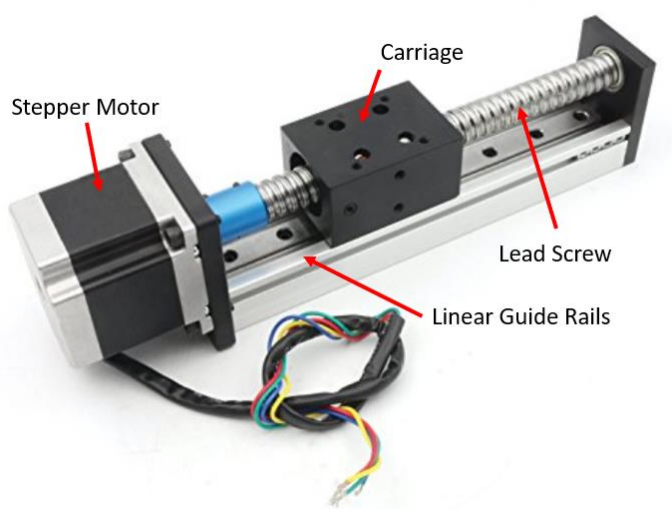


Figure 16: 100mm Length Travel Linear Stage Actuator with Square Linear Rails; components of a lead screw setup are labeled.

The force required to move the carriage comes from the stepper motor; the following equation shows the consideration of Conservation of Energy (torque in, force out) where the input is work done by the stepper motor while the output is the carriage movement.

$$\tau \cdot \theta = F \cdot x$$

τ is motor torque
θ is angle rotated by shaft
F is force on carriage
x is displacement of carriage

$$\tau \cdot 2\pi\eta = F \cdot l$$

η is motor efficiency
l is lead screw pitch

$$\tau = \frac{F \cdot l}{2\pi\eta} \qquad \text{(Eq. 3)}$$

Using the above equation (Eq. 3) and the values for pitch, conservative estimate for efficiency, and estimated mass of the carriage shown in the following table (Table 2), the conservatively estimated force required to move the carriage is approximately 20 N with a corresponding torque requirement of approximately 12.5 N*mm (1.25 N*cm).

Table 2: Values used for estimating the torque required by the stepper motor to move the linear carriage along the lead screw

Parameter	Value	Unit
Pitch of lead screw	2	mm
Estimated motor efficiency	0.5	-
Estimated mass of carriage	2	kg
Acceleration due to gravity	9.8	m/s^2

This required torque is well within the range of a common stock stepper motor. The specifications of the stepper motor used in this experiment, NEMA 23, may be found in Appendix C. Given the factor of safety of over ten on the torque requirement, the length of the idler pulley is not realistically limited by any additional "jamming" force from the fiber loading the tip of the idler pulley that may require additional motor torque to overcome. The following diagram illustrates the "jamming" of the carriage from excessive force applied by the fiber. In reality, the fiber would likely break prior to reaching a force that would stall the stepper motor.

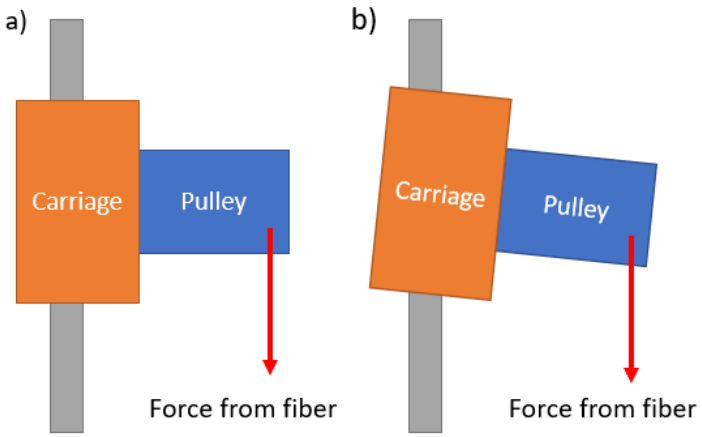


Figure 17: Diagram (side view) showing the potential "jamming" of the carriage on the lead screw in the event that excessive force is applied to the tip of the idler pulley by the fiber; assumes the carriage and the pulley are a rigid body.

6.2.4 Buffer Size

This section outlines how a preliminary estimate of the buffer size is obtained. This estimate helped determine the buffer parameters, such as vertical height of the buffer, the number of grooves on the idler pulley, the diameter of the idler pulley, etc. The tradeoff with having a large buffer capacity and longer run time in this design is that in order for the accumulator to effectively operate in steady state, there must be some amount of fiber initially produced and manually threaded through the accumulator prior to beginning the fabric machine's operation.

The buffer size is geometrically dependent on the diameter of the idler pulley, the center-to-center distance between the pulleys (denoted as h), and the number of windings around the idler pulley. Figure 18b helps visualize the stacked windings to derive the resulting equation for an estimate of the buffer size is:

$$x = (n-1)(\pi d + 2h) \qquad \text{(Eq. 4)}$$

Where x is the amount of fiber in the buffer, n is the number of windings around the idler pulley, d is the diameter of the idler pulley, and h is the center-to-center distance between the idler pulleys.

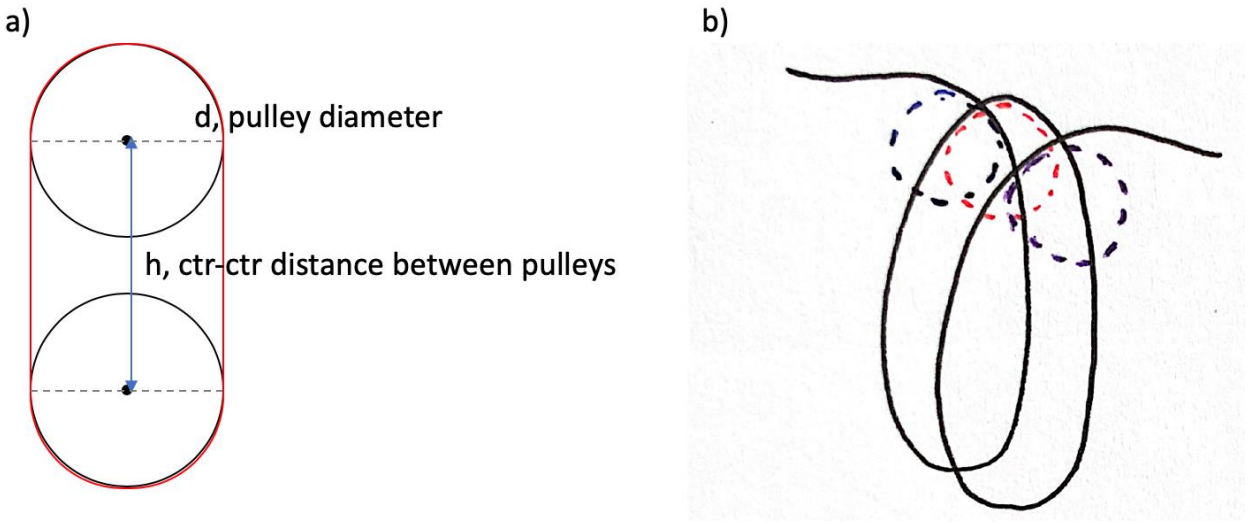


Figure 18: a) labeled front view of idler pulley b) sketch of isometric view of the buffer to help visualize and derive buffer equation (Eq. 4). The dotted circles represent grooves on the idler pulley.

With an idler diameter of 20 mm, the minimum center-to-center distance between the pulleys is set at 50 mm to allow a 10 mm safety buffer between the idler pulleys in the event that the

controls fail to stop the moving pulley, giving the operator time to manually stop the line before collision. With n = 4 windings, the calculated buffer minimum is 488.5 mm. Assuming the maximum distance between the pulleys is 250 mm, with n = 4, this gives a maximum buffer size of 1688.50 mm, or approximately 1.7 m, more than enough for the state of the current fabric machine.

6.3 WiLMA's Geometric Model

This section derives the model of WiLMA's motor behavior based on the geometry of the buffer. That is, the input is the difference in FrED's velocity and the fabric machine's velocity; the output is the speed command to WiLMA's stepper motor to take up the slack in the fiber produced by that velocity difference.

From the previous section, the amount of fiber in the buffer is described by Eq. 4:

$$x = (n-1)(\pi d + 2h) \quad \text{(Eq. 4)}$$

The change in the buffer amount divided by the change in time results in the difference in velocity. The term corresponding to the amount of fiber wrapped around the idler pulley itself, that is, the πd term, is constant through the experiment. Therefore, it is not included in the change in the buffer amount equation below. Equation 6 provides the geometric relation between the velocity difference between FrED and the fabric machine and the buffer amount.

$$\Delta x = (n-1)(2\Delta h) \quad \text{(Eq. 5)}$$

$$\Delta v = \frac{(n-1)(2\Delta h)}{\Delta t} \quad \text{(Eq. 6)}$$

The geometric change in the buffer corresponds to the change in the distance between the idler pulleys, which is driven by the stepper motor and the carriage movement along the lead screw.

$$\Delta h = \omega \cdot \Delta t \cdot l \quad \text{(Eq. 7)}$$

Where Δh is the change in the center-to-center distance between the idler pulleys [mm], ω is the angular velocity of the stepper motor [rev/s], Δt is the timestep [s], and l is the pitch of the lead screw [mm/rev].

To relate the angular velocity of the stepper motor to pulses, the unit of stepper motor velocity using the Tic T249 USB Multi-Interface Stepper Motor Controller (Pololu Part #: 3138):

$$\omega \left[\frac{rev}{s}\right] = p\left[\frac{microstep}{10^4 \cdot s}\right] \cdot m\left[\frac{full\ step}{microstep}\right] \cdot \frac{1}{j_{step}}\left[\frac{rev}{steps}\right]$$

ω is angular velocity of motor
p is motor speed in units of pulses
m is step mode of motor
$j_{step} = 200\ steps/rev$ for NEMA23

$$p = \frac{\omega \cdot j_{step}}{m} \cdot 10^4 \quad \text{(Eq. 8)}$$

Combining Eq. 6 and Eq. 7 by eliminating Δh and rewriting to relate the angular velocity of the stepper motor as a function of the difference in velocity:

$$\Delta v \cdot \Delta t = 2(n-1)(\omega l \Delta t) \quad \text{(Eq. 6)}$$

$$\frac{\Delta v \cdot \Delta t}{2(n-1)} = \omega l \Delta t \quad \text{(Eq. 6, 7)}$$

$$\omega = \frac{1}{l\Delta t} \cdot \frac{\Delta v \cdot \Delta t}{2(n-1)}$$

$$\omega = \frac{\Delta v}{2l(n-1)} \quad \text{(Eq. 9)}$$

Then substituting this function of difference in velocity for angular velocity into Eq. 8 to get the speed of the stepper motor (in the units of pulses) as a function of the velocity difference between FrED and the fabric machine:

$$p = \frac{\omega \cdot j_{step}}{m} \cdot 10^4 = \frac{j_{step}}{m} \cdot \frac{\Delta v}{2l(n-1)} \cdot 10^4 = \frac{j_{step}\Delta v \cdot 10^4}{2lm(n-1)} \quad \text{(Eq. 8, 9)}$$

$$p = \frac{j_{step}\Delta v \cdot 10^4}{2lm(n-1)} \quad \text{(Eq. 10)}$$

6.4 Design Iterations for Buffer

6.4.1 Buffer Revision A

Preliminary Design

For the first iteration of the buffer, modeled after the linear tension control accumulator used in industry, the linear guide rail is constructed out of 80/20 T-slot aluminum rails to adhere to the modularity requirement since the 80/20 rails are already used in the design of FrED. Starting with the moving idler pulley, a custom nut adapter is designed for the lead screw (red component in Fig. 19) that allows a surface on which to attach an off-the-shelf lead screw nut as well as a cavity on the perpendicular face to insert a bearing for smooth rotation of the idler pulley. The custom nut adapter is constrained to move only linearly by the 80/20 rails as the stepper motor drives the lead screw. Opposite the nut adapter is the other end constraint of the idler pulley (purple component in Fig. 19). Similarly, this custom component has an inset for the bearing, sliding along the wall constraint (transparent component in Fig. 19).

For the stationary idler pulley, it is mounted directly to the 80/20 rail structure. The mounting is done via custom bearing blocks (green component in Fig. 19), which mount to the 80/20 structure using off-the-shelf components compatible with the 80/20 T-slot framing system. This same bearing block may be used to cap off the lead screw on the end opposite the motor, as seen in the CAD (Fig. 19b). The stationary pulley is positioned at the top of the structure so that the moving pulley is lower down to address stability concerns. Similarly, the stepper motor is also at the bottom of the structure to decrease the effects of any vibrations that may be amplified with having a large mass high off the ground plane. The diameter of the idler pulley was chosen to fit in the 1-inch space defined by the 80/20 rail system, in particular for the stationary idler pulley. Note that the 1-inch 80/20 T-slot framing system is used, as opposed to a metric version, to maintain modularity since FrED was designed with the 1-inch 80/20 system. At this point, the groove width and the pitch of the grooves in the idler pulley are chosen arbitrarily as the design is in its preliminary stage.

CAD images of the individual custom components may be found in Appendix B.1.

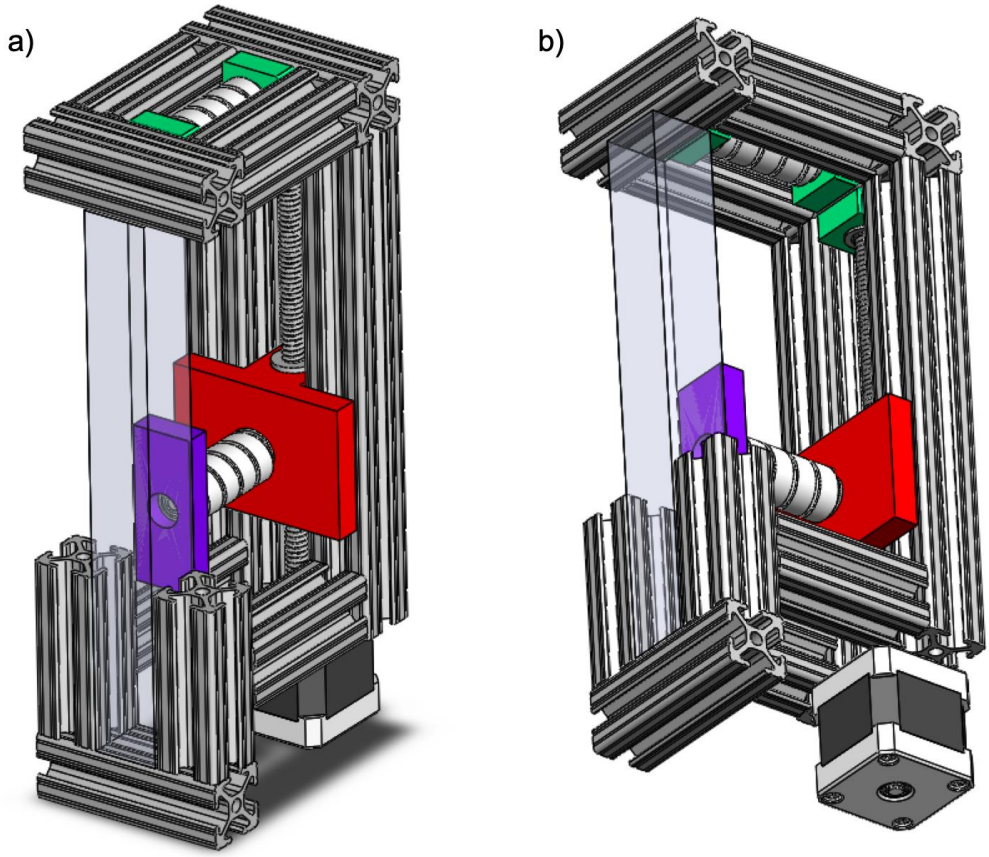


Figure 19: Preliminary CAD of Buffer Revision A, highlighting designed components. Individual CAD of custom components may be found in Appendix B.1.

Peer Design Review Feedback

The feedback outlined in the following section is courtesy of doctoral candidate David Donghyun Kim, a fellow member of the Device Realization Lab and designer of FrED:

- The boxy design may make it difficult to access the idler pulleys for manual threading of the fiber.
- Motor vibrations and lead screw alignment may become an issue since the stepper motor is sitting directly on the ground plane.
- Check that the diameter and pitch on the idler pulley ensure that the fiber will not be damaged from too small of a radius or slip out of the grooves easily.

6.4.2 Buffer Revision B

In the re-design to address the design review feedback, a comparison between the custom linear rail system proposed in Buffer Revision A and an off-the-shelf linear stage was done. The evaluation criteria were cost, performance, and scalability. Although the custom 80/20 rail system has more potential for cost reduction in "mass production," the custom rail system is not expected to perform as reliably as an off-the-shelf linear stage for two main reasons:

1) it does not include bearing elements in the linear rails as an off-the-shelf linear stage would have;

2) its performance is highly dependent on how well the 80/20 rails are assembled.

Given the nature of the buffer being part of an educational kit, the quality of the assembly cannot be ensured as it would be assembled by the students.

In low volumes, the cost of the custom 80/20 rail system is comparable to a generic linear stage. Therefore, the design of the buffer moved forward with replacing the custom system with a generic, off-the-shelf linear stage. The specific one used, Fig. 16, includes a NEMA 23 stepper motor (specifications may be found in Appendix C) and a 5 mm pitch lead screw, suitable for the buffer needs as checked by first-order analysis (explained in Section 6.2.3: Linear Carriage Force).

With the choice to use an off-the-shelf linear stage, the sliding constraint is no longer needed. This change helps address the difficulty of manually threading the fiber through the buffer in Revision A. The shaft and bearing combination in Revision A is replaced with a threaded shoulder bolt to keep the idler pulley in place. The shoulder bolt threads directly into the nut adapter (pink component in Fig. 20) on the stepper motor. This design change decreases the part count as well, replacing the sliding constraint (purple component in Fig. 20), the constraining wall (transparent component in Fig. 20), as well as the bearing that would have been inset into the sliding constraint. The nut adapter mounts directly to the nut of the linear stage, which came with threaded mounting holes. A shoulder bolt is chosen for its ability to handle shear load, in the event that the system is over-tensioned and the idler pulleys apply shear force on the shoulder bolts. Similarly, a shoulder bolt is used to secure the stationary idler pulley. The bolt threads into a custom top plate (yellow component in Fig. 20).

To mitigate effects of motor vibration, a custom motor mounting plate is designed to have mounting holes that allow the plate and the motor to be mounted to 80/20 legs to lift the motor off the ground. As for misalignment, the motor coupler that originally came with the stepper motor is replaced with flexible shaft coupling for high-parallel misalignment (McMaster-Carr part: 9889T1) to address any misalignment in the assembly. Lastly, the diameter of the idler pulley is set to 20 mm to be consistent with the diameter of the spool used in FrED. This larger diameter will be able to accommodate fiber materials that are more resistant to bending and less flexible overall.

CAD images of the individual custom components may be found in Appendix B.2.

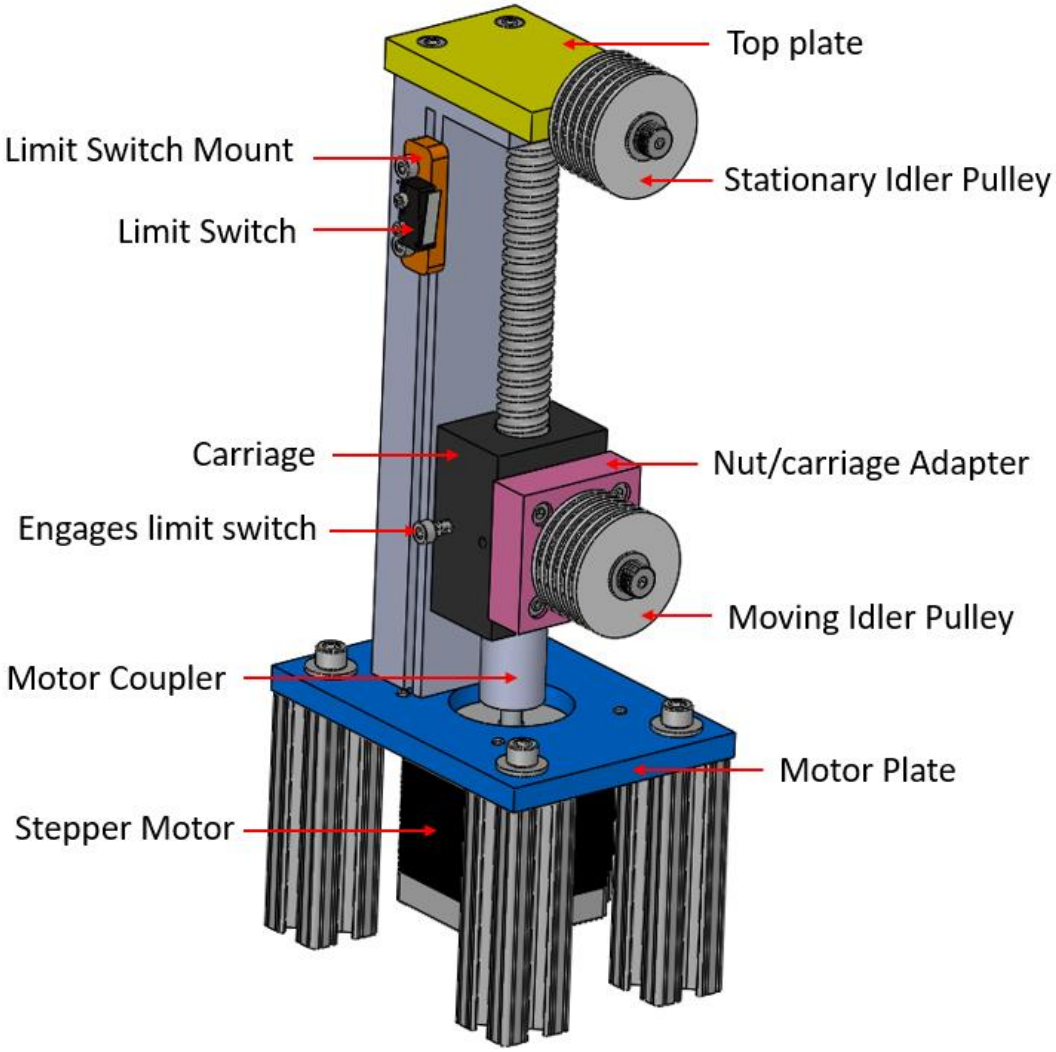


Figure 20: CAD of Buffer Revision B, labeling designed components. Individual CAD of custom components may be found in Appendix B.2.

6.5 Tensioning Module

The purpose of the tensioning module, or tensioner, is to assist in maintaining relatively constant tension of the fiber as it passes from FrED, through WiLMA, and to the fabric machine. A tensioner is not traditionally included in the industrial accumulators used in 3D printing filament production because the filament is not as sensitive to tension changes and the diameter of the 3D printing filament is kept constant, as described in Section 3.2: 3D Printing Filament Spools. However, the fiber winding through WiLMA may contain tension-sensitive components or embedded materials and the diameter may be varied dynamically in the production process. Therefore, a tensioner is needed on either the input to the buffer, the output, or both. The following sections describe the ideation, prototyping, and chosen design for the tensioning module.

6.5.1 Design Concepts

The following figure (Fig. 21) shows a sketch of some design concepts considered for the tensioner. The main functional requirement of the contact tensioner is to help regulate the tension of the fiber by “pushing” into the fiber to increase contact and hence tension or “withdraw” from the fiber to decrease tension, similar to the gate-type tensioner (Fig. 7). This movement to increase and decrease the amount of contact with the fiber translated to a linear movement component of the tensioner. This linear movement can be achieved actively with mechanisms such as a lead screw or rack and pinion, or it can be achieved through a passive system, such as a spring-loaded linkage mechanism. To minimize complexity in terms of controls as well as cost for the kit, the chosen design for the tensioner passively applies force to the fiber using a spring, so that a motor and a separate controller for that motor are not needed.

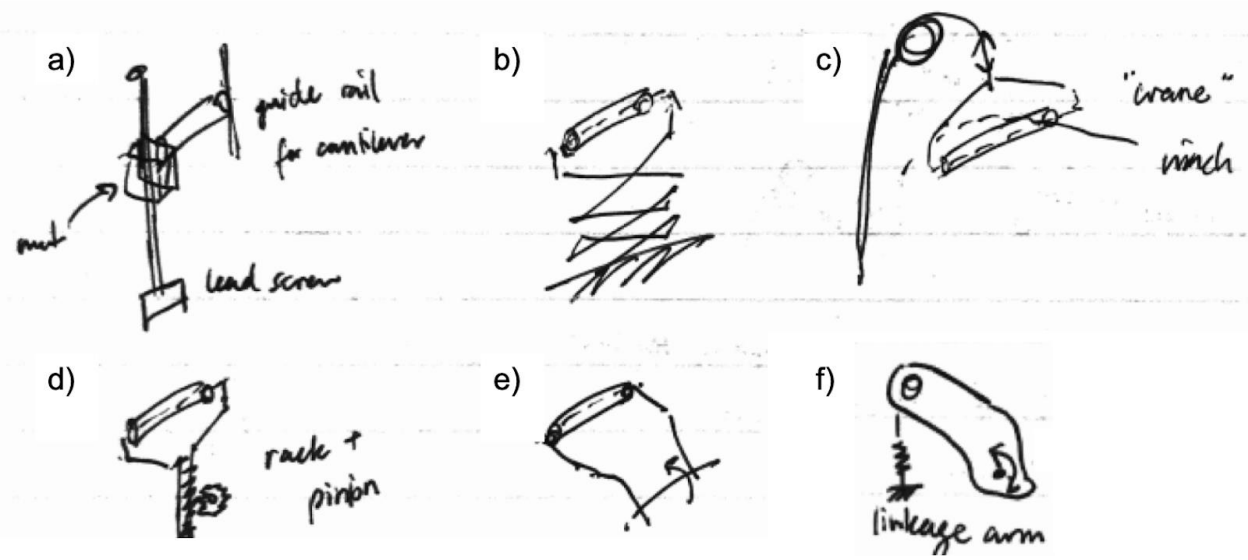


Figure 21: Tensioner Design Concepts: a) actively driven by lead screw; b) move linearly via scissor lift mechanism; c) move linearly via winch mechanism; d) actively drive tensioner via rack and pinion mechanism; e) linkage mechanism; f) spring-loaded linkage mechanism

6.5.2 Tensioner Revision A

Concept

Moving forward with the passive spring-loaded mechanism design, a torsional spring is used instead of the compression spring seen in the preliminary sketch (Fig. 21f) for a more accurate estimate of spring force over a larger range since the linkage motion is rotational. If a compression spring was used, the calculated force based on the measured height *y* (Fig. 22) would only be accurate in the range where the spring is being compressed or stretched perfectly vertically. However, since the linkage motion is rotational, this range of solely vertical movement by the spring is limited; once the linkage rotates outside of this limited range, the behavior of the spring force is no longer linear, making it more difficult to model accurately. An estimate of the spring force is necessary to provide the controller feedback on the tension; the feedback controller is described in more detail in Section 8.2: Controller Design. In short, a laser distance sensor, or time-of-flight sensor, can be mounted to the base of the tensioner module to measure the distance between the linkage arm and the base, allowing an estimate of the angle of the spring and hence the spring force. A diagram of the spring-loaded linkage mechanism is shown in Fig. 22, followed by the calculation of spring force based on the position of the linkage arm, or dancer.

For nomenclature purposes, the tensioner module consists of a dancer component, the main spring-loaded linkage mechanism, as well as a positioning idler (discussed in Section 6.5.3: Tensioner Revision B). The purpose of the positioning idler is to guide the fiber through the dancer as it comes off from various locations on the length of the spool.

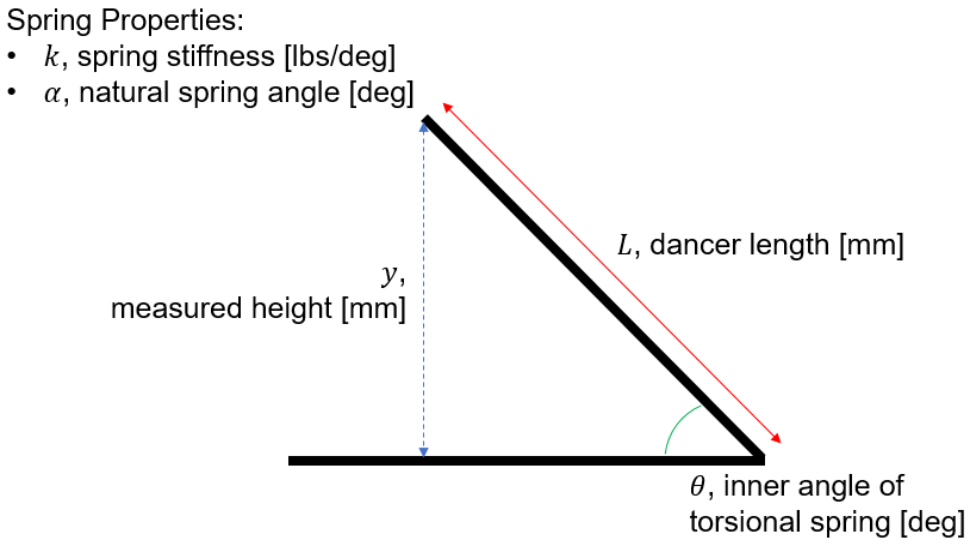


Figure 22: Diagram of spring-loaded linkage mechanism physics (dancer part of tensioner)

To relate the angle of the dancer to the spring angle:

$$\theta = \sin^{-1}\left(\frac{y}{L}\right)$$

y is measured height by sensor [mm]

L is dancer length [mm]

To relate the spring angle to the force applied by the torsional spring:

$$F = k(\alpha - \theta) \qquad \text{(Eq. 11)}$$

Where F is the force applied to the fiber by the spring/dancer, k is the torsional spring stiffness, α is the natural spring angle of the torsional spring, and θ is the inner angle of the torsional spring.

Preliminary Design

A CAD for the preliminary design of the dancer can be seen in Fig. 23. The dancer is composed of two halves: the bottom half of the dancer includes mounting holes to mount to the 80/20 frame structure while the top half includes a mounted sleeve bearing to interface with the fiber. For the top half of the dancer, the pieces were designed to be all planar so that it could be manufactured with a laser cutter.

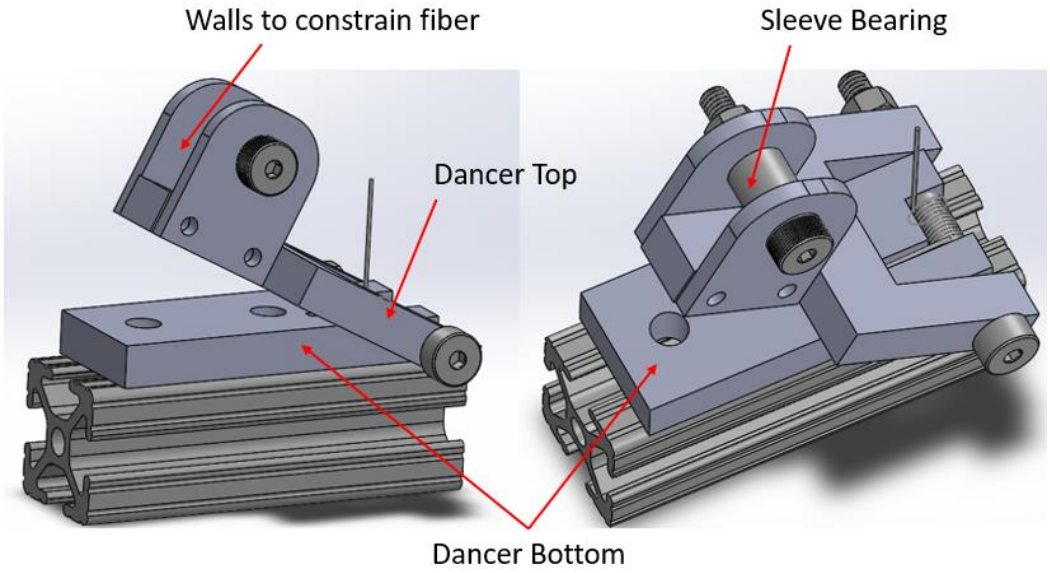


Figure 23: Preliminary CAD of Dancer Revision A

Design Feedback

The key feedback provided by doctoral candidate David Donghyun Kim, a fellow member of the Device Realization Lab and designer of FrED, is that there may be a risk of the fiber being caught in between the sleeve bearing and walls that constrain it since there is some nominal tolerance to allow free rotation of the sleeve bearing as the fiber passes over. Secondly, in considering the scalability of this design, in particular the assembly process and tolerance stack-up of each planar component, it was decided that the top half of the dancer may be better designed as a single piece, with the intent of injection molding for mass production.

From a prototype of the tensioner, it was also found that the time-of-flight sensor used (Adafruit Product ID: 3316) performed more reliably in reading the position of the dancer top when there is more surface area of the dancer over the sensor. In other words, for the next iteration, the dancer top will be wider to fully cover the time-of-flight sensor and minimize noise from its surroundings.

6.5.3 Tensioner Revision B

To address the key concern of the fiber being caught in the gap between the sleeve bearing and the bearing's constraints, the bearing is changed to a custom part with a grooved shape (dark grey component in Fig. 24). The grooved bearing is made of nylon to provide a low resistance surface for both the fiber to pass over and for the bearing itself to easily rotate on the shoulder bolt. The groove is sufficiently deep to provide the fiber from "skipping" out. Additionally, since the angle of the fiber going into WiLMA varies depending on where along the length of the mockup spool the fiber is leaving, a positioning idler (brown component in Fig. 24), a curved component to guide the fiber into the dancer, is added. Similarly, on the output of WiLMA, the fabric machine is not grabbing the fiber at the same location since it depends on which needle is activated. Therefore, a positioning idler is also used on the output of WiLMA; the final arrangement of the tensioning module in the assembly can be seen in Fig. 26b.

Based on the prototyping results, the dancer top is widened to fully cover the sensor. As well, the length of the dancer top is also increased to 45.25 mm to increase the range of motion for the torsional spring that can still be picked up by the sensor. As mentioned in the previous section of design feedback, there were some scalability concerns with the planar components, in particular concerning assembly, tolerance stack-up, as well as the features themselves being relatively small, such as the slot for the spring leg. As such, revision A of the dancer is likely difficult to scale reliably and cheaply. Therefore, for the purposes of this experiment, revision B of the dancer is 3D printed. Heat-set inserts are installed into the 3D-printed dancer bottom to secure the sensor. To scale for production, the custom dancer components can be easily injection molded.

The assembly of the final tensioner module can be seen in the following figure. CAD images of the individual custom components may be found in Appendix B.3.

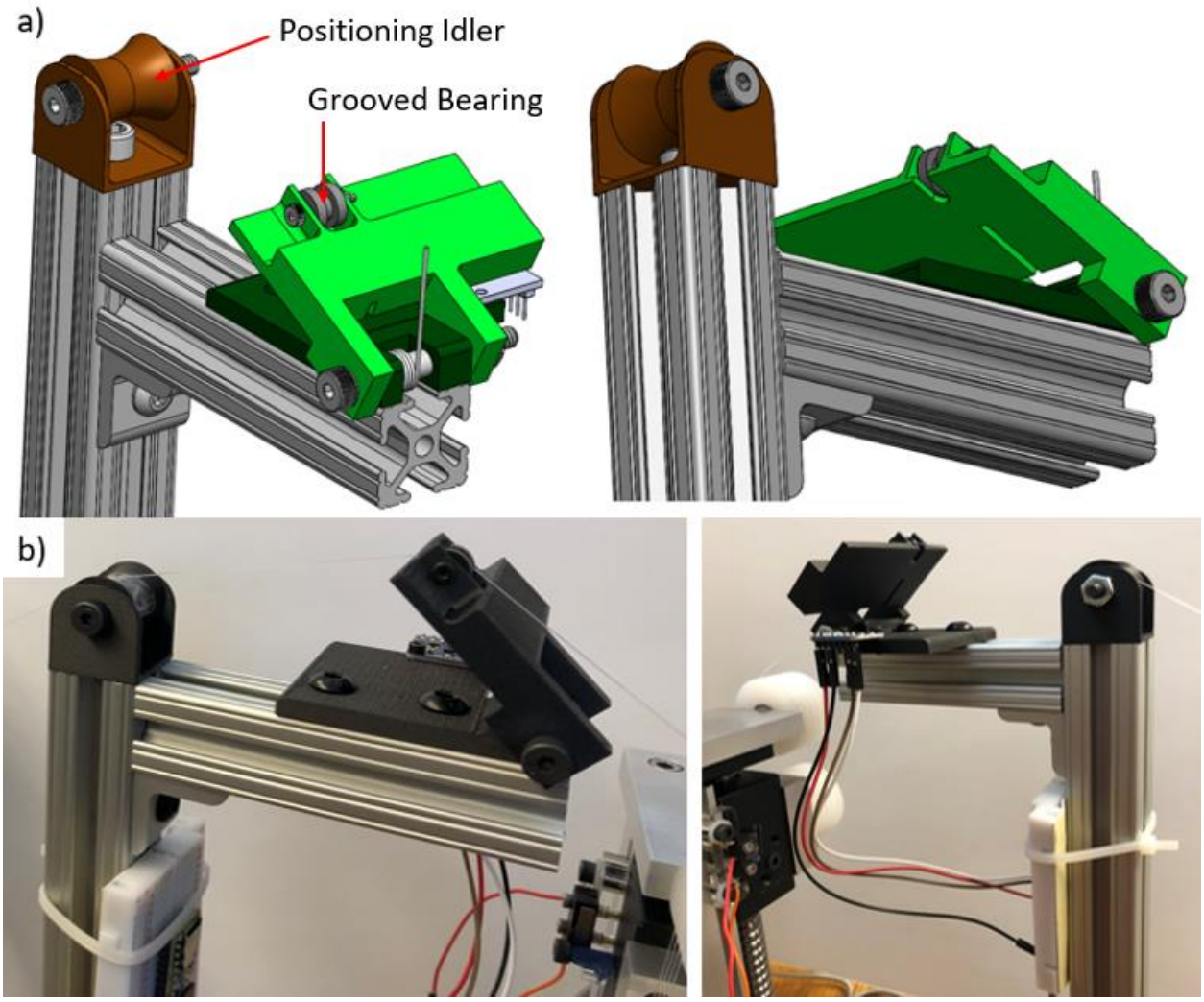


Figure 24: a) CAD of Tensioner Revision B. Individual CAD of custom components may be found in Appendix B.3; b) photos of the actual tensioner assembly

The required spring stiffness of the torsional spring in the tensioner is empirically estimated using a spring scale on PMMA solid fiber and polycarbonate fiber samples. The preload/tensioning force was chosen to be small enough so that the fiber is not permanently deformed but large enough to take up the slack since it is more rigid than typical string or yarn.

7. Testing Setup

In order to test the effectiveness of WiLMA as an accumulator, mockups of FrED and the fabric machine were designed and built. This provides a testing platform for WiLMA that is independent of the development of the fabric machine as it is still in progress at the time of this writing. Additionally, the mockup module allows for flexibility and control over varying both the input and output parameters for the accumulator.

7.1 Design of Mockup Module

In essence, both FrED and the fabric machine can be modeled as a machine that is outputting fiber at some speed and a machine that is taking up fiber at some speed, respectively. Therefore, the simplest form of both machines that effectively models their behaviors is a spool driven by a DC motor. Since the motor-driven spool is not expected to experience large loads and does not require significant torque, a belt drive is used for its simplicity, as opposed to chains or gears.

Since WiLMA will be receiving feedback via the tensioner, the absolute velocity by the belt drive transmission does not have to be exact. As such, the main design consideration in choosing the belt drive design parameters is its reliability. The actual velocity of each mockup motor will be logged continuously for use in comparing WiLMA's behavior to the geometric model that is dependent on the difference between the mockups' velocities (Eq. 10). The belt design and loop size were chosen to minimize the footprint of the mockup modules, maintain reasonable eccentricity of the spool, and ensure that the belt and gears will not noticeably wear in the operating lifetime of the mockup modules. Wear of the belt and gears is not a significant concern as the mockup modules run at relatively low speeds and for short periods of time. The resulting design of the mockup module can be seen in Fig. 25.

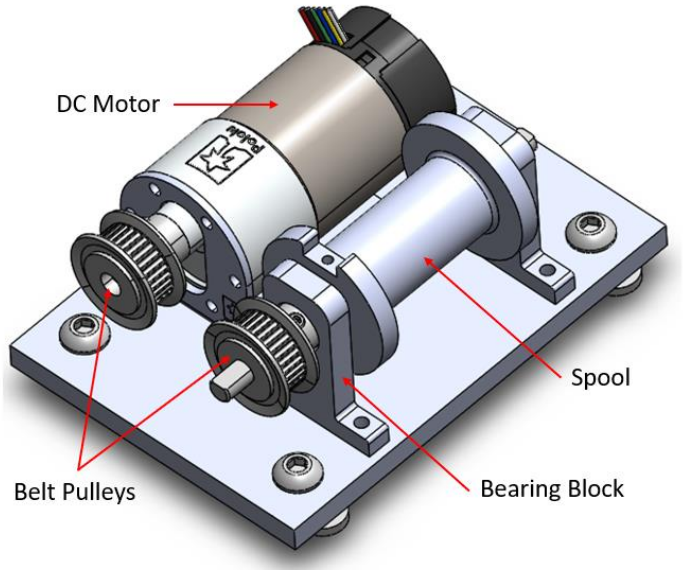


Figure 25: CAD of spool system in mockup module (belt not pictured), using a spool belt-driven by a DC motor to mimic operation of FrED and the fabric machine

The mockup module was designed such that it is largely composed of off-the-shelf parts or custom parts with mostly planar features to reduce manufacturing cost. Starting with the belt-drive system, pulleys, belts, and motor mounts were all sized according to the motor and the spool diameter. The DC motor used in the mockup module is a 131:1 metal gearmotor (Pololu part 4756). This motor was chosen as it satisfies the functional requirements of the mockup module in terms of rotational velocity range and torque; it is also one of the motors used in the FrED system. Therefore, the choice of this motor keeps the unique part count in this educational kit at a minimum. The motor is attached to the mockup plate using the off-the-shelf motor mount provided by Pololu (Pololu part 1084).

Similarly, the spool diameter was chosen to match the diameter of the spool used in the FrED system. On one of the flanges of the spool is a hole for a set screw (seen in Fig. 25). Having the set screw on the flange provides easier access during assembly. The shaft is machined with a D profile on both ends to provide the set screw with a greater contact surface. The shaft, attached to the spool via set screw, is supported on both ends by bearings. The bearing blocks were custom designed for the appropriate height. The rectangular profile is intended to provide multiple surfaces for fixturing as necessary in the machining process to reduce manufacturing cost. The spool assembly mounts to the mockup plate via these bearing blocks. Individual CAD of the custom spool and bearing blocks may be found in Appendix B.4.

The mockup plate itself contains only planar features: clearance holes for ease of assembly and tolerance in tensioning the belt as the system is installed. In addition to the holes to mount the belt drive system, the mockup plate contains holes to mount to the 80/20 framing system, consistent with the theme of modularity across all the machines in this production line. This allows flexibility in rearranging the structure of the mockup as necessary to connect with WiLMA. As shown in the following figure, Fig. 26a was the original design of the mockup for an earlier iteration of WiLMA that did not include tensioning modules. As WiLMA evolved, between the parts available for WiLMA and the mockup arrangement in Fig. 26a, no new parts were used to create the final arrangement in Fig. 26b.

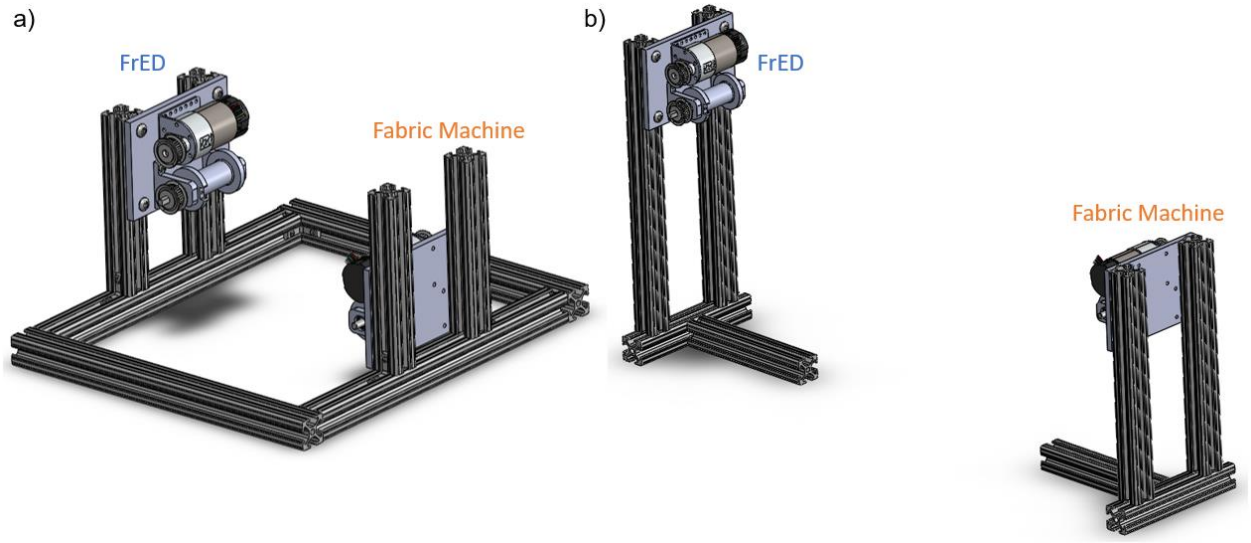


Figure 26: Demonstrating modularity of mockup module with different arrangements

7.2 Implementation of Mockup Module

As shown in the previous figure, the mockups' structures have evolved to accommodate the final implementation of WiLMA. A photo of the actual setup may be seen below (Fig. 31). A comprehensive bill of materials for the two mockups may be found in Appendix A.3. A comment from the assembly process that may be addressed for future implementations is that the belt is fairly tight because the location of the holes in the mockup plate were designed for the purchased belt's nominal dimensions, as in line-to-line with no gap tolerances. This was driven by prior knowledge from FrED's belt-driven system where the belt was too slack and caused some wear on the belts and pulleys themselves. Though the belts in the mockup modules are tight, there is no sign of wear from operation. However, it is recommended for future implementations to add some

tolerance, i.e. increase the distance between the mounting holes for the motor and the mounting holes for the spool, to ease the process of assembling the belt-drive system.

Through performance testing of the DC motors, it was found that the motors have a lower rotational speed limit of 0.1 rev/s (linear velocity of 5 mm/s with a 20 mm diameter spool), due to the 131:1 gear ratio. As a result, all tested velocities for fiber intake and fiber output will be above this lower threshold. The mockup modules must be commanded the appropriate duty cycle to rotate the spools at certain velocities for testing WiLMA. In order to achieve the approximate velocity, an Arduino script involving a simple closed loop proportional controller is used to map a desired rotational or linear velocity to the approximate duty cycle. The Arduino script may be found in Appendix D. The velocity to duty cycle map may be found in the following figures. Note that this characterization of the DC motors was done under no load. As mentioned previously in Section 7.1: Design of Mockup Module, the absolute velocity of the mockup module is not important since WiLMA's controller only uses the feedback from the tensioner.

In order to compare against the feedback controller (described in Section 8.2: WiLMA's Controller), the actual velocity measured by the DC motors in the mockups corresponding to FrED and the fabric machine are recorded. The following equation is used to convert the raw encoder reading of steps per second to millimeters per second, the units of the Δv input to Eq. 10.

$$\Delta v = \frac{\pi d_{spool}}{j_{DC}} \cdot \left(\frac{\Delta steps}{\Delta t}_{M1} - \frac{\Delta steps}{\Delta t}_{M3} \right) \qquad \text{(Eq. 12)}$$

Where d_{spool} is the diameter of the spool used in the mockup (Section 7.1: Design of Mockup Module), $\Delta steps$ is the recorded change in position by the motor's encoder, Δt is the change in time, and j_{DC} is equal to 8400 counts or steps per revolution for the specific DC motor used (Pololu part 4756).

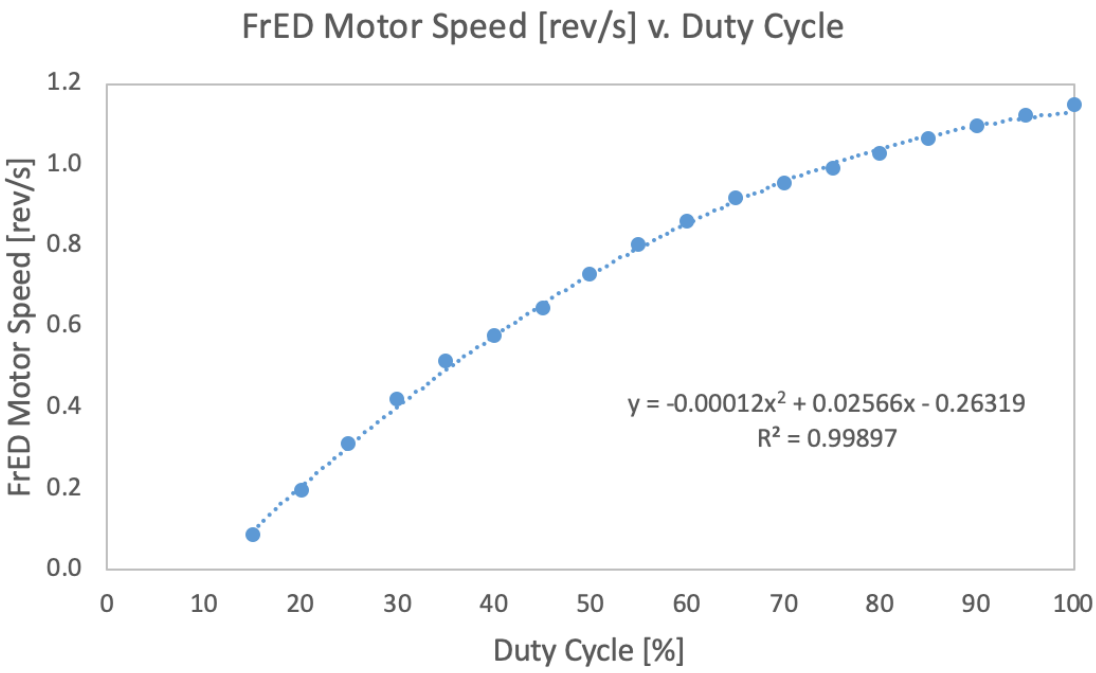


Figure 27: DC motor characterization (rotational velocity to duty cycle map) for the Pololu 131:1 Metal Gearmotor (Pololu part 4756) used in the mockup for FrED

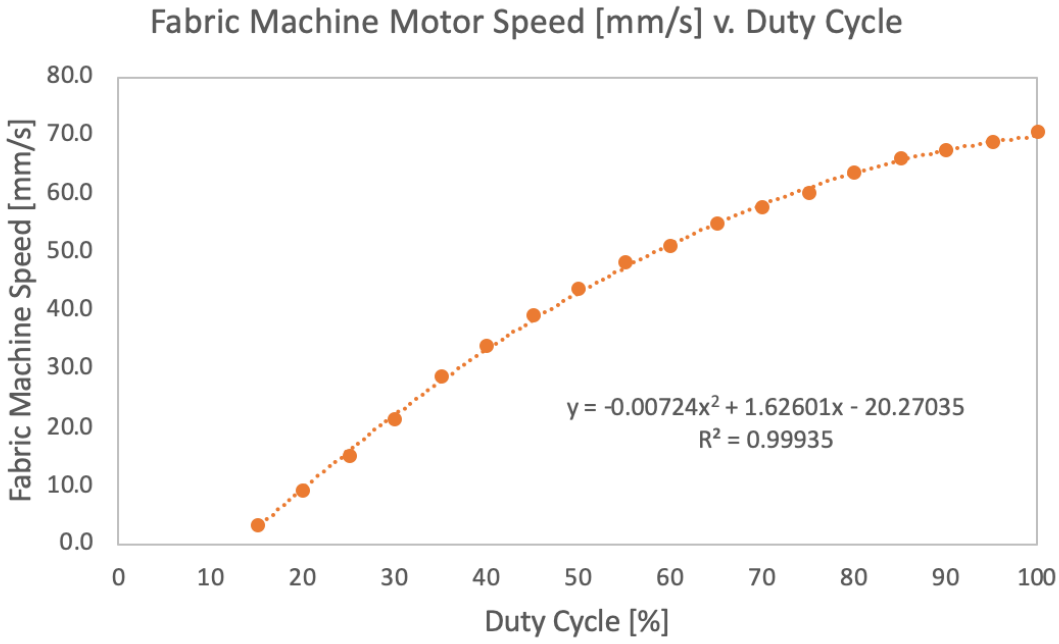


Figure 28: DC motor characterization (linear velocity to duty cycle map) for the Pololu 131:1 Metal Gearmotor (Pololu part 4756) used in the mockup for the fabric machine

7.3 Master Assembly of Testing Setup

The testing setup includes an assembly of the mockup of FrED, WiLMA (buffer and tensioner), and the mockup of the fabric machine. The following figures show the master assembly in CAD as well as in its physical form.

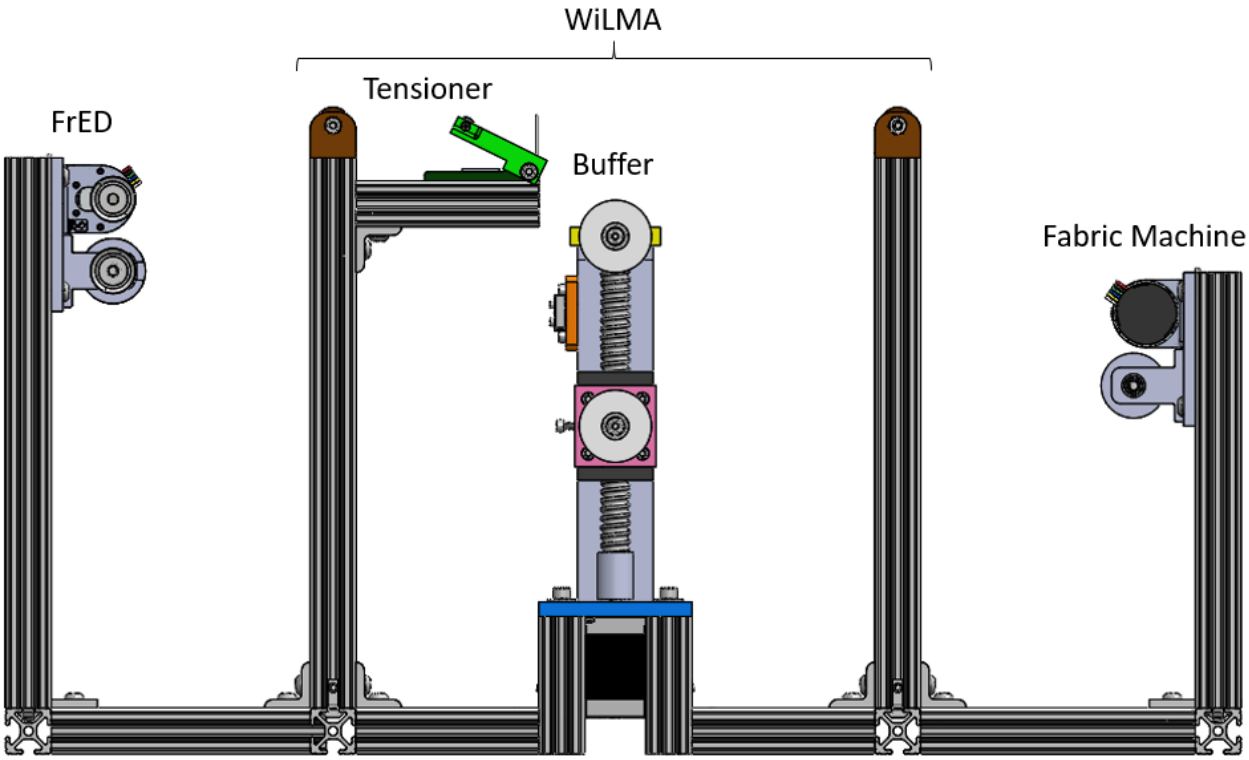


Figure 29: Front view of master assembly CAD used for testing WiLMA using mockup modules to represent FrED and the fabric machine

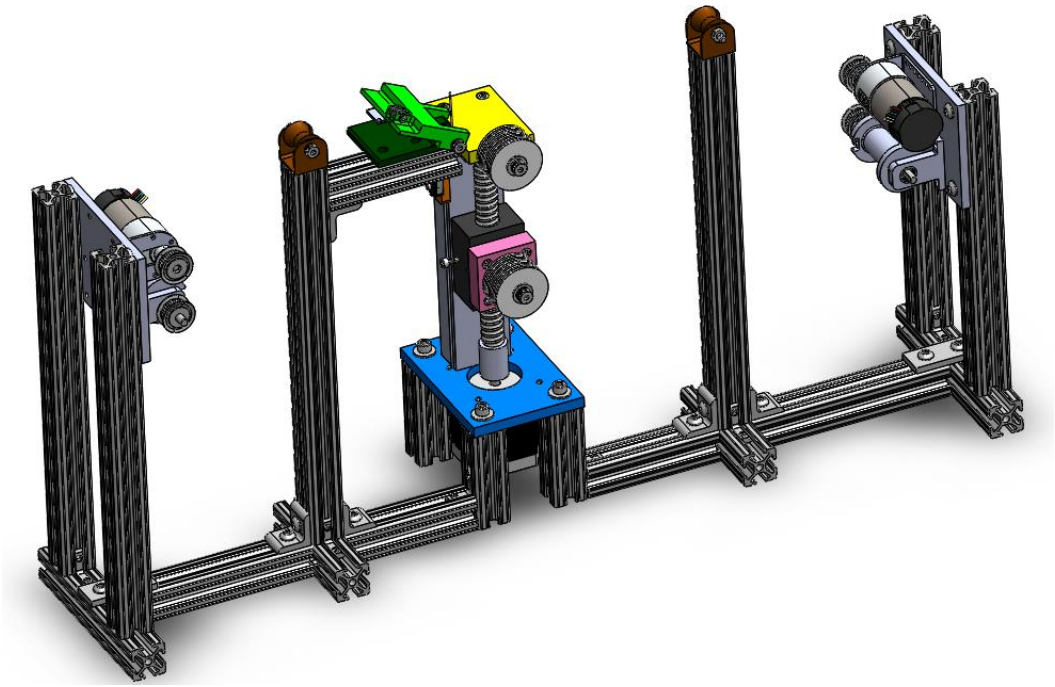


Figure 30: CAD of master assembly for testing WiLMA using mockup modules to represent FrED and the fabric machine

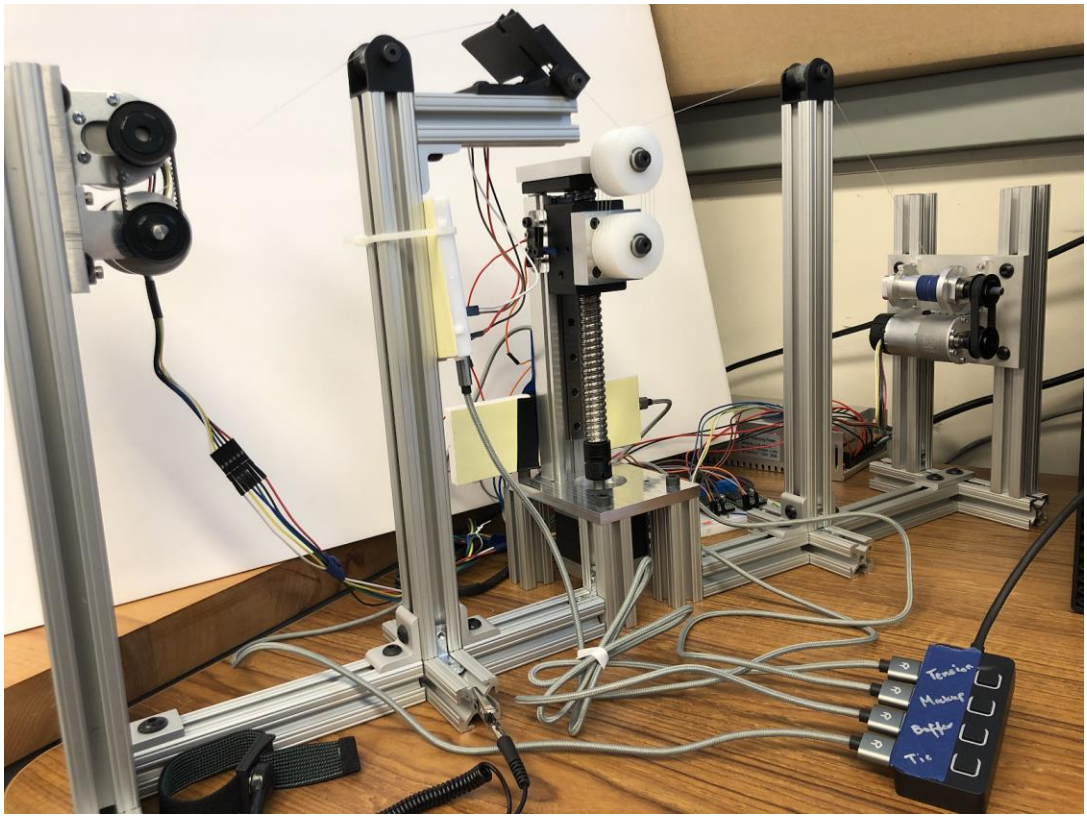


Figure 31: Physical hardware of master assembly for testing WiLMA using mockup modules to represent FrED and the fabric machine

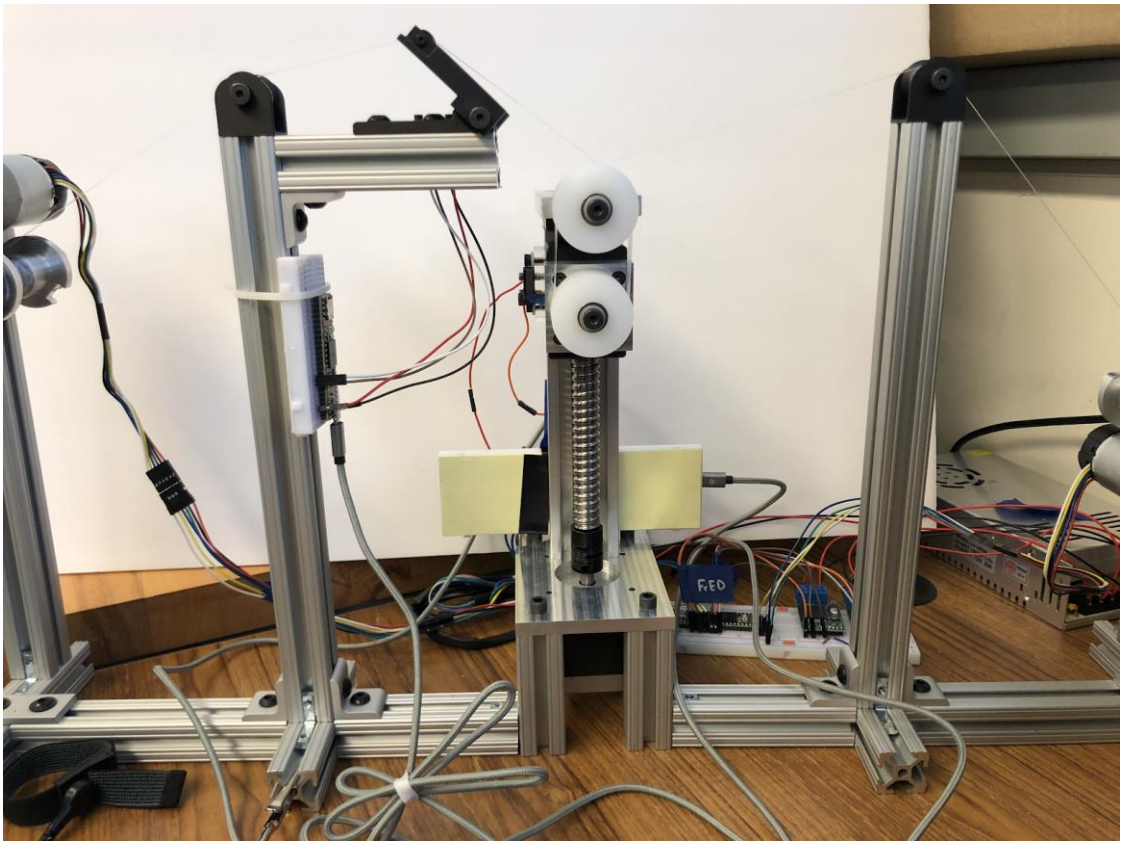


Figure 32: Front view of the physical prototype of WiLMA

8. Controls

8.1 Feedback Control Background

A feedback, or closed loop, control system is used as the controller for WiLMA. This section briefly outlines relevant background information on controls theory as well as associated terminology.

8.1.1 Closed Loop Control System

A feedback, or closed loop, system refers to a system that receives information feedback, typically from a sensor, to drive the controller. The sensor monitors the *process variable*: "the system parameter that needs to be controlled, such as temperature, pressure," or in the case of WiLMA, spring force (proxy for tension) [20]. The goal of the controller is to use the feedback from the sensor to reach and maintain the set point of the process variable; the *set point* is "the desired or command value for the process variable, such as 100 degrees Celsius in...a temperature control system" [20]. In the case of WiLMA, the set point is a set force, unique to the fiber material, that is the preload force necessary to hold tension in that fiber while not damaging the fiber.

A *block diagram* is often used in controls to illustrate the interactions between the different parts of a controls system. A block diagram for a typical closed loop system may be seen in the following figure (Fig. 33), courtesy of National Instruments [20]; note that in the block diagram, the compensator is synonymous with controller.

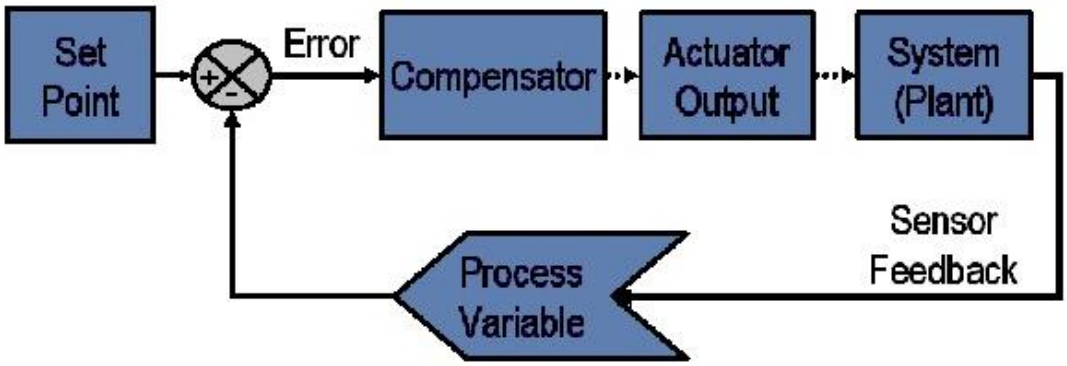


Figure 33: Block diagram of a typical closed loop system, courtesy of National Instruments [20]

Additional relevant terminology to describe a system's response and controller's performance are:

- *Error*: the difference between the measured value of the process variable and the set point
- *Settling time*: the amount of time it takes for the process variable "to settle to within a certain percentage (commonly 5%) of the final value" [20]
- *Overshoot*: the amount that the process value overshoots the final (steady-state) value
- *Steady-state error*: the difference between the final (steady-state) value of the process variable and the set point

In general, the ideal response minimizes all of the aforementioned response characteristics: error, settling time, overshoot, and steady-state error. The following figure (Fig. 34), courtesy of National Instruments, demonstrates the response of a typical closed loop system with the above terms graphically defined.

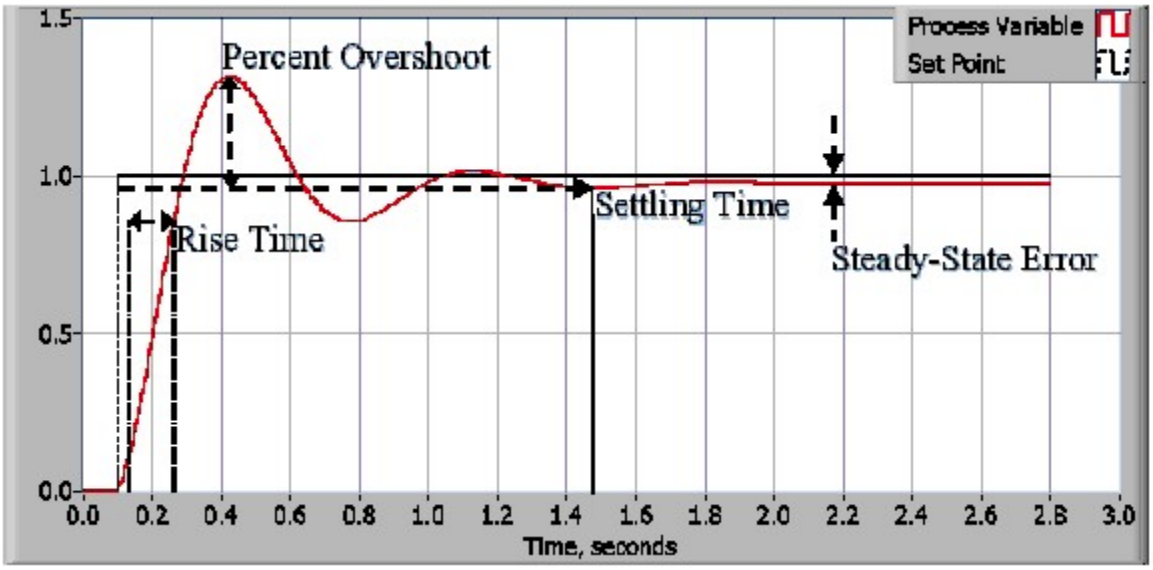


Figure 34: Response of a typical closed loop system, courtesy of National Instruments [20]

8.1.2 PID Theory

A PID controller is "the most common control algorithm used in industry and has been universally accepted in industrial control" [20]. As the name suggests, a PID controller, or compensator, is the sum of three components: proportional, integral, and derivative. A typical PID controller (boxed in red) can be seen in the following figure (Fig. 35).

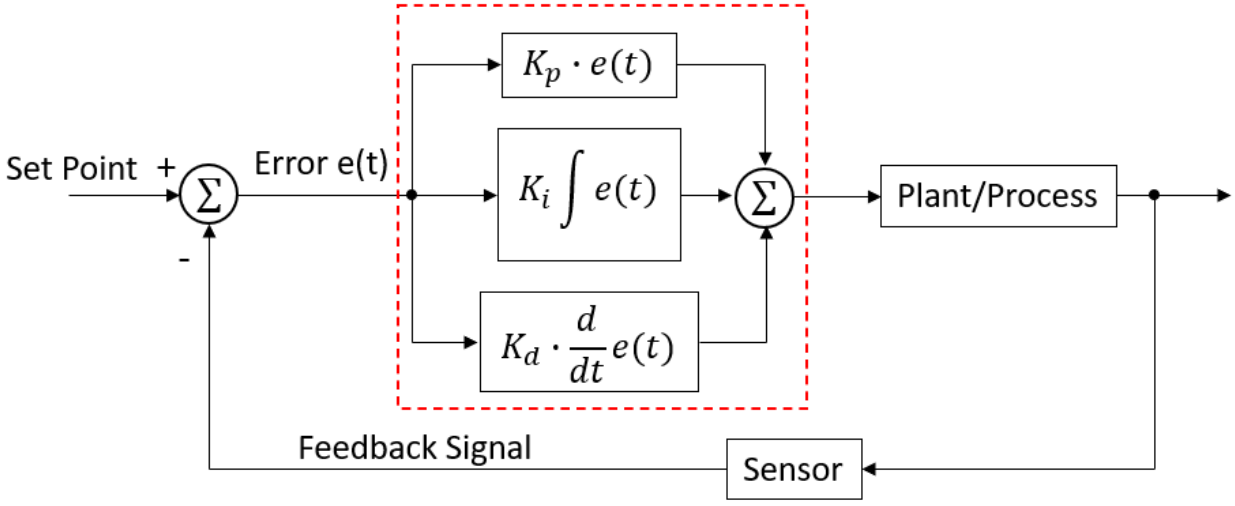


Figure 35: Block diagram of a typical PID controller in a feedback control system

The proportional component is directly proportional to the error signal, scaled by a gain denoted as K_p. The proportional gain is typically the gain that is first optimized in the system to achieve a stable system response that is stable enough to optimize the other two gains. In general,

the proportional component should contribute the bulk of the output change; hence, increasing K_p typically increases the speed of the control system's response.

The integral component is proportional to the sum of the error over time, scaled by a gain denoted as K_i. The integral component will increase over time unless the error is zero. Therefore, the integral component can be tuned to drive the steady-state error to zero. In addition to minimizing steady-state error, increasing K_i often minimizes settling time but tends to increase overshoot.

The derivative component is proportional to the rate of change of the error (proxy for the rate of change of the process variable), scaled by a gain denoted as K_d. Increasing K_d typically improves settling time, but the derivative component is also closely tied to the stability of the system. The derivative component is seldom included in practice because of its susceptibility to noise (potentially large changes in error in a short period of time) and hence its impact on the stability of the system. Therefore, in this experiment, the controller used for WiLMA is a PI controller, excluding the derivative component.

8.2 WiLMA's Controller

8.2.1 Controller Design

The process variable in this accumulator system is the spring force applied to the fiber, acting as a proxy for the tension in the fiber. Tension is the parameter of interest since the most important functional requirement is to not damage or mechanically stress the fiber. This requirement translates to keeping the variation in the fiber tension at a minimum as the fiber travels through the accumulator. As described in Section 6.5.2: Tensioner Revision A – Concept, the force is measured using a time-of-flight sensor to measure the distance of the dancer top from the dancer bottom, hence the angle, from which the force applied by the torsional spring can be calculated. The set point for the force is dependent on the fiber material and the fabric machine settings. In this experiment, the set point ranged from 0.7 lbs to 0.8 lbs (3.1 N to 3.6 N) for the nylon monofilament and the PMMA fibers used.

As mentioned previously, a PI controller is used for WiLMA, excluding the derivative component as it is sensitive to disturbances. However, in the case of the fabric machine running a square profile, in addition to the traditional proportional and integral component of the controller, a component scaled by K_s is included. The gain K_s scales the amplitude of the square wave to help

compensate for the sudden change in the difference between FrED's output and the fabric machine's intake. When the fabric machine is running a constant profile, the K_s term is not activated; in other words, K_s is zero. A block diagram illustrating the WiLMA's controller (boxed in red) and how it interacts with the accumulator feedback system is shown in the following figure (Fig. 36). The Δv is the effectively the amplitude of the square profile in this experiment.

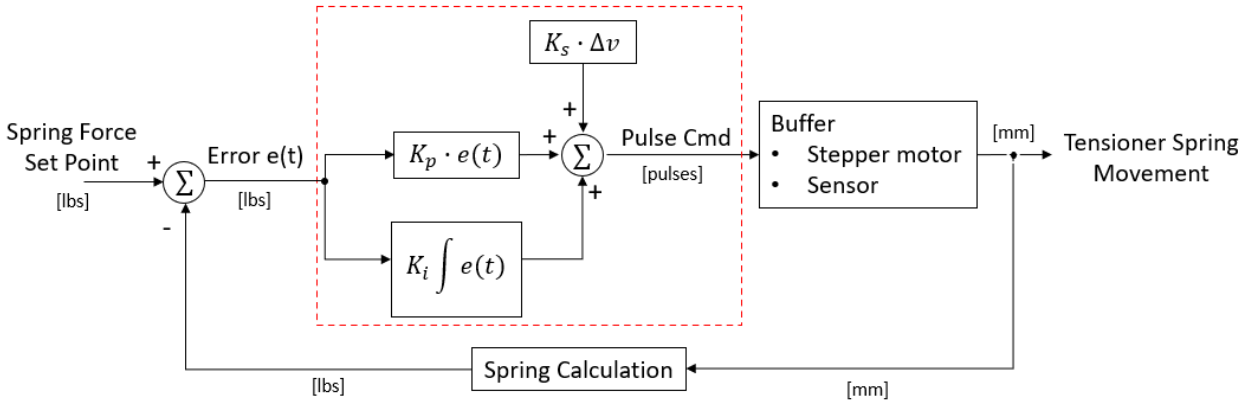


Figure 36: Block diagram of the WiLMA feedback control system for maintaining fiber tension; the modified PI controller is boxed in red, where the K_s term is only activated if the fabric machine is running a square profile

Note that there is only a singular tensioner on the input of the buffer. During the initial prototyping stage, it was determined that a second tensioner on the output of the buffer does not act independent of the first tensioner. Therefore, the decision was made to proceed with a single tensioner on the input of the buffer and assume that the tension does not significantly change as it passes through the buffer system.

8.2.2 Gain Optimization and Troubleshooting

To achieve a stable system response, which is defined as a trial running to completion (a set amount of time in each trial), the first gain tuned is K_p since it constitutes the bulk of the controller output. The initial guess for the K_p gain is 1e7 [pulses/lbs] because the expected pulse command based on the geometric model is on that order of magnitude. The K_p gain is incrementally increased until the system is able to run a trial to completion without the tensioner ever losing contact with the fiber. The settling time was noted in the various test trials; when the system is stable, the settling time is always less than 5 timesteps. The final optimized gain that allows the system to remain stable for all settings and fibers used in this experiment is K_p = 4.5e7 [pulses/lbs].

The integral gain K_i is tuned next to help correct the steady-state error. Similar to choosing an initial value for K_p, K_i was initially guessed to be on the order of 10^7. However, after a few test runs, it was found to cause instability in the system until K_i was in the range of 10^4. When the K_i value was such that the trial was able to run to completion, upon inspection of the trial run data, it can be seen that the error appears to reach a steady-state (approximately timestep 100 in Fig. 37) before beginning to go unstable and diverge sinusoidally.

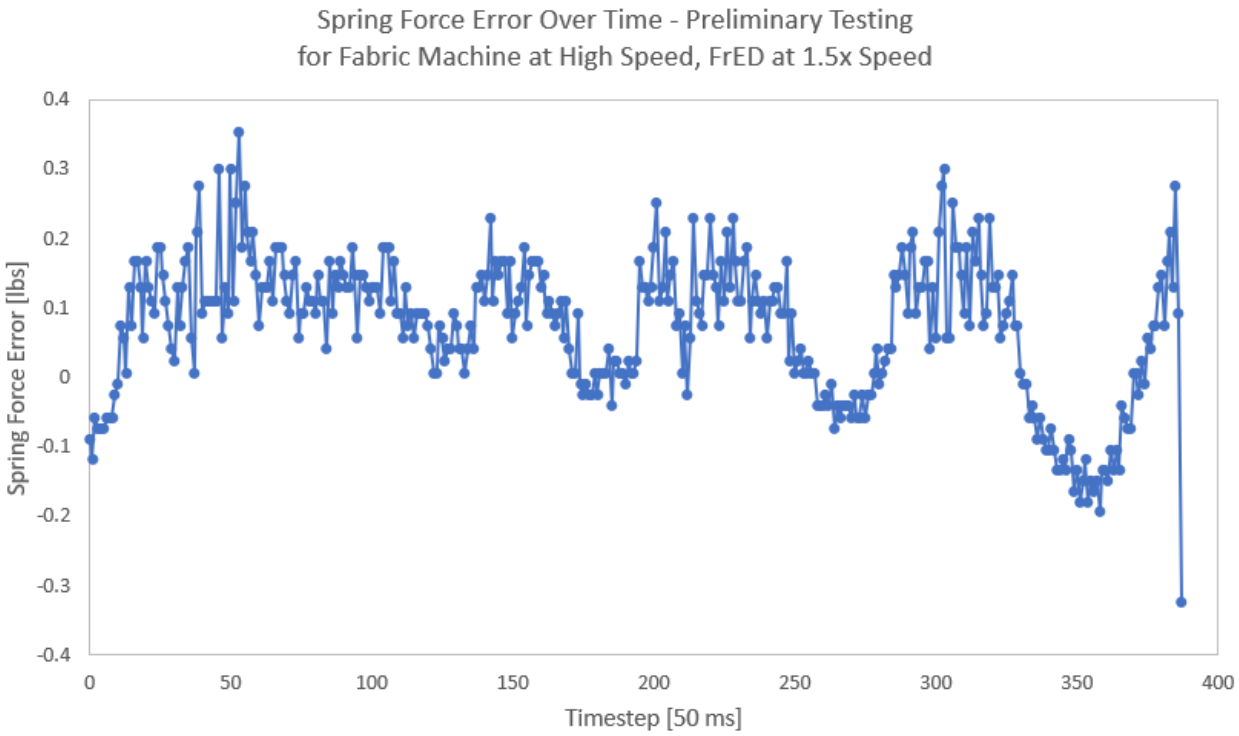


Figure 37: Plot of error in the spring force growing sinusoidally after appearing to reach a steady-state under the following conditions: 50 ms timestep, K_p = 3.7e7 [pulses/lbs], K_i = 1.25e5 [pulses/lbs], high speed fabric machine nominally set at 50 mm/s and FrED set at 1.5x the speed of the fabric machine

This characteristic of the system response seemed to suggest that the buffer was unable to respond fast enough to the feedback from the tensioner and the controller. As a result, the speed command of the buffer by the controller was plotted against the actual velocity of the buffer, shown in the following figure (Fig. 38), which showed a delay of 4 timesteps (200 ms total) between the command and the actual velocity. As the sum of the error grows over time, the lag between the command and the actual velocity causes the system to overshoot in both the positive and negative directions. For example, as the buffer moves to reach the commanded velocity, the controller has

already received feedback for the next command before the buffer was able to act on the last command. This delay causes the buffer to overshoot in one direction, which prompts the controller to command a large velocity in the opposite direction to try and return to the tension set point, hence the growing oscillations. To correct this delay, the frequency of the controller feedback is slowed to allow the buffer enough time to react. The timestep is gradually increased from 50 ms to 100 ms, at which point the delay between the command and the actual velocity is consistently one timestep.

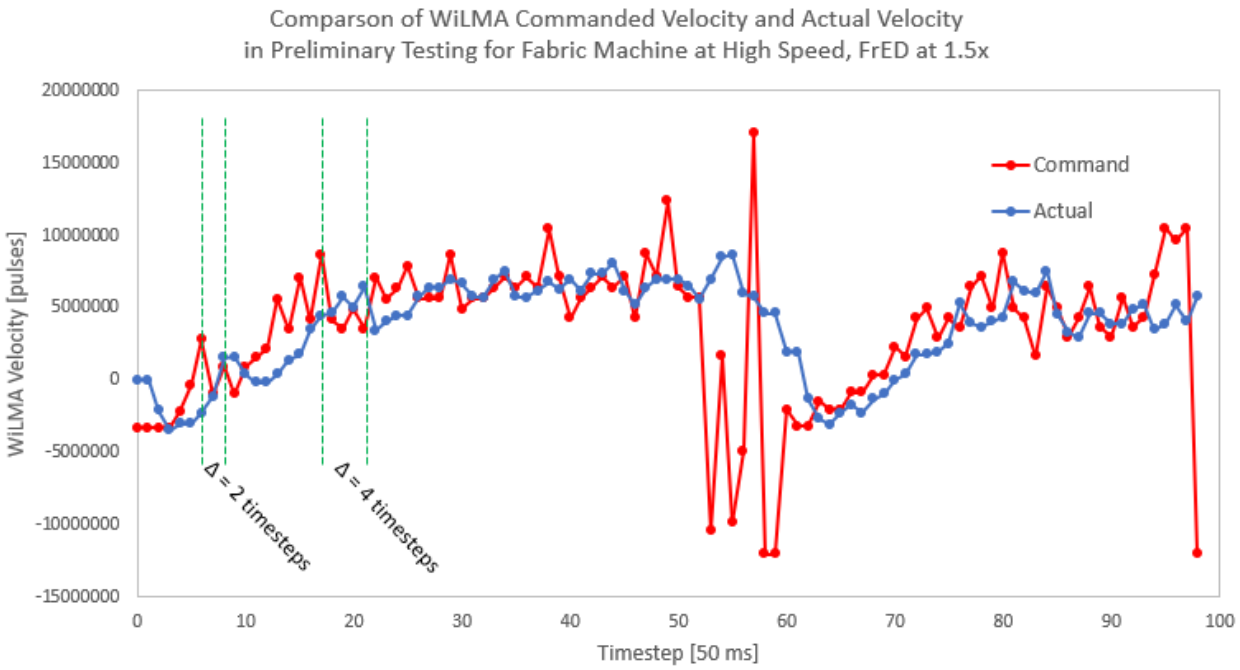


Figure 38: Plot to compare the commanded velocity for WiLMA's stepper motor and the actual velocity, highlighting the delay; data obtained under the following conditions: 50 ms timestep, K_p = 3.75e7 [pulses/lbs], K_i = 1.75e4 [pulses/lbs], high speed fabric machine nominally set at 50 mm/s and FrED set at 1.5x the speed of the fabric machine

8.3 Robot Operating System (ROS)

8.3.1 Background and Terminology

Robot Operating System, or ROS, is "an open-source, meta-operating system for your robot" [21], originally created by PhD students at Stanford University to build the "Linux of Robotics" and eliminate the wasted time in robotics research spent on reproducing software from previously published papers [22]. The motivation for ROS is succinctly illustrated in the following comic:

Figure 39: Comic presented in the IEEE Spectrum article titled "The Origin Story of ROS, the Linux of Robotics". Original caption: Comic we later commissioned at Willow Garage, from Jorge Cham, to illustrate the wasted time in robotics R&D, [22]

The main goals of ROS are to: be thin (as a result, easy to integrate), have ROS-agnostic libraries, be language independent (ROS-based libraries for common programming languages such as Python and C++), have easy testing (built-in test/integration framework), and be scalable (appropriate for large runtime systems and development processes). To date, ROS runs on Unix-based platforms [21].

Some relevant terminology include: nodes, messages, topics, publishers, and subscribers. A *node* is a process that performs computation. In a robot control system, with the modularity of ROS, there can be many such nodes. For example, one node could process a laser range-finder sensor, one could be responsible for path planning while another controls the wheel motors. Messages enable communication between these nodes; a *message* is "simply a data structure, comprising typed fields," such as integers, floating point, etc. [23]. *Topics* are essentially the names of the messages that are used to identify the content of the message. ROS introduces a topic as logically similar to "a strongly typed message bus. Each bus has a name, and anyone can connect to the bus to send or receive messages as long as they are the right type" [23]. In order to connect to this "bus" or topic, a node must have a publisher, subscriber or both. As the name suggests, if a node contains a *publisher* for a given topic, that node is able to publish, or send, messages for that topic. Similarly, if a node contains a *subscriber* for a given topic, that node is able to subscribe, or receive, messages for that topic. Nodes are not limited by the number of publishers or subscribers it can contain, so "there may be multiple concurrent publishers and subscribers for a single topic" [23].

The relationship between these key concepts is shown in the following diagram (Fig. 40); an example of a ROS graph, which can be generated for debugging purposes. Essentially, nodes are represented by ovals, topics are represented by rectangles, and the messages are along the arrows from a node to a topic. The arrows also provide information on the publisher and subscriber status; an outgoing arrow from a node represents a publisher for that topic while an incoming arrow to a node represents a subscriber for that topic.

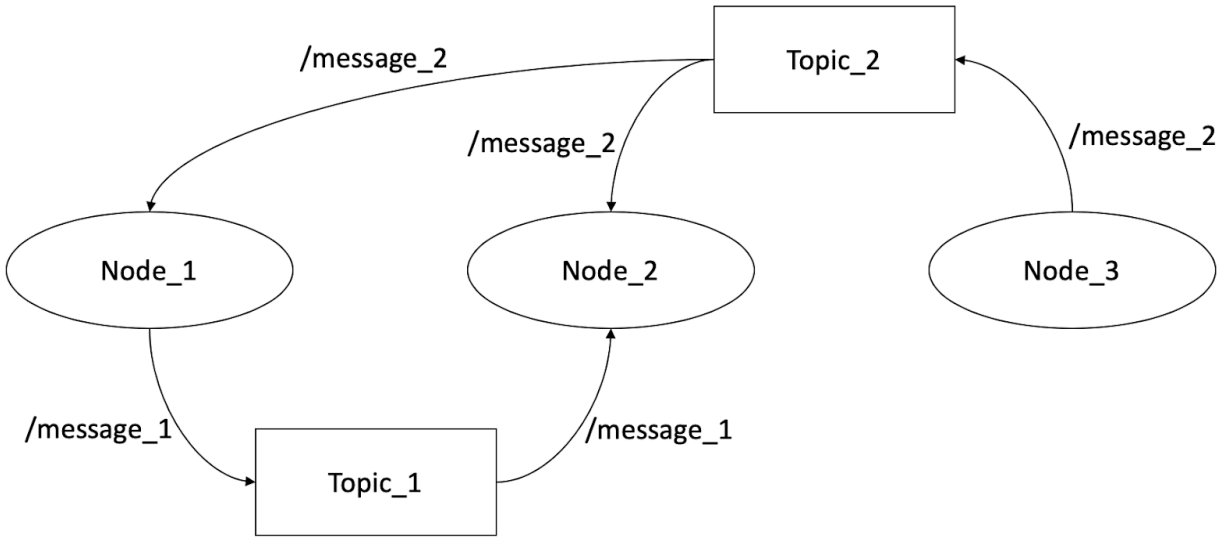


Figure 40: Example diagram of ROS graph with nodes, topics, and messages labeled; outgoing arrows represent publishers while incoming arrows represent subscribers

8.3.2 ROS for WiLMA

Choice of ROS

An appealing characteristic of ROS is the fact that it is language independent, compatible with Python and C++; this allows ROS to be used to implement systems that involve multiple microcontrollers, as in the case of WiLMA. With all the microcontrollers existing as nodes in the same ROS system, it is easier to sync the timing of each nodes' actions.

Also, WiLMA is designed to be a machine that is a part of the smart manufacturing production line that includes FrED and the fabric machine. For his S.M. thesis, Sangwoon Kim, also a member of MIT's Device Realization Lab, explored the use of deep reinforcement learning to control and vary the diameter of the fiber produced by FrED. Ultimately, his novel control method is working towards the potential to create fibers that have diameter modulations along its length, providing additional functionality – "smart fibers" [24]. The fact relevant to the development of WiLMA is that this novel controller is implemented with ROS software. To achieve the vision of a fully integrated smart manufacturing production line, it is thus ideal to maintain a common software across platforms that enables communication between the various machines in the line. This cross-communication will also provide more data for better dynamic feedback control.

Implementation of ROS

ROS is used for the mockup modules as well as WiLMA, to better enable cross-communication and imitate the controllers that will eventually be implemented for FrED and the fabric machine. As a result, there are three ROS nodes for this experiment: brain (Python-based node that reads the tensioner module data and performs the feedback control for WiLMA), buffer (Teensy microcontroller controlling the stepper motor), and mockup modules (Teensy microcontroller controlling both DC motors). The relationship between the nodes and their respective topics are shown in Fig. 41.

The brain node is run from the computer's terminal and is not directly controlling any motor. The only hardware that the brain node is directly interfacing with is the tensioner module. The brain node reads the time-of-flight sensor's range output via the serial port as the tensioner's sensor uses a Teensy microcontroller's USB port to connect to the computer. This step of the process does not involve ROS software. Additionally, the brain node functions as a control center, performing the calculations based on the tension feedback to output the speed that the buffer should move at to accommodate. The main callback function in the brain node that publishes the buffer's speed is periodically called with a ROS timer to ensure constant and known publishing rate. In addition to acting as WiLMA's PI controller, the brain node also logs all of the system's data (listed in Section 9.3.1: Summary of Data Collected).

The buffer node is the Teensy microcontroller that controls the stepper motor via the Tic T249 stepper motor controller (Pololu part 3138). The buffer node subscribes to the speed command for WiLMA, which is published by the brain node. In return, the buffer node publishes the position of the carriage on WiLMA to be logged by the brain node. The position feedback serves two main functions: 1) proxy for speed since the time step is constant; 2) limit control to track the position of the carriage so that the brain node knows when to publish a "stop" command when the buffer has reached either its lower or upper limit.

The mockup modules' node is the Teensy microcontroller that controls the DC motors used in both the FrED mockup and the fabric machine mockup. The message received by this mockup node contains the duty cycles for FrED and the fabric machine. In turn, the mockup node publishes the actual velocities of the DC motors as well as the difference in velocity between the two DC

motors as measured by their encoders. This velocity difference is logged by the brain node to compute the speed command based on WiLMA's geometric model.

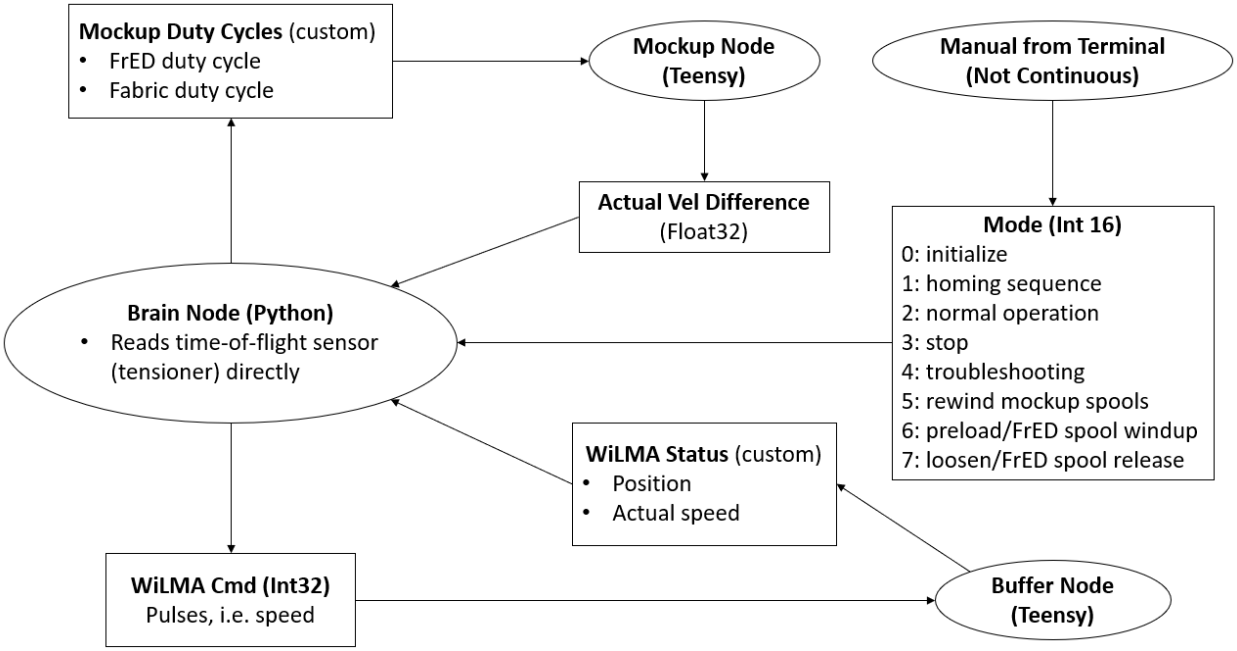


Figure 41: ROS graph for WiLMA, showing nodes, topics, and message types; the arrows indicate the publisher and subscriber relationships.

Based on the logic of each node, the sequence of actions by the nodes are triggered by the brain's callback function that publishes WiLMA's motor speed command. Therefore, all the nodes are essentially "clocked" so that their respective actions, whether running a motor or sending information, occur within the set time step that follows the period dictated by the brain's callback function.

Note that Teensys were chosen as the microcontrollers used in this experiment mainly because of its large memory size. For example, Arduino Unos and Arduino Nanos do not have enough dynamic memory for queueing the messages to be published and/or buffering the messages that are received. In addition to the large memory size, the Teensy 3.5 microcontroller used for this experiment has four sets of SDA/SCL pins, used for I2C serial communication, whereas the Arduino-branded boards typically only have one set. Described in the Further Work Section (Section 12.1), the additional sets of SDA/SCL pins may be used to reduce the number of microcontrollers from four to two – decreasing overall cost and number of parts.

9. Experimental Design

The experiment is centered on quantifying the effectiveness of WiLMA as an accumulator that buffers the speed difference of the fiber flow upstream and downstream of it. As such, the input parameters are the fiber output speed of the mockup module representing FrED and the fiber intake speed of the mockup module representing the fabric machine. The measured parameter of interest is the spring force applied by the tensioner to the fiber, effectively the tension of the fiber, as it enters the accumulator.

9.1 Mockup of FrED's Operational Profile

The operational profile of FrED is simply the mockup unspooling the fiber at a constant velocity. This corresponds to FrED being set to produce a fiber of constant diameter since the spooling speed directly relates to the fiber diameter in the version of FrED shown in Section 2.2: Overview of FrED. As mentioned previously, work by Sangwoon Kim for his S.M. thesis involves a dynamic controller for FrED that explores the potential of creating fibers that have diameter modulations along its length. Therefore, future work may involve FrED outputting fiber with a time-varying profile.

For reference, according to data obtained by Sangwoon Kim on FrED, the velocity range of approximately 0.1 rev/s to 1.4 rev/s yields fiber of diameter ranging from 0.3 mm to 0.6 mm. This fiber diameter is desirable for the fabric machine knitting as well as other potential applications for FrED. Therefore, the mockup of FrED will run at a constant rotational velocity within this expected range, above the lower threshold of stability for the DC motor, and faster than the fabric machine.

For this experiment, FrED will operate at a range of rotational velocities, 1.1 times to 3 times the operating velocity of the fabric machine, depending on whether the fabric machine is low speed or high speed.

9.2 Mockup of Fabric Machine's Operational Profile

As described in Section 2.3: Overview of Fabric Machine, the mechanisms of the fabric machine will likely result in fiber being taken from the accumulator in discrete steps. For the mockup of the fabric machine, the stop-and-go nature of the mechanisms is modeled as a discrete time-varying profile (square wave). Prior to testing WiLMA with a time-varying fabric machine mockup, WiLMA is tested with the fabric machine mockup pulling fiber at a constant velocity that is less than FrED's output velocity. A constant operational profile for the fabric machine mockup helps establish a baseline for WiLMA's performance.

Once the baseline performance for WiLMA has been established with a constant profile for the fabric machine mockup, WiLMA will be tested against a number of time-varying profiles as it better resembles the expected behavior of the fabric machine. For this experiment, a square wave is selected as the profile type to model the cyclical and stop-start nature of the fiber machine's operations (each needle pulling the fiber through and the stage resetting for each new row). WiLMA is tested using square waves with amplitudes at low speeds (10 mm/s) and high speeds (30 mm/s) as well as varying frequencies (0.1 Hz and 1 Hz). The amplitude and frequency ranges are chosen based on the estimates provided by David Donghyun Kim, who is currently developing the fabric machine, as well as considering the limits of the hardware.

The amplitude is limited by the DC motors in the mockup while the frequency is limited by the stepper motor's speed and response to feedback. The stable linear velocity range for the DC motors used in the mockup module is approximately 5 mm/s to 70 mm/s with a 20 mm diameter spool for linear velocity. Therefore, the low speed amplitude of the fabric machine is set at 10 mm/s while the high speed amplitude is set at 40 mm/s. For the high speed setting, the maximum ratio tested between FrED and the fabric machine is 1.5 since that results in FrED's motor hitting the upper limit of the DC motors. As for the frequency, as explained briefly in Section 8.2: WiLMA's Controller, the controls system's sampling rate is once every 100 milliseconds, or 10 Hz. Therefore, the frequencies tested are 0.1 Hz, to most realistically the fabric machine, and 1 Hz, to see how the behavior of WiLMA may differ with a tenfold change in frequency.

9.3 Data Collection Process

9.3.1 Summary of Data Collected

In summarizing the profiles described in the previous section, the data collected in this experiment includes the following settings with 20 trials each using a nylon monofilament as the test fiber:

- Constant profile for fabric machine
    - Low speed – fabric machine at 10 mm/s
        - FrED speed: 1.25x, 1.5x, 2x, 3x
    - High speed – fabric machine at 40 mm/s
        - FrED speed: 1.25x, 1.5x
- Square profile for fabric machine
    - Low speed – fabric machine amplitude at 10 mm/s
        - Frequencies: 0.1 Hz, 1 Hz
        - FrED speed: 1.5x
    - High speed – fabric machine amplitude at 30 mm/s
        - Frequencies: 0.1 Hz, 1 Hz
        - FrED speed: 1.5x

For all trials, the following relevant parameters are continuously logged:

- Spring position as read by the tensioner's time-of-flight sensor [mm]
- Spring force, calculated based on tensioner's sensor reading [lbs]
- Error in spring force [lbs]
- Sum of the error over time [lbs]
- PI controller's velocity command for WiLMA [pulses]
- WiLMA's actual velocity [pulses]
- WiLMA's position [step]
- Linear/tangential velocity of FrED's mockup, measured using the mockup DC motor's encoder [mm/s]
- Linear/tangential velocity of the fabric machine's mockup, measured using the mockup DC motor's encoder [mm/s]
- The difference in velocity between the mockup motors [mm/s]
- WiLMA's calculated velocity based on the geometric model (Section 6.3: WiLMA's Geometric Model) [pulses]
- The ratio of FrED's velocity to the fabric machine's velocity [-]
- The linear velocity command to the fabric machine mockup [mm/s]

The length of the trial is set by how much fiber is taken up by the fabric machine mockup with the size of the buffer taken into consideration. Depending on the settings, specifically the velocity ratio and how fast FrED is outputting relative to the fabric machine's intake, the length of fiber taken up by the fabric machine mockup ranges from approximately 0.2 m to 1.0 m, corresponding to a trial runtime of 20 seconds to 100 seconds.

9.3.2 WiLMA Operating Procedure for Data Collection

Prior to beginning data collection, steps must be taken to set up the hardware, ROS, and necessary serial connections. These steps are detailed in Appendix F along with a step-by-step operating procedure for the data collection process. The following paragraphs describe the operating procedure for the data collection process anecdotally.

Once the hardware is set up to begin data collection, the user may input the test parameters/settings into the brain node Python code. Prior to recording data, the gains (K_p, K_i, K_s) should be tuned to allow the trial to run to completion without the tensioner losing signal.

Additionally, since the DC motors were characterized under no load, the commanded velocity for the mockups may not correspond directly to the actual output velocities. Therefore, the input command velocity for the mockups may need to be adjusted to achieve the desired velocity for the fabric machine and the desired ratio of velocities. These values are printed as the code runs to aid in this adjustment.

Once the gains are optimized and the test parameters are validated, the user must preload the fiber so that the tensioner begins in contact with the fiber and run the test file. Once the test trial runs to completion, the data log is saved as a CSV. To reset for the next trial, the user must reset the Teensy controlling the buffer to trigger the homing sequence again while also inputting the command to spool up the fiber that was in the buffer using FrED's mockup. When FrED's mockup spool runs out of fiber, the user must "rewind" the fiber through the accumulator to return the fiber now on the fabric machine mockup's spool to FrED's spool.

10. Results and Interpretations

10.1 Statistics of Interest

This section evaluates the results of the experiment; the key variable is the spring force, the force applied to the fiber by the tensioner, since this force is a proxy for the fiber's tension. The spring force is calculated from the time-of-flight sensor's distance reading. The sensor outputs integer readings of the distance, with units of millimeters. Therefore, the resulting resolution on the spring force is estimated to be a hundredth of a pound (0.01 lbs) as that is the difference between adjacent distance readings. The following plot shows the relationship between the distance reading as read by the sensor and the calculated spring force applied to the fiber for a torsional spring with stiffness 0.01 lbs/deg. The reliable range of the sensor is 5 mm to 100 mm. The upper limit of the tensioner is a distance reading of 45 mm since the dancer length is 45.25 mm. Therefore, past 45.25 mm, the tensioner would no longer be in contact with the fiber; hence the limited range of 5 mm to 45 mm for the distance reading.

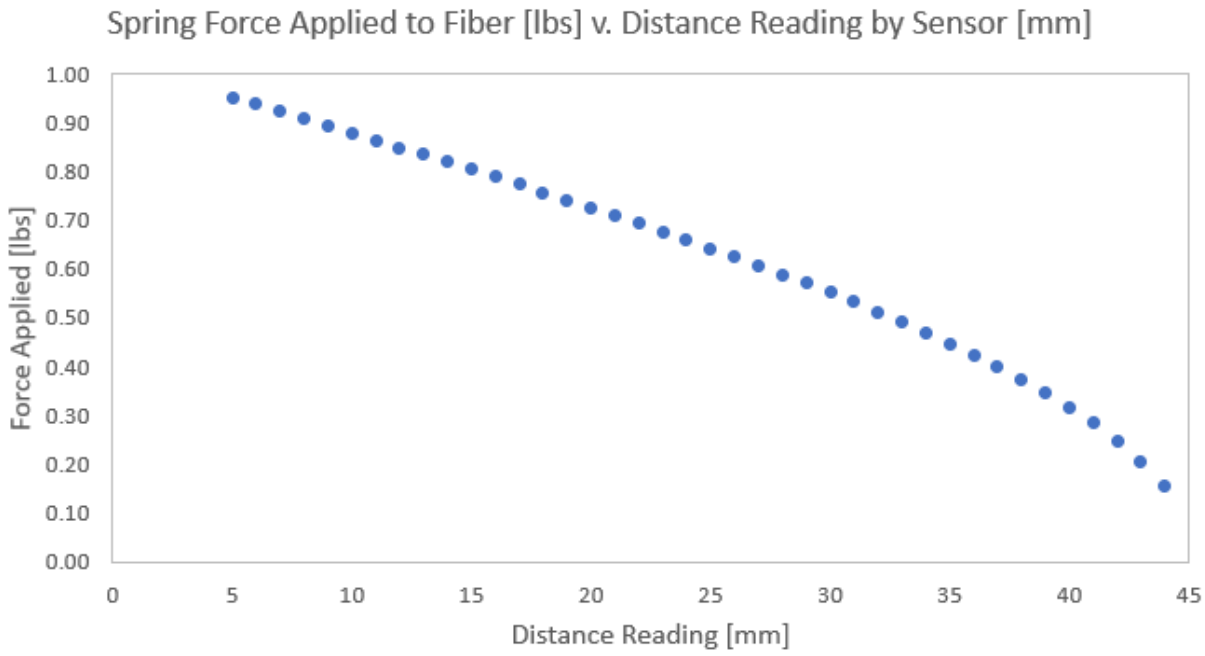


Figure 42: Plot of the relationship between the distance reading [mm] by the time-of-flight sensor in the tensioner and the calculated spring force applied to the fiber [lbs]

The average spring force is taken for each trial as well as a grand average for each set of operational profiles (fabric machine and FrED), referred to as settings. For each setting, the uncertainty on the grand average of the spring force is determined by the following equation:

$$unc_x = t_f \cdot \frac{\sigma_x}{\sqrt{N}} \qquad \text{(Eq. 13)}$$

Where t_f is the t-factor to define 95% confidence intervals, σ_x is the standard deviation of the data set, and N is the number of samples in the data set.

This method of calculating uncertainty is valid for finding the uncertainty in an averaged value. Note that finding the uncertainty of the grand average using the grand standard deviation assumes that the uncertainty in the individual averages of each trial are negligible compared to this grand averaging uncertainty. This assumption is valid for this experiment since each trial contained 200 to 1000 data points.

The performance of WiLMA with certain settings is evaluated based on the average measured spring force, as a proxy for tension, the standard deviation of the spring force over the course of the trial, and the average error.

10.2 Analysis of Constant Profile for Fabric Machine Operation

10.2.1 Sample Data for Constant Profile

This section includes representative plots of the mockups' operational profiles, WiLMA's behavior, and spring force error from a single trial for the settings corresponding with FrED's velocity at 1.5 times that of the fabric machine. This ratio was chosen for comparing the sample plots since the data for this ratio is available for both the constant low speed and high speed tests as well as the square profile tests.

The details for the following plots are:

- Figure 43 – 45: Sample Constant Low Speed Trial
    - Fabric machine running a constant profile with nominal linear velocity of 10 mm/s
    - FrED running a constant profile with nominal linear velocity of 15 mm/s, 1.5 times that of the fabric machine
    - Trial length: 1000 timesteps or 100 seconds
- Figure 46 – 48: Sample Constant High Speed Trial
    - Fabric machine running a constant profile with nominal linear velocity of 40 mm/s
    - FrED running a constant profile with nominal linear velocity of 60 mm/s, 1.5 times that of the fabric machine
    - Trial length: 250 timesteps or 25 seconds

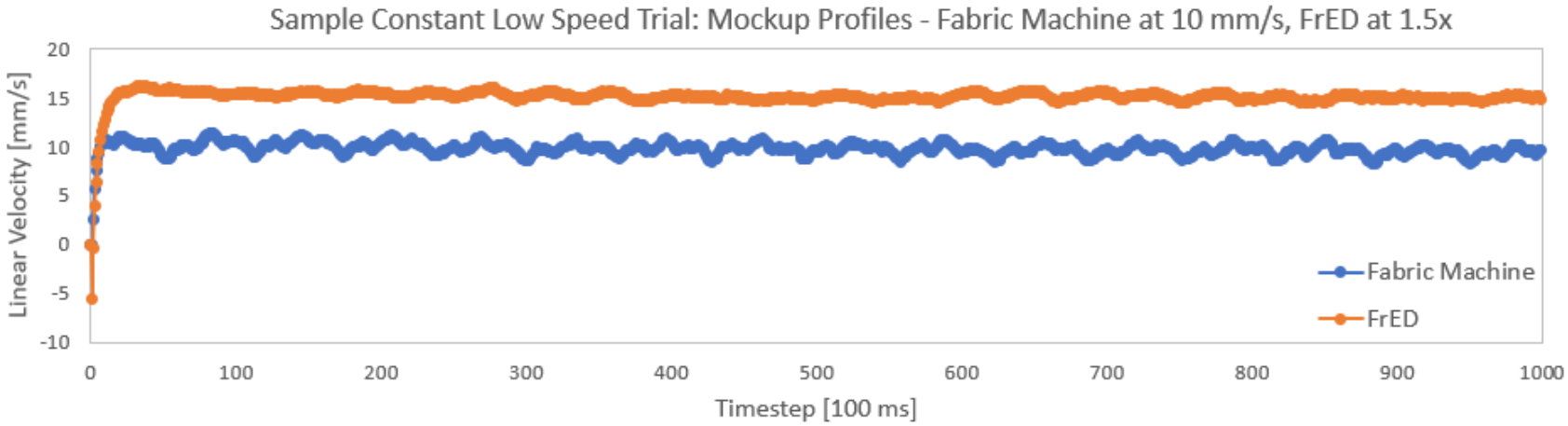


Figure 43: Sample plot of mockup profiles for a constant low speed trial for nylon monofilament

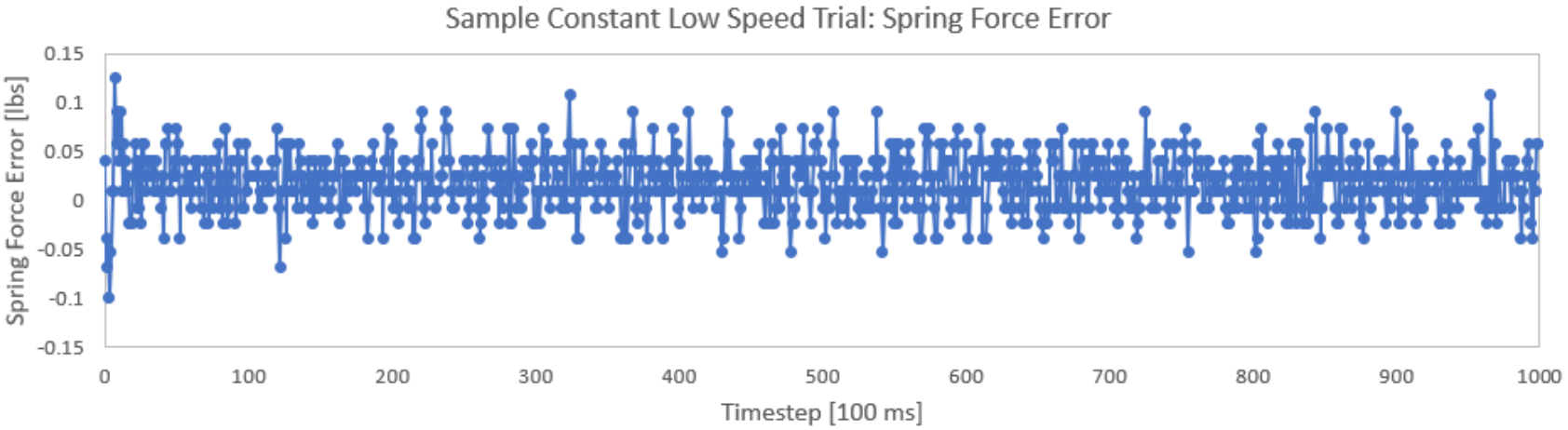


Figure 44: Sample plot of spring force error for a constant low speed trial for nylon monofilament

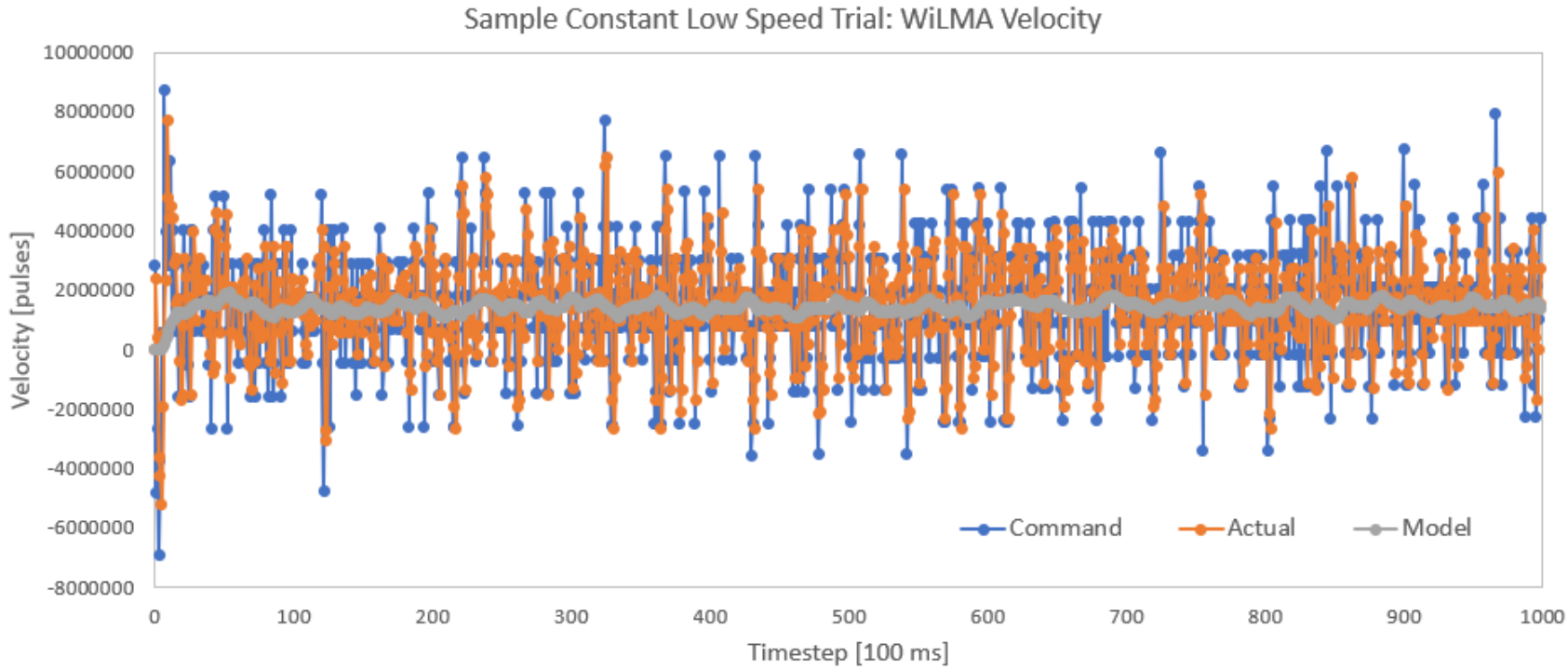


Figure 45: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for nylon monofilament

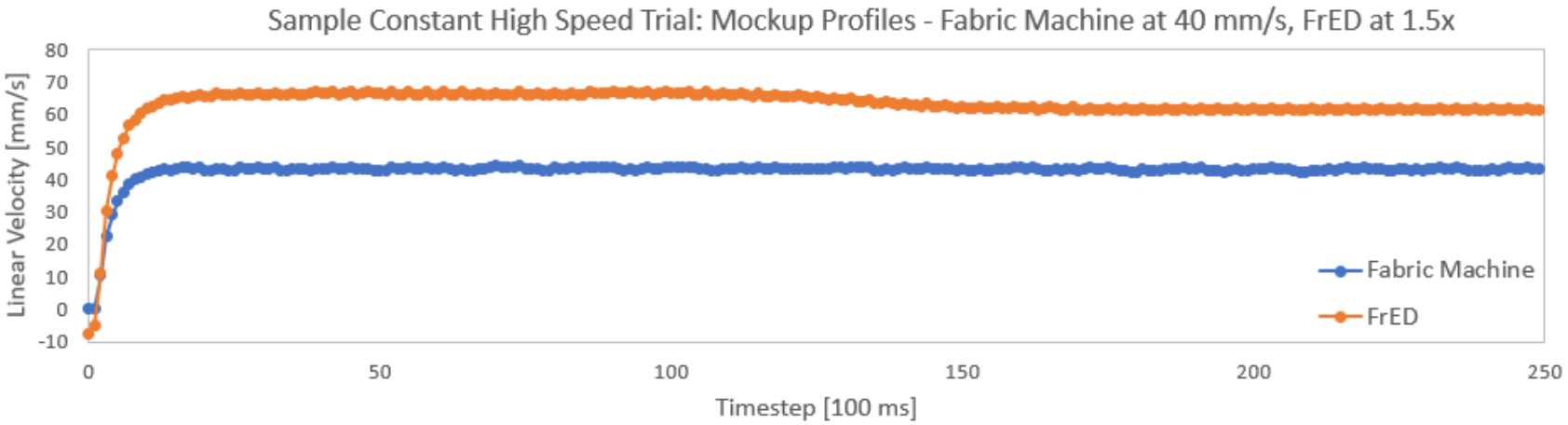


Figure 46: Sample plot of mockup profiles for a constant high speed trial for nylon monofilament

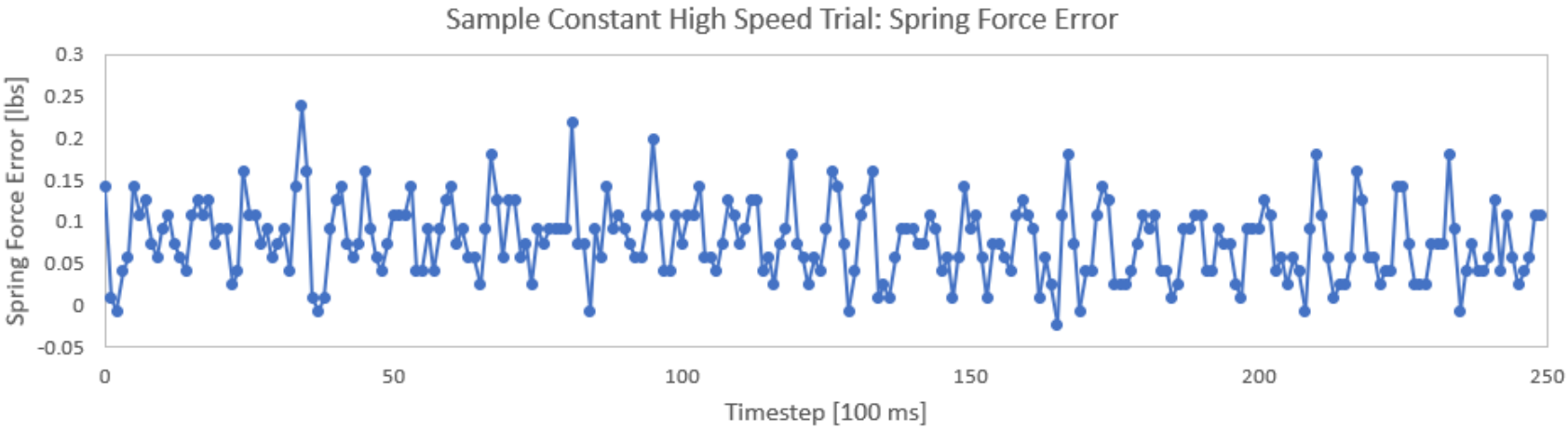


Figure 47: Sample plot of spring force error for a constant high speed trial for nylon monofilament

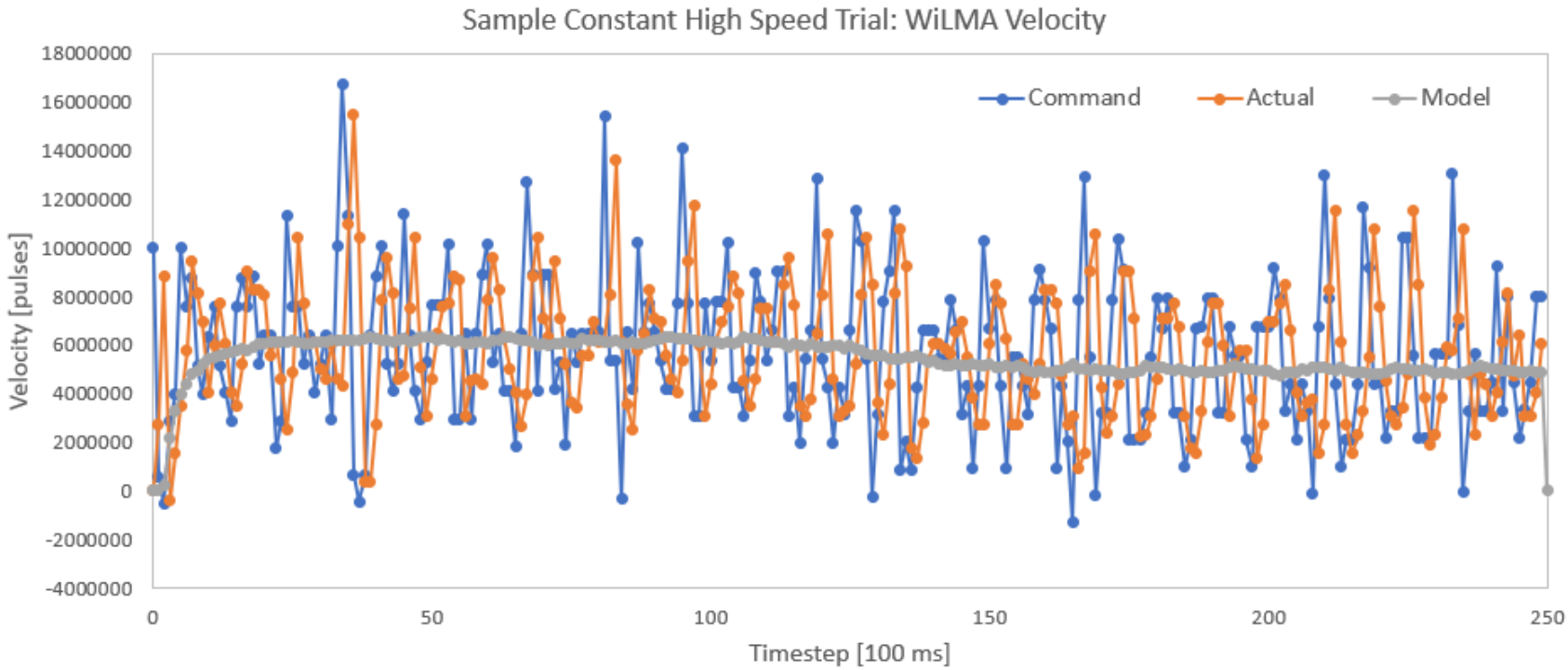


Figure 48: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for a square high speed trial for nylon monofilament

10.2.2 Constant Profile Performance Evaluation

As seen in Table 3, comparing the results from all the constant profile settings, the best performance of WiLMA, considering the average spring force and average error, is with the following settings: fabric machine at low speed (nominally 10 mm/s) and FrED at 1.25 times the speed of the fabric machine. The average spring force was 0.74 ± 0.01 lbs, resulting in an average error of 0.01 lbs relative to a set point of 0.75 lbs. Also, the summed error over the whole trial for this setting was 8.49 lbs, the only trial less than 10 lbs across all of the constant and square profile trials.

Table 3: Aggregate statistics for all constant fabric machine profile settings with nylon monofilament

Mockup Input		Grand Statistics Across 20 Trials				
Fabric Machine	FrED Ratio	Avg Ratio [-]	Avg Force [lbs]	Std Force [lbs]	Avg Error [lbs]	Summed Error [lbs]
Low Speed 10 mm/s	1.25	1.28 ± 0.08	0.74 ± 0.01	0.03	0.01	8.49
	1.50	1.47 ± 0.05	0.73 ± 0.01	0.03	0.02	16.08
	2.00	2.02 ± 0.08	0.72 ± 0.01	0.03	0.03	14.64
	3.00	3.05 ± 0.14	0.68 ± 0.01	0.04	0.07	16.57
High Speed 40 mm/s	1.25	1.27 ± 0.03	0.71 ± 0.01	0.03	0.04	11.09
	1.50	1.50 ± 0.02	0.67 ± 0.01	0.04	0.08	19.96

In general, the data suggests that the low speed trials performed better, that is a lower average error and lower standard deviation, than the high speed trials. The low speed trial with FrED at 3.0x is comparable to the high speed trial with FrED at 1.5x in terms of both average error in the spring force, 0.07 lbs – 0.08 lbs, and the standard deviation in the spring force, 0.04 lbs. All the other constant profile trials have a spring force standard deviation of 0.03 lbs and less than 0.05 lbs for the average error. The following figure compares the average spring force across all the trials, in which the set point was 0.75 lbs.

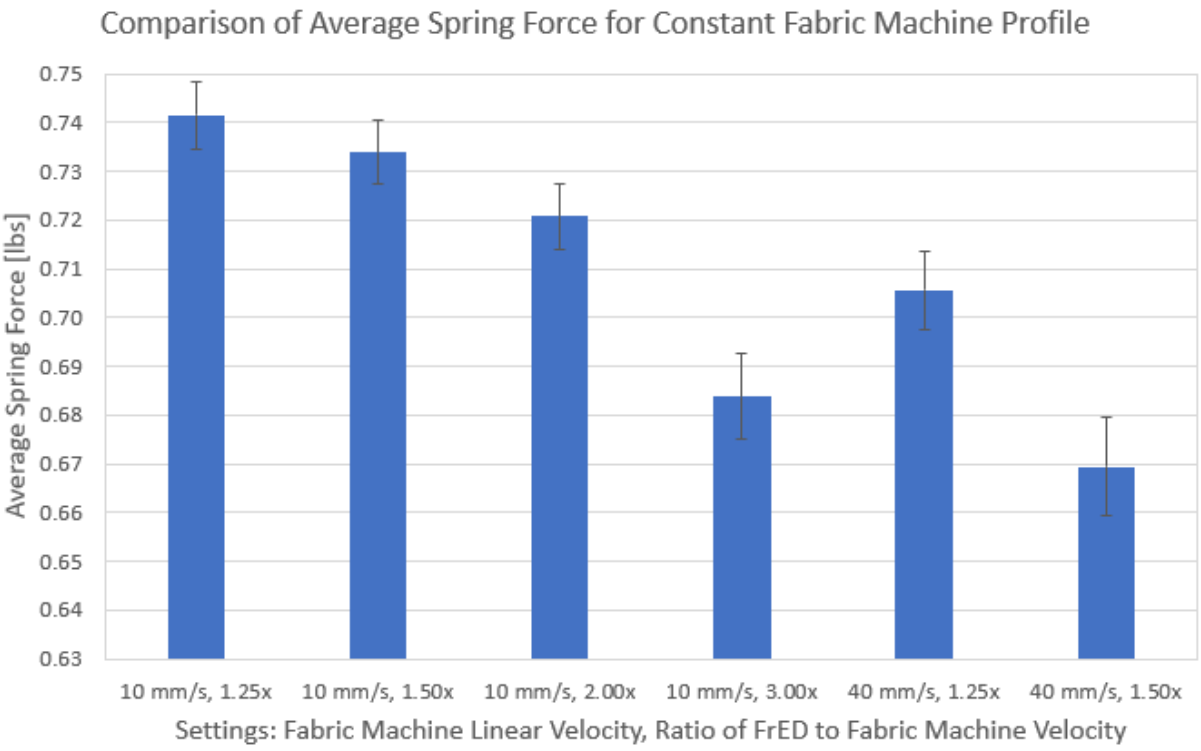


Figure 49: Comparison of the average spring force for all constant fabric machine profile trials with the set point at 0.75 lbs for nylon monofilament

10.3 Analysis of Square Profile for Fabric Machine Operation

10.3.1 Sample Data for Square Profile

Similarly, this section includes representative plots of the mockups' operational profiles, WiLMA's behavior, and spring force error from a single trial for the settings corresponding with FrED's velocity at 1.5 times that of the fabric machine. This ratio was chosen for comparing the sample plots since the data for this ratio is available for both the constant low speed and high speed tests as well as the square profile tests.

The details for the following plots are:

- Figure 50 – 55: Sample Square Low Speed Trial
    - Fabric machine running a square profile with nominal linear velocity amplitude of 10 mm/s and frequencies of 0.1 Hz (Fig. 49 – 51) and 1 Hz (Fig. 52 – 54)
    - FrED running a constant profile with nominal linear velocity of 15 mm/s, 1.5 times that of the fabric machine
    - Trial length: 1000 timesteps or 100 seconds
- Figure 56 – 61: Sample Square High Speed Trial*
    - Fabric machine running a square profile with nominal linear velocity amplitude of 30 mm/s and frequencies of 0.1 Hz (Fig. 55 – 57) and 1 Hz (Fig. 58 – 60)
    - FrED running a constant profile with nominal linear velocity of 45 mm/s, 1.5 times that of the fabric machine
    - Trial length: 250 timesteps or 25 seconds

*Note that for the high speed trials in the square profile data set, the nominal speed for the fabric machine is 30 mm/s, not 40 mm/s, as it was in the high speed trials in the constant profile data set. Through experimentation, it was found that the accumulator could not perform reliably with the fabric machine at 40 mm/s, so the velocity was incrementally decreased by 5 mm/s until a speed was found that allowed WiLMA to be stable.

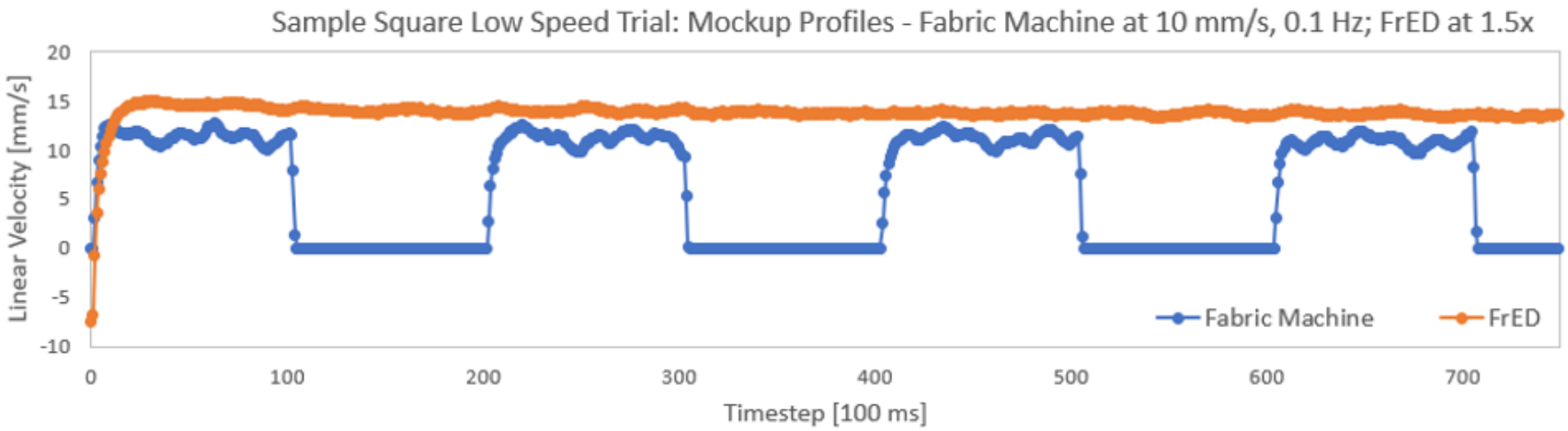


Figure 50: Sample plot of mockup profiles for a square low speed trial at 0.1 Hz for nylon monofilament with fabric machine nominally at 10 mm/s and FrED nominally at 1.5x

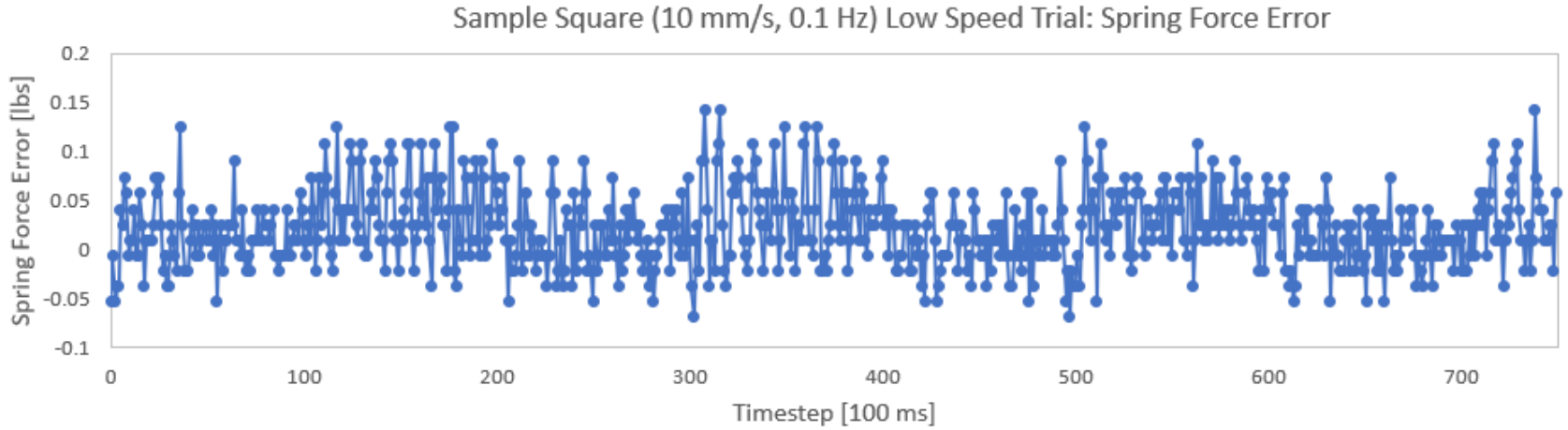


Figure 51: Sample plot of spring force error for a square low speed trial at 0.1 Hz for nylon monofilament with fabric machine nominally at 10 mm/s and FrED nominally at 1.5x

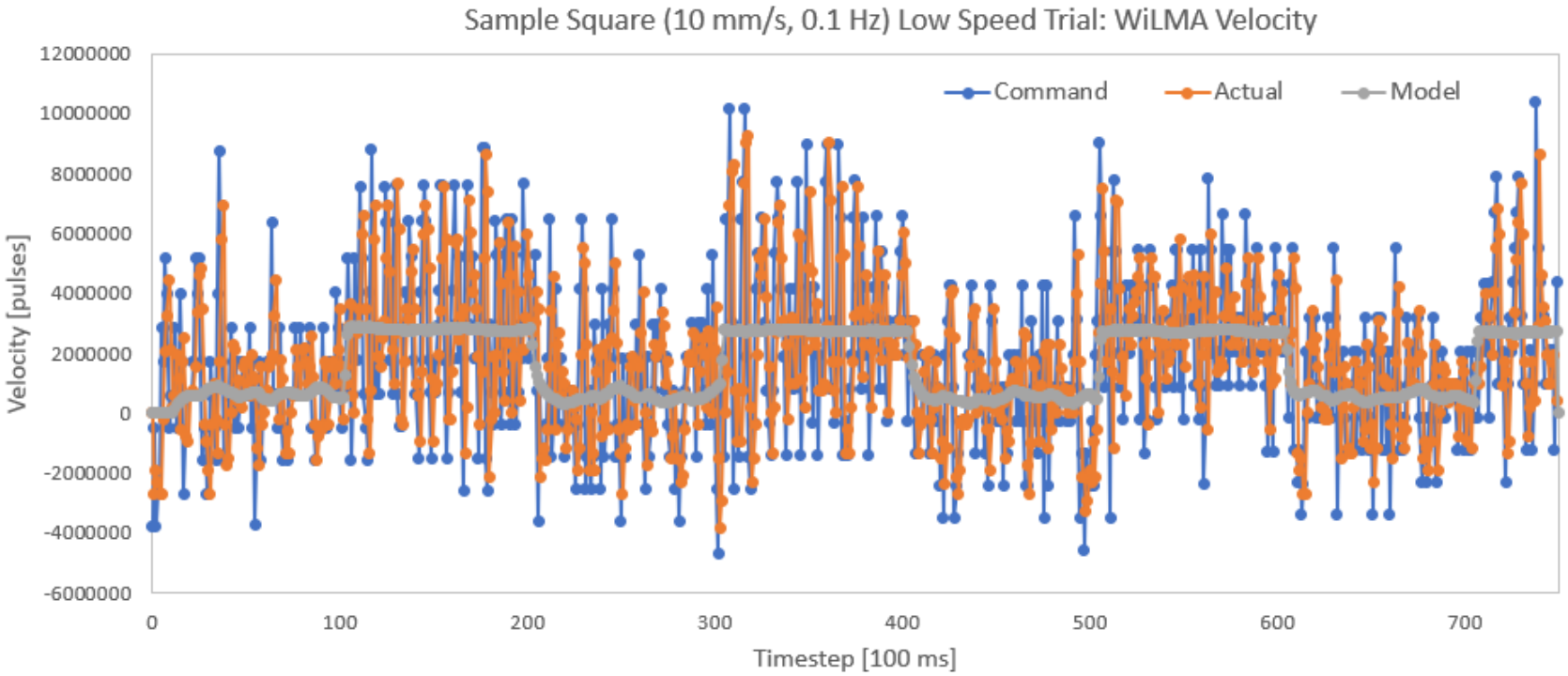


Figure 52: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for a square low speed trial at 0.1 Hz for nylon monofilament with fabric machine nominally at 10 mm/s and FrED nominally at 1.5x

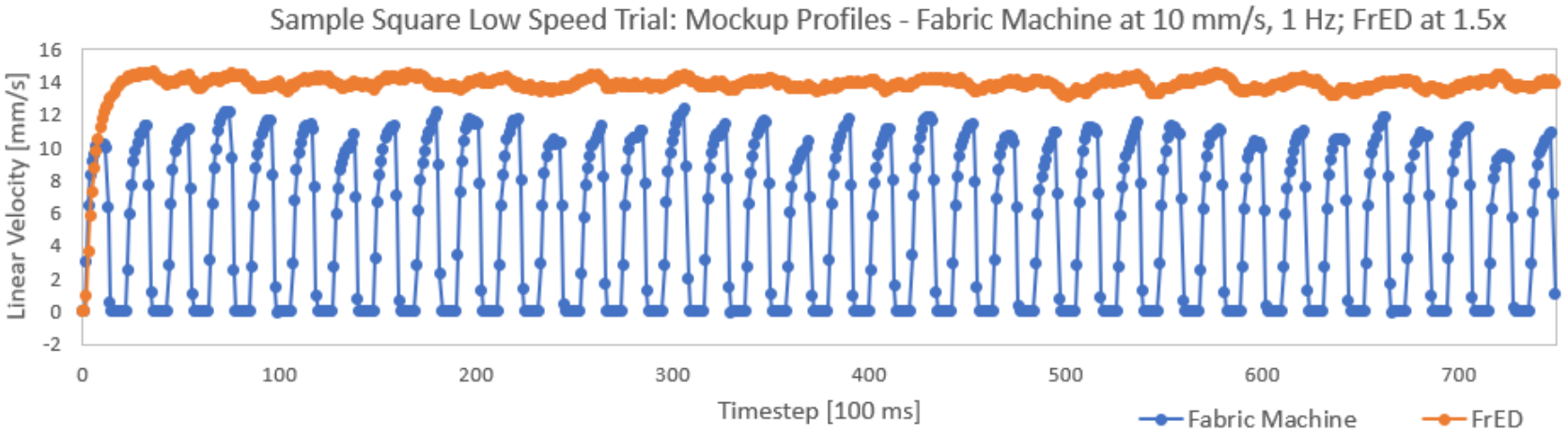


Figure 53: Sample plot of mockup profiles for a square low speed trial at 1 Hz for nylon monofilament with fabric machine nominally at 10 mm/s and FrED nominally at 1.5x

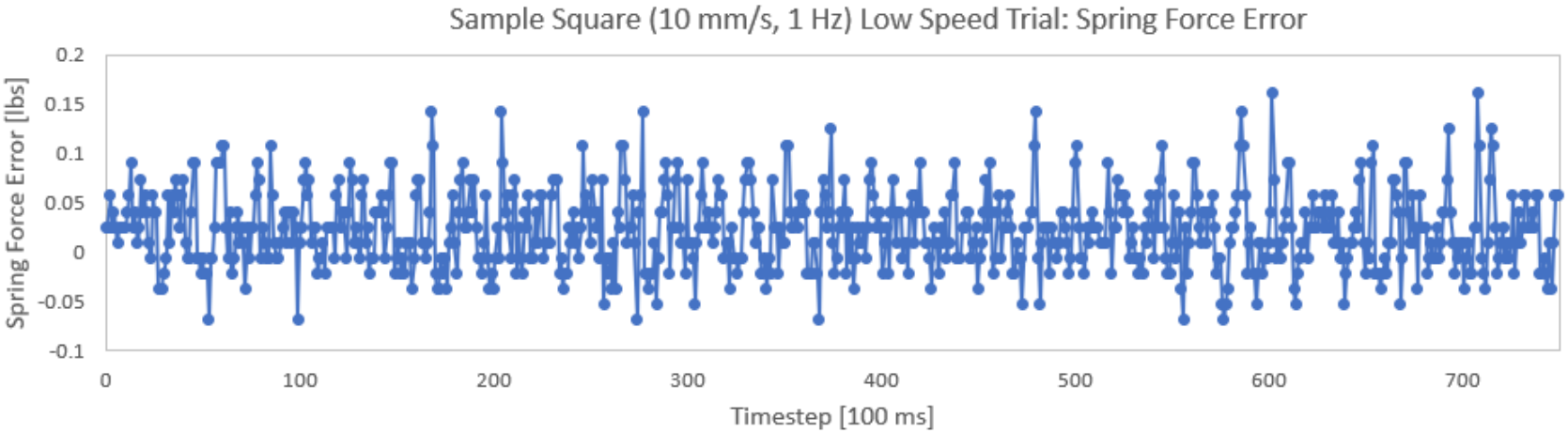


Figure 54: Sample plot of spring force error for a square low speed trial at 1 Hz for nylon monofilament with fabric machine nominally at 10 mm/s and FrED nominally at 1.5x

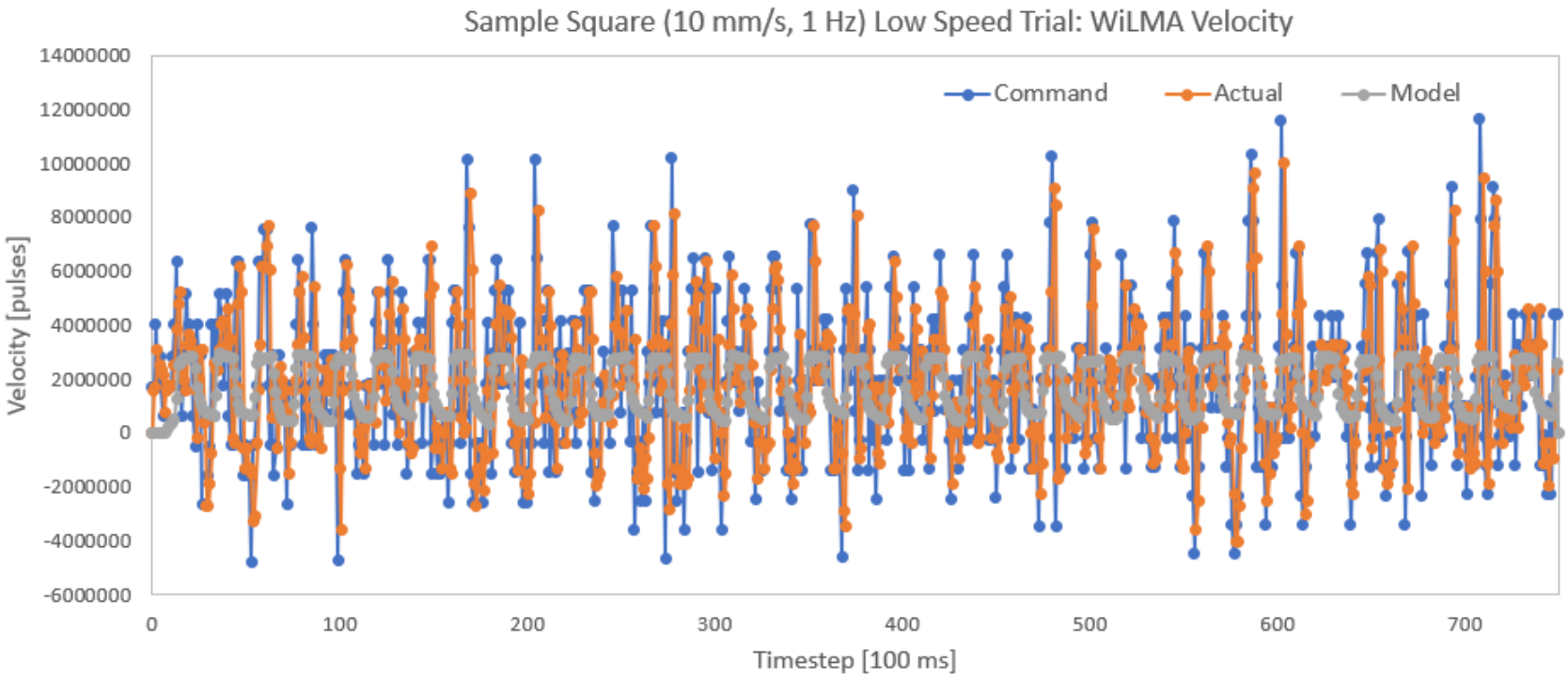


Figure 55: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for a square low speed trial at 1 Hz for nylon monofilament with fabric machine nominally at 10 mm/s and FrED nominally at 1.5x

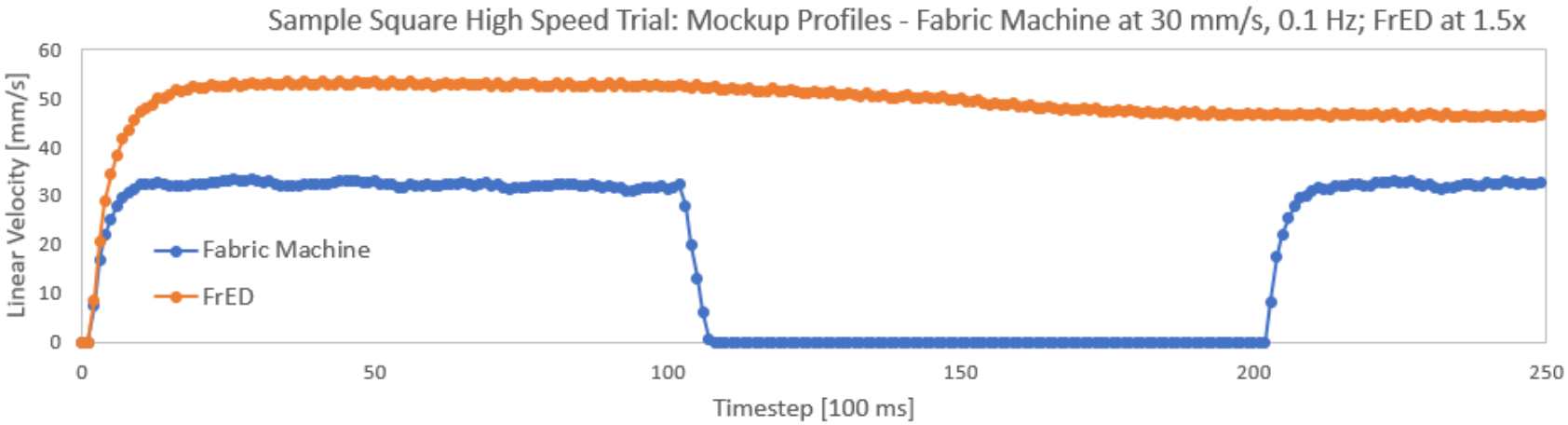


Figure 56: Sample plot of mockup profiles for a square high speed trial at 0.1 Hz for nylon monofilament with fabric machine nominally at 30 mm/s and FrED nominally at 1.5x

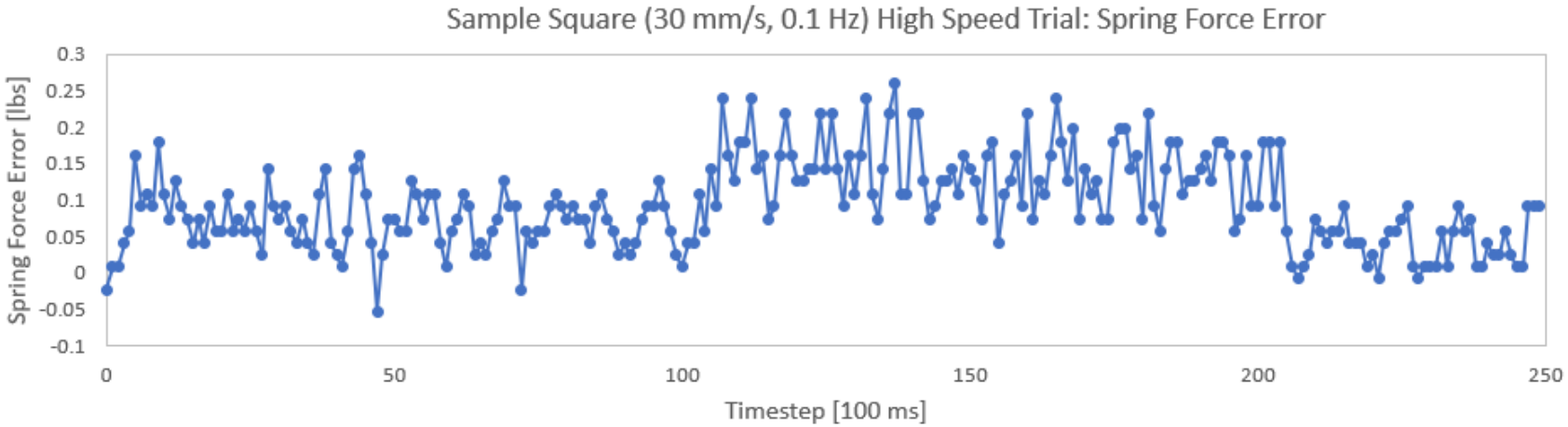


Figure 57: Sample plot of spring force error for a square high speed trial at 0.1 Hz for nylon monofilament with fabric machine nominally at 30 mm/s and FrED nominally at 1.5x

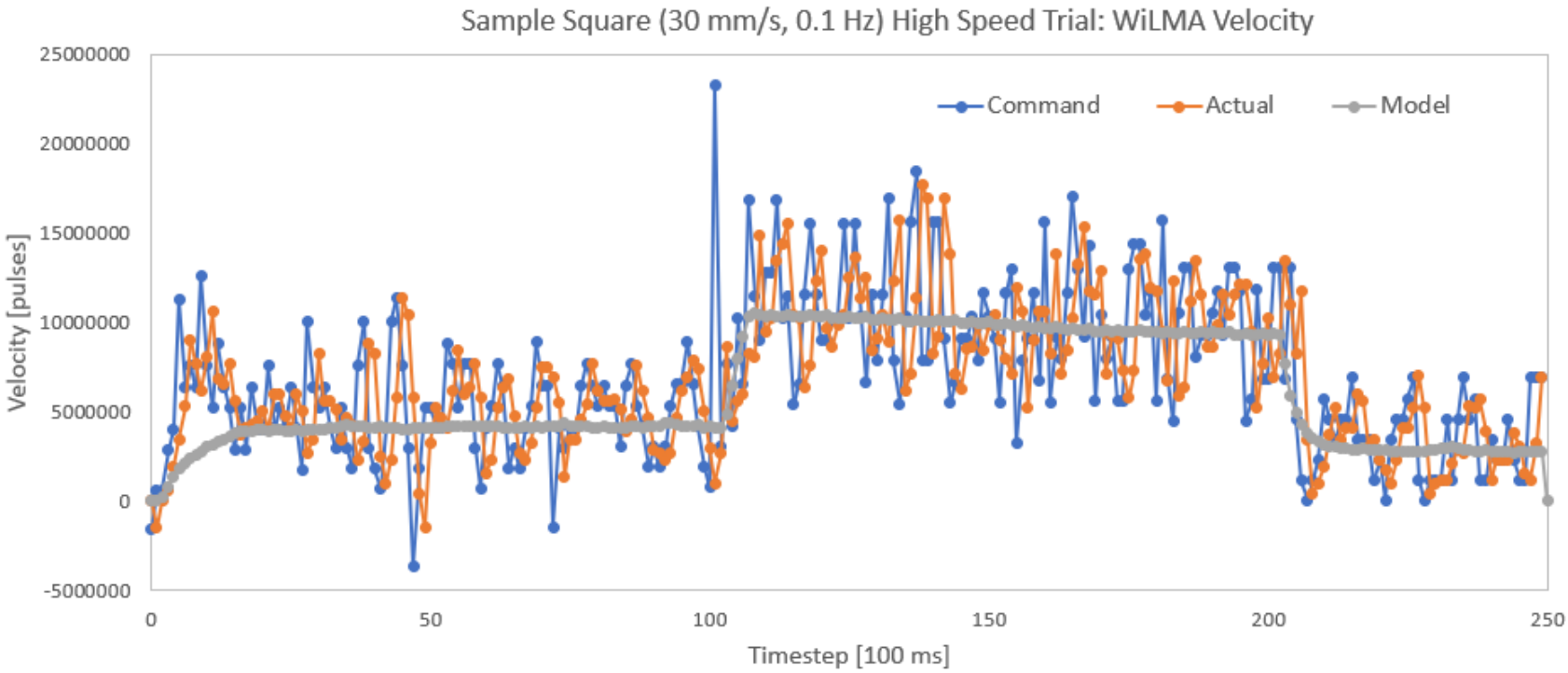


Figure 58: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for a square high speed trial at 0.1 Hz for nylon monofilament with fabric machine nominally at 30 mm/s and FrED nominally at 1.5x

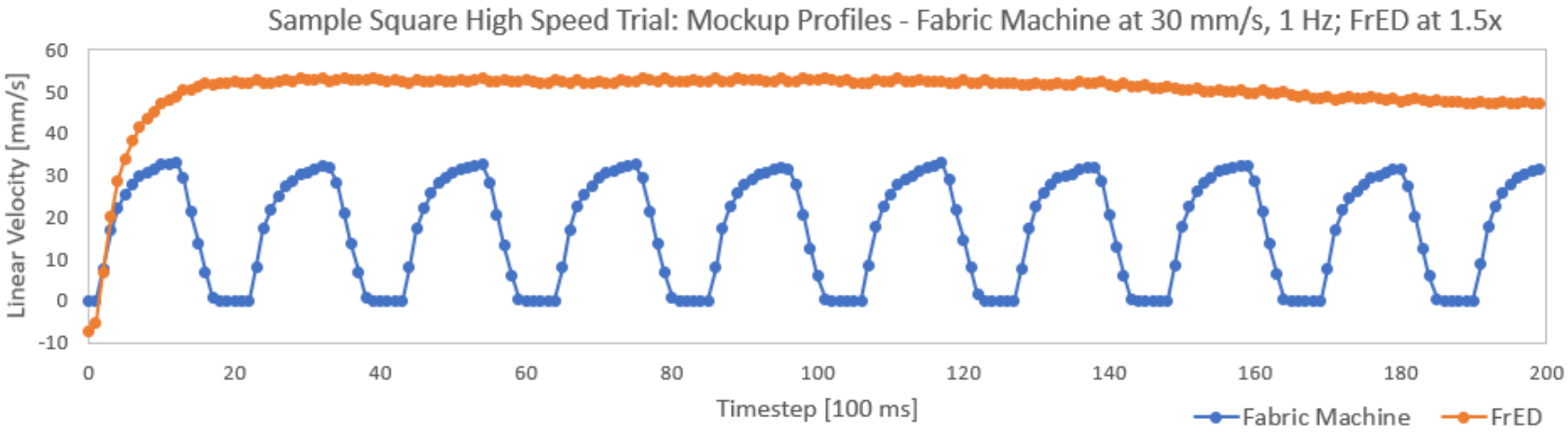


Figure 59: Sample plot of mockup profiles for a square high speed trial at 1 Hz for nylon monofilament with fabric machine nominally at 30 mm/s and FrED nominally at 1.5x

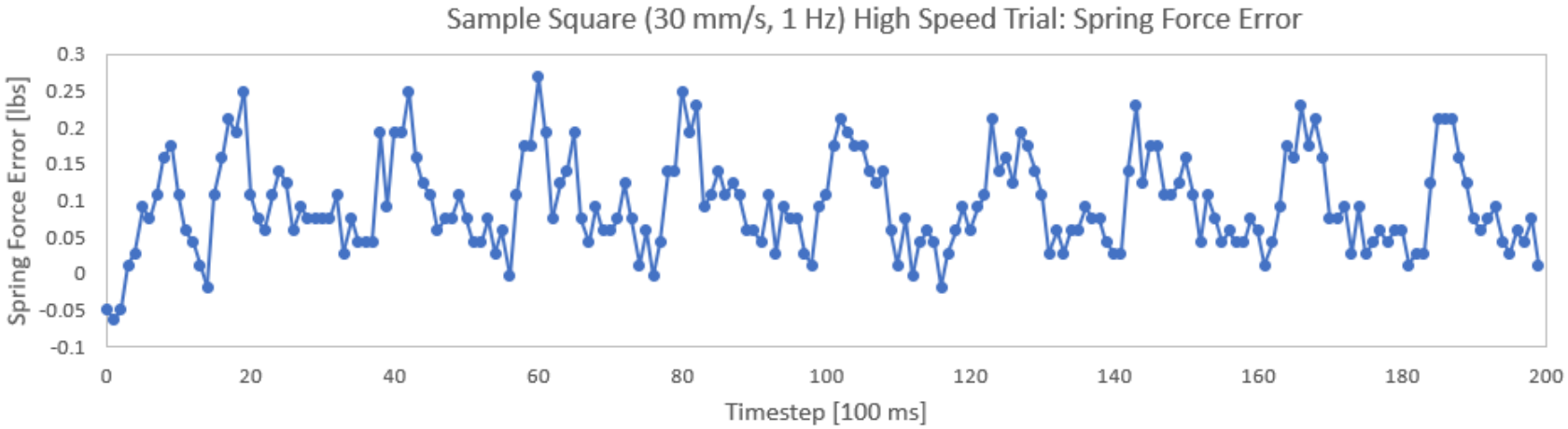


Figure 60: Sample plot of spring force error for a square high speed trial at 1 Hz for nylon monofilament with fabric machine nominally at 30 mm/s and FrED nominally at 1.5x

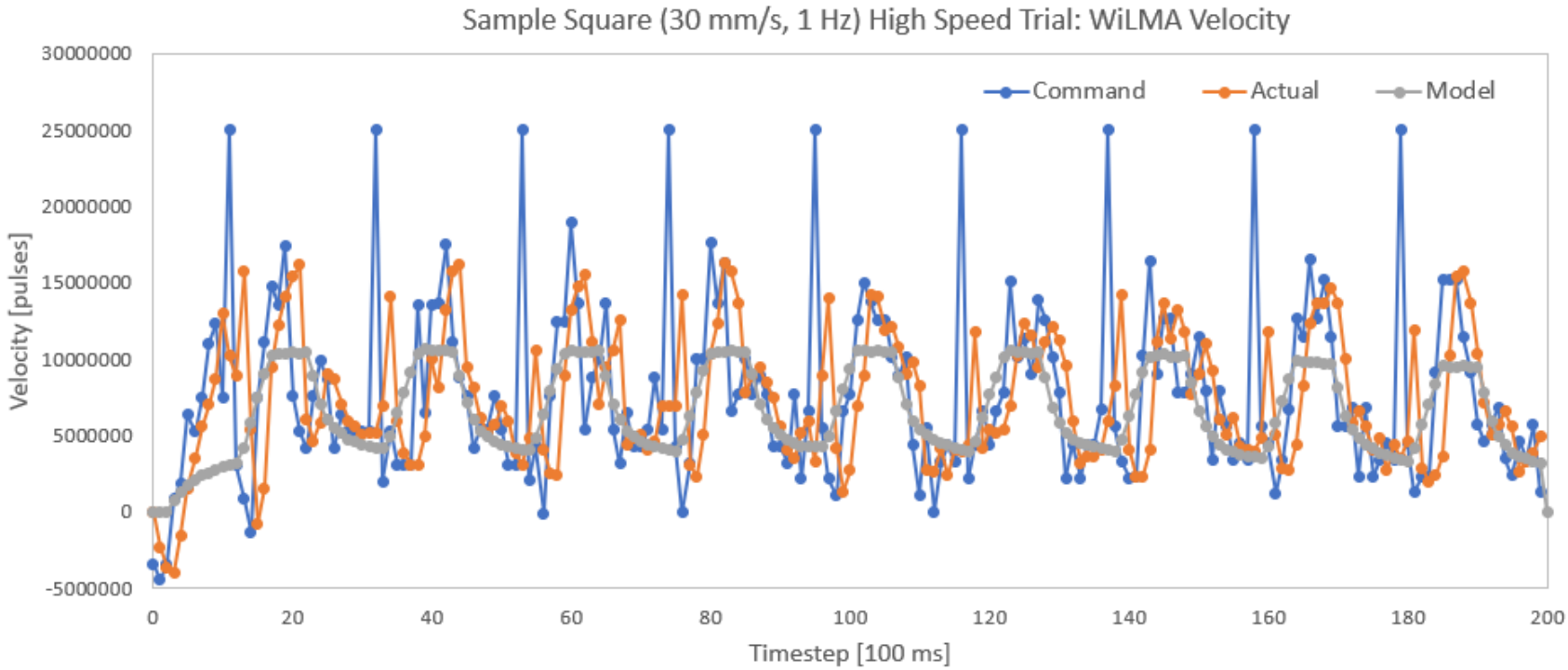


Figure 61: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for a square high speed trial at 1 Hz for nylon monofilament with fabric machine nominally at 30 mm/s and FrED nominally at 1.5x

10.3.2 Square Profile Performance Evaluation

As seen in Table 4, comparing the results from all the square profile settings, the best performance of WiLMA, considering the average spring force and average error, is with the following settings: the amplitude of the fabric machine at low speed (nominally 10 mm/s) with the frequency at 0.1 Hz, and FrED constantly at 1.5 times the nonzero speed of the fabric machine. The same settings but at the higher frequency of 1 Hz yielded very similar values for the average error and standard deviation. Both settings, regardless of frequency, yielded an average error of 0.02 lbs and an average standard deviation of 0.04 lbs.

Table 4: Aggregate statistics for all square fabric machine profile settings with nylon monofilament

Mockup Input			Grand Statistics Across 20 Trials				
Fabric Machine	Freq. [Hz]	FrED Ratio	Avg Ratio [-]	Avg Force [lbs]	Std Force [lbs]	Avg Error [lbs]	Summed Error [lbs]
Low Speed 10 mm/s	0.1	1.50	1.26 ± 0.14	0.73 ± 0.01	0.04	0.02	15.17
		3.00	2.99 ± 0.36	0.67 ± 0.01	0.05	0.08	19.75
	1.0	1.50	1.45 ± 0.30	0.73 ± 0.01	0.04	0.02	16.37
		3.00	3.47 ± 0.77	0.67 ± 0.01	0.05	0.08	20.64
High Speed 30 mm/s	0.1	1.50	1.58 ± 0.18	0.66 ± 0.02	0.07	0.09	22.20
	1.0	1.50	1.85 ± 0.37	0.71 ± 0.02	0.07	0.09	18.76

Again, the low speed trials overall performed better than the high speed trials. All the low speed trials have a lower average error than the high speed trials. Although the average error for the low speed case with FrED at 3.0 times the speed of the fabric machine is comparable to the average errors in the high speed cases, the standard deviation is lower in the low speed case. Additionally, note that the high speed trials for the square profile were performed at a nominal "high speed" of 30 mm/s, as opposed to the nominal of 40 mm/s as in the constant profile case. As mentioned in the previous section, this is because the system was not stable when the nominal amplitude of the fabric machine was set at 40 mm/s, causing the tensioner to lose signal, despite attempts to stabilize the system by tuning the gains.

An interesting result is that the average error for both frequencies in the high speed cases are comparable, yet the average spring force for the higher frequency (1 Hz) is closer to the actual set point. A possible explanation may be that as the frequency approaches the accumulator's operating

frequency (10 Hz), the fabric machine's profile begins to appear more continuous than discrete. Further testing must be done to test this hypothesis. The following figure compares the average spring force across all the trials, in which the set point was 0.75 lbs.

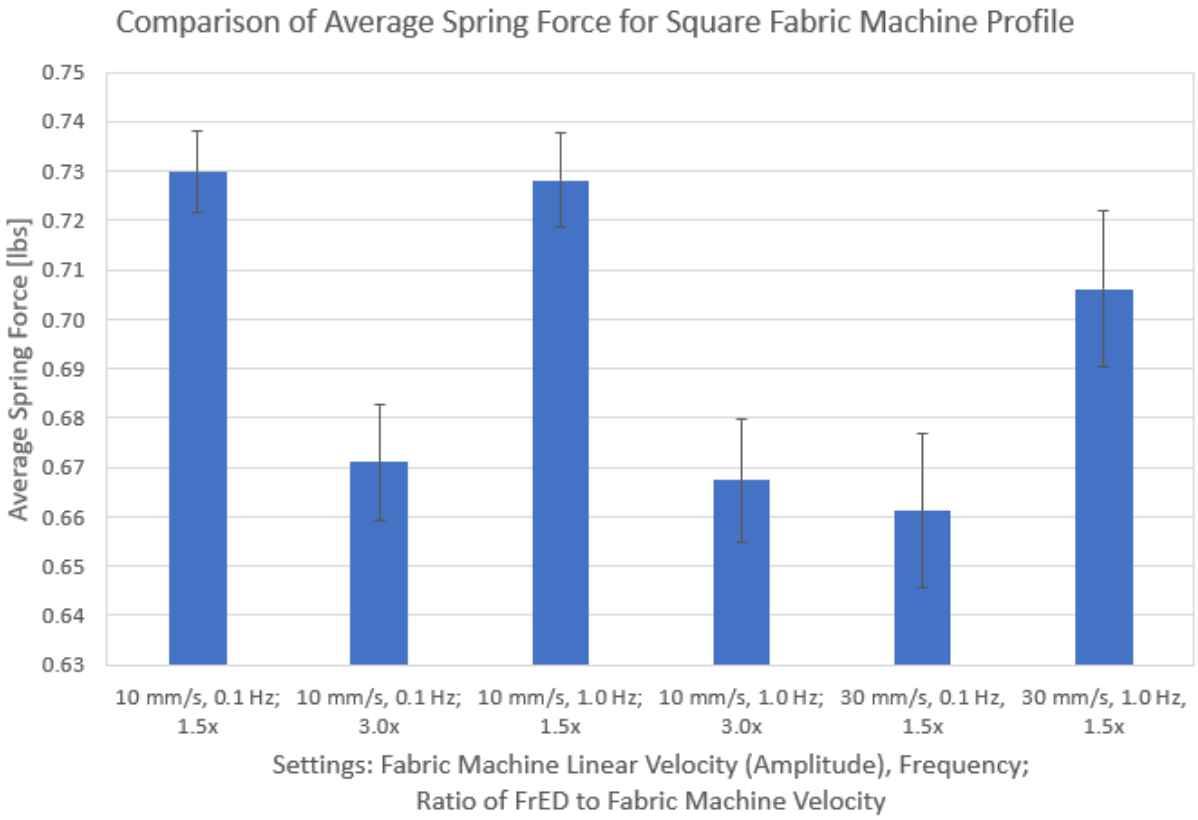


Figure 62: Comparison of the average spring force for all square fabric machine profile trials with the set point at 0.75 lbs for nylon monofilament

10.4 Overall Profile-Dependent Performance Evaluation

The below table compares all the trials with different settings together. A student's t-test was performed for every pair of the 12 trials. Of the 66 total pairs being compared, all datasets except for 3 were found to be statistically significantly different (effective 100% confidence) when comparing the average spring force. In other words, the values in the table (Table 5) can be interpreted with little uncertainty. The three exceptions were:

- With approximately 60% confidence, the average spring force in a square, high speed trial with FrED at 1.5x the fabric machine and a frequency of 1.0 Hz is greater than the average spring force in a constant, high speed trial with FrED at 1.25x.
- With approximately 90% confidence, the average spring force in a square, low speed trial with FrED at 3.0x the fabric machine and a frequency of 0.1 Hz is greater than the average spring force in a constant, high speed trial with FrED at 1.5x.
- With approximately 93% confidence, the average spring force in a square, low speed trial with FrED at 3.0x the fabric machine and a frequency of 1.0 Hz is greater than the average spring force in a constant, high speed trial with FrED at 1.5x.

Table 5: Aggregate statistics for all fabric machine profile settings with nylon monofilament

Mockup Input			Grand Statistics Across 20 Trials				
Fabric Machine	Freq. [Hz]	FrED Ratio	Avg Ratio [-]	Avg Force [lbs]	Std Force [lbs]	Avg Error [lbs]	Summed Error [lbs]
Low Speed 10 mm/s	-	1.25	1.28 ± 0.08	0.74 ± 0.01	0.03	0.01	8.49
	-	1.50	1.47 ± 0.05	0.73 ± 0.01	0.03	0.02	16.08
	-	2.00	2.02 ± 0.08	0.72 ± 0.01	0.03	0.03	14.64
	-	3.00	3.05 ± 0.14	0.68 ± 0.01	0.04	0.07	16.57
High Speed 40 mm/s	-	1.25	1.27 ± 0.03	0.71 ± 0.01	0.03	0.04	11.09
	-	1.50	1.50 ± 0.02	0.67 ± 0.01	0.04	0.08	19.96
Low Speed 10 mm/s	0.1	1.50	1.26 ± 0.14	0.73 ± 0.01	0.04	0.02	15.17
		3.00	2.99 ± 0.36	0.67 ± 0.01	0.05	0.08	19.75
	1.0	1.50	1.45 ± 0.30	0.73 ± 0.01	0.04	0.02	16.37
		3.00	3.47 ± 0.77	0.67 ± 0.01	0.05	0.08	20.64
High Speed 30 mm/s	0.1	1.50	1.58 ± 0.18	0.66 ± 0.02	0.07	0.09	22.20
	1.0	1.50	1.85 ± 0.37	0.71 ± 0.02	0.07	0.09	18.76

An important result from these experiments is that the average spring forces never exceeded the set point force. The average spring force range was 0.67 lbs to 0.74 lbs with a set point of 0.75 lbs for the majority of the trials. This result is significant and promising because none of the settings resulted in the fiber being over-tensioned or mechanically stressed for an extended period of time. Although there is a steady-state error present in all the trials, this error, below the set point, did not fall below the lower limit of the spring force that would have caused the tensioner to completely lose contact with the fiber and lose signal.

In order of performance as evaluated based on average spring force and average standard deviation (best to worst):

- Constant, low speed, 1.25x
- Constant, low speed, 1.5x
- Square, low speed, 0.1 Hz, 1.5x
- Square, low speed, 1 Hz, 1.5x
- Constant, low speed, 2x
- Constant, high speed, 1.25x
- Constant, low speed, 3x
- Square, low speed, 0.1Hz, 1.5x
- Constant, high speed, 1.5x
- Square, low speed, 1 Hz, 1.5x
- Square, high speed, 1 Hz, 1.5x
- Square, high speed, 0.1 Hz, 1.5x

To better visualize this comparison across all the trials, the performance of each setting is plotted in the following figure, where the x- and y-axis are average error in the spring force and average standard deviation, respectively.

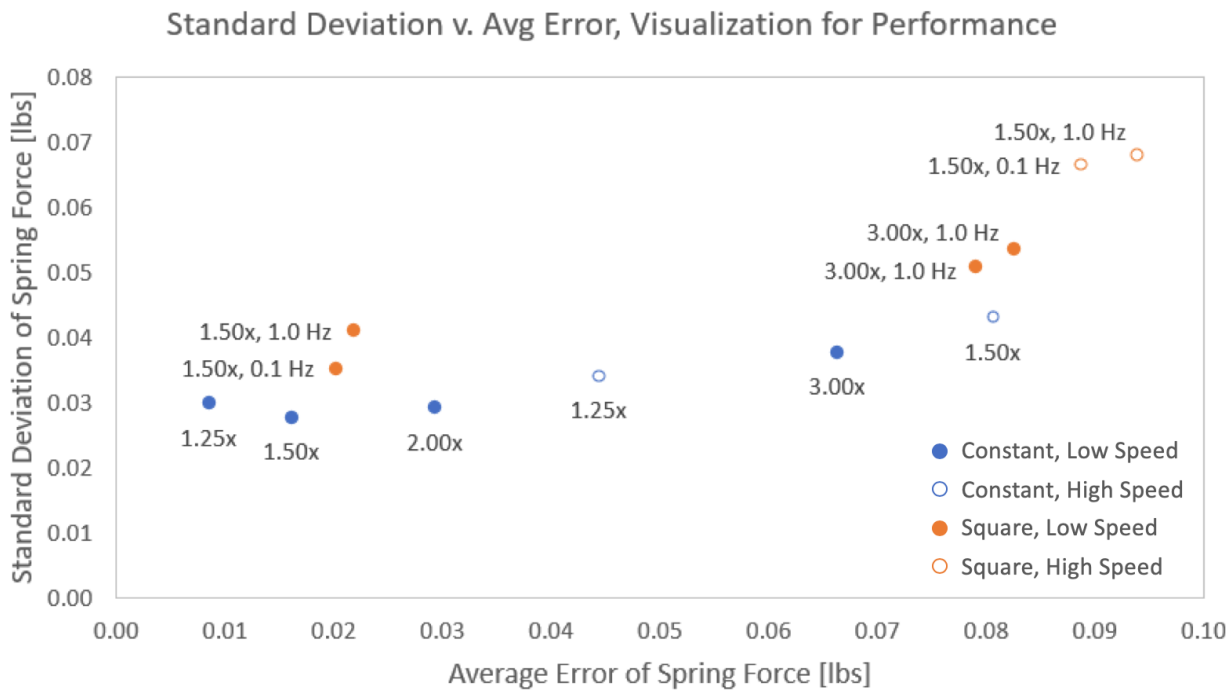


Figure 63: Visualization to compare performances across all settings for nylon monofilament

In Figure 63, the ideal performance is the bottom left corner because that corresponds to both low standard deviation and low error. As seen in the plot, this cluster in the bottom left corner is composed of all low speed trials, as expected. The cluster also seems to indicate better performance when FrED is operating at less than 2x the fabric machine for the constant profile and less than 1.5x, regardless of frequency, for the square profile. Of the two performance criteria, the more important statistic is the standard deviation of the spring force since a steady-state error can be corrected in the controller with a systematic shift. In considering only this axis on the plot, the constant profile trials generally have lower standard deviation than the square profile trials.

10.5 Preliminary Performance with PMMA Fiber

In addition to conducting the experiment with the nylon monofilament, preliminary tests were done with PMMA fiber. The use of PMMA fiber is relevant since the fiber used in these tests was actually produced by FrED. Using another fiber material tested the versatility of WiLMA since the PMMA, compared to the nylon monofilament, is less flexible and has a larger diameter (0.55 mm to 0.60 mm) compared to the nylon (0.35 mm nominal).

The results from testing with the PMMA fiber are only for proof-of-concept since at the time of testing the PMMA, the ratio of FrED's velocity to the fabric machine's velocity was not yet recorded. The ratio was recorded beginning in preliminary nylon monofilament trials; it was found that the nominal speed commands resulted in ratios that were up to 1.5 times what was expected. This error is likely due to the fact that the DC motors were characterized under no load and separately from the accumulator. Hence, the actual velocity varied slightly but resulted in a significantly larger ratio than the nominal because the DC motor for the fabric machine tended to run significantly slower than originally characterized under no load.

However, the preliminary results for the PMMA fiber are promising given that the ratio of FrED's velocity to the fabric machine velocity was likely higher than nominal. Also, the nominal high speed was 50 mm/s for the preliminary PMMA trials, as opposed to 40 mm/s for nylon. The tests only involve the fabric machine running a constant profile. The nominal ratios tested with the PMMA fiber are: 1.1x (low and high speed), 1.25x, (low and high speed), 1.5x (low speed only), and 2x (low speed only); all settings were tested with 20 trials each. Based on how quickly the buffer filled up for these trials, it is estimated that the actual ratio was at least 1.5x for the majority of the trials. The results from the preliminary testing are summarized in Table 6.

Table 6: Aggregate preliminary statistics for all constant fabric machine profile settings with PMMA fiber

Mockup Input		Grand Statistics Across 20 Trials			
Fabric Machine	Nominal FrED Ratio	Avg Force [lbs]	Std Force [lbs]	Avg Error [lbs]	Summed Error [lbs]
Low Speed 10 mm/s	1.10	0.66 ± 0.02	0.05	0.04	18.19
	1.25	0.66 ± 0.02	0.05	0.04	19.16
	1.50	0.66 ± 0.02	0.05	0.04	18.46
	2.00	0.65 ± 0.02	0.05	0.05	21.35
High Speed 40 mm/s	1.10	0.65 ± 0.02	0.05	0.05	10.15
	1.25	0.63 ± 0.02	0.05	0.07	14.83

For all these trials, except the high speed "1.25x" trial, WiLMA operated with an average error in the spring force of 0.04 lbs to 0.05 lbs while the standard deviation was consistently 0.05 lbs. Sample plots for the PMMA fiber trials at both low speed and high speed, with FrED at "1.25x," may be seen in the following figures (Fig. 64 – 67). Another significant result from the preliminary testing of the PMMA fiber is that the gain parameters used were the same as those in the nylon test trials. In other words, although a different material fiber with different mechanical properties was used, the gain parameters that allowed WiLMA to run stably did not need to be re-optimized.

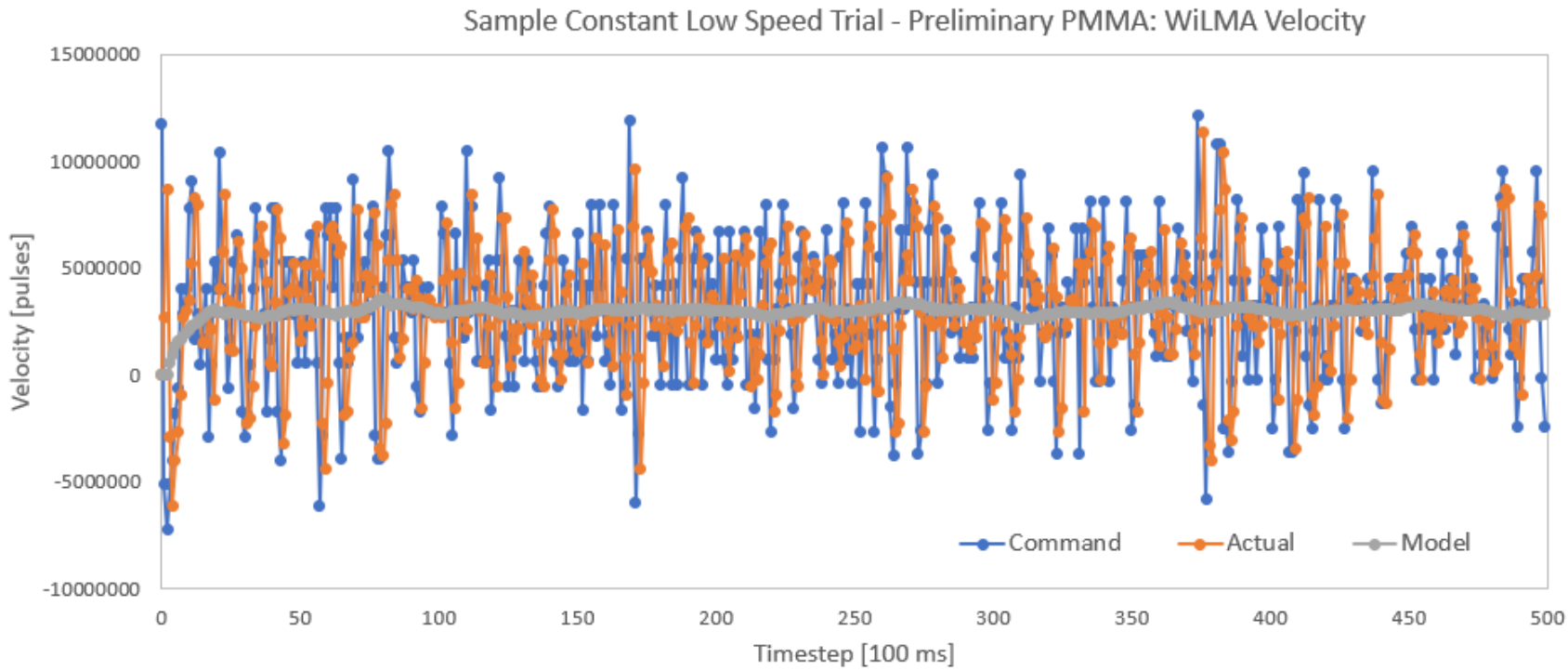


Figure 64: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for a constant low speed trial for PMMA with fabric machine nominally at 10 mm/s and FrED at "1.25x"

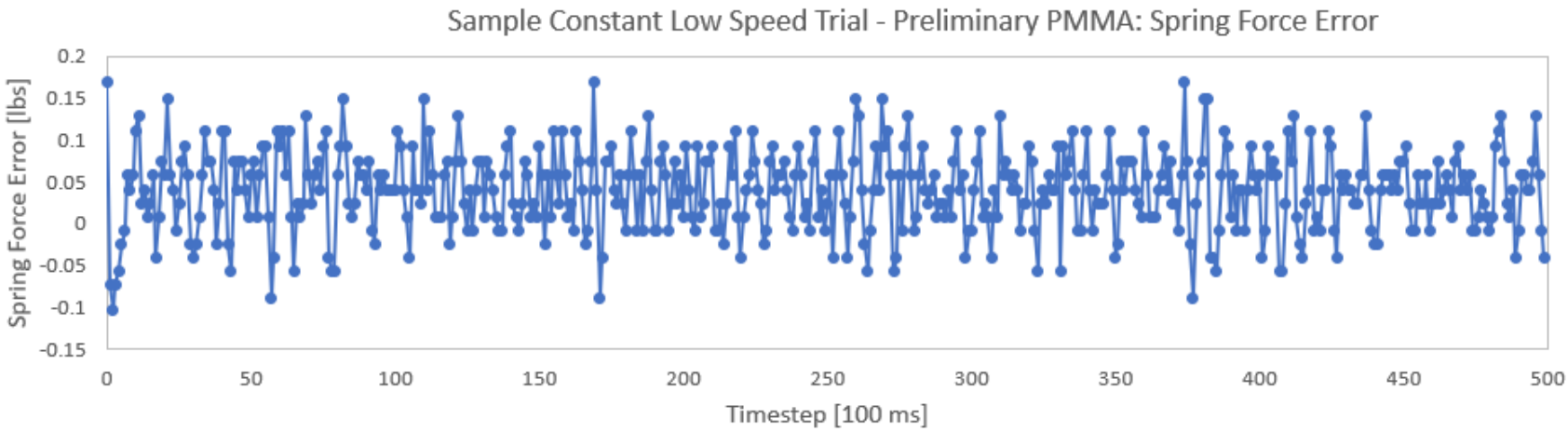


Figure 65: Sample plot of spring force error for a constant low speed trial for PMMA with fabric machine nominally at 10 mm/s and FrED at "1.25x"

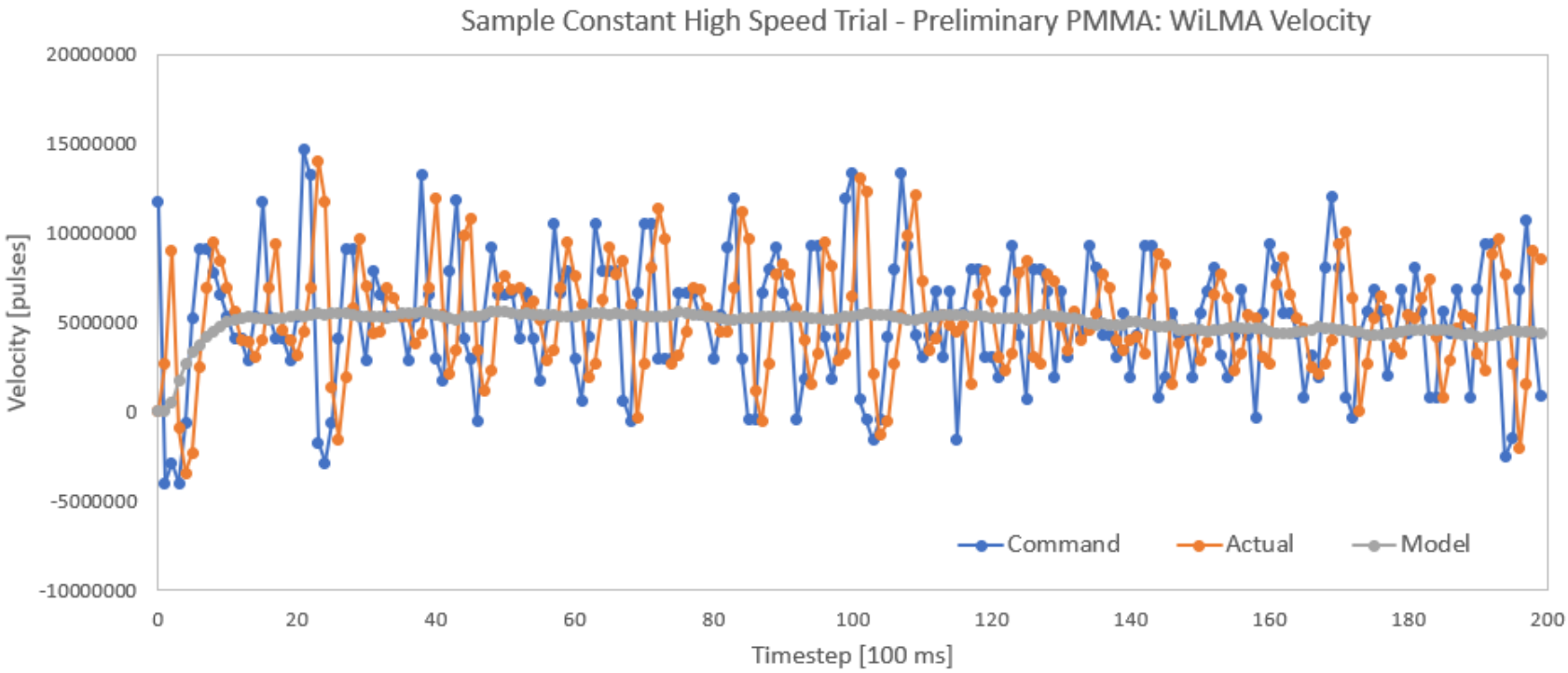


Figure 66: Sample plot of WiLMA's Velocity, comparing the commanded value, the actual value, and the model value for a constant high speed trial for PMMA with fabric machine nominally at 50 mm/s and FrED at "1.25x"

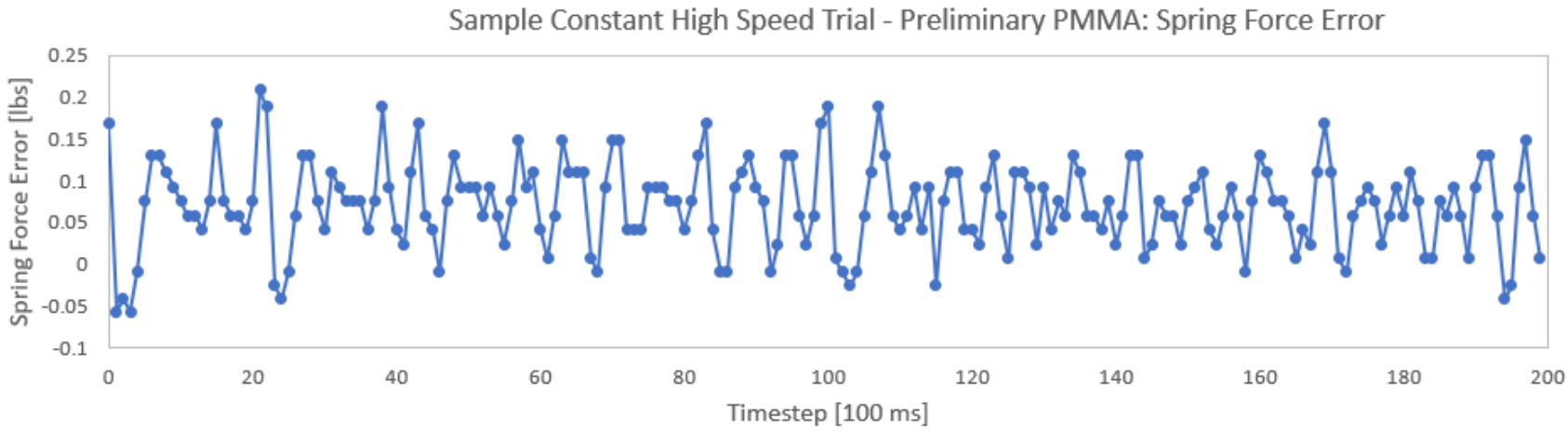


Figure 67: Sample plot of spring force error for a constant high speed trial for PMMA with fabric machine nominally at 10 mm/s and FrED at "1.25x"

10.6 WiLMA's Modified PI Controller v. Geometric Model

As seen in the previous figures that showed WiLMA's velocity, qualitatively, the model value appears to be the mean value of the controller values. Therefore, the model, based solely on geometry, provides a good estimate of the system's expected behavior. In other words, friction, motor behavior, and other physical do not contribute significantly to the system's response.

For the constant profile trials, the range of the controller value relative to the model value seems to be ±3e6 pulses for the low speed condition and ±5e6 pulses for the high speed condition. For the square profile trials, the range seems to be ±5e6 pulses for the low speed condition and ±10e6 pulses for the high speed condition. The width of the range seems to correspond to the settings' variation in tension, or the standard deviation in the spring force.

11. Conclusions

In general, WiLMA is able to act as an effective accumulator to buffer the fiber between production from FrED and intake/use by the fabric machine. WiLMA was tested with various settings: constant profile, square profile, low speed, and high speed. Regardless of the setting, all trials had an average spring force below the set point. The spring force being less than the set point is ideal since the set point is presumably the threshold, with some safety factor, of the fiber's tension limit before being mechanically deformed. In all of the trials, the tensioner never lost signal, allowing all the trials to run to completion. In other words, even though the spring force was below the set point, the force never reached the lower limit that corresponds to the fiber losing tension and potentially tangling or otherwise disrupting the production line.

In assessing the performance of the accumulator, the standard deviation of the spring force and the resulting error is compared across the different settings. Provided the standard deviation is low, the absolute value of the steady-state error is insignificant since the systematic shift can be corrected/offset in the controller. Therefore, minimizing the variation in the tension, i.e. the standard deviation of the spring force, is more important than the absolute value of the spring force itself. The standard deviation was less than 0.07 lbs for all the different settings and less than 0.05 lbs for eight of the twelve settings.

The accumulator performed satisfactorily with the machine upstream, FrED mockup, outputting fiber at a constant rate while the machine downstream, the fabric machine mockup, took up the fiber with constant and square profiles. WiLMA's dynamic response and control when

tested with the square profile distinguishes WiLMA from industry versions of the linear tension control accumulator that use a fixed physical counterweight to maintain tension. The data-rich controls system also allows for systematic troubleshooting and debugging for functional and education purposes.

Overall, WiLMA successfully addresses the functional requirements set forth in Section 4 by effectively buffering the speed difference between the machine upstream and the machine downstream, not permanently deforming the fiber through over-tensioning it, and is easy to manufacture, assemble, and scale.

12. Recommendations for Further Work

12.1 Design Improvements

To increase the buffer capacity and hence runtime, the idler length and the corresponding number of grooves can be easily increased with little consequence. The only consideration is not making the idler length so long that it becomes cantilevered (Section 6.2.3: Linear Carriage Force). Increasing the buffer capacity also allows a larger range of speed ratios to be tested.

As mentioned in Section 8.3.2: ROS for WiLMA, the Teensy 3.5 microcontrollers used in this experiment each have the physical circuitry to support up to four I2C communication channels (SDL, SCL pins). Since software optimization was not the focus of this experiment but rather physical proof of concept, more microcontrollers were freely used. However, in total, only four I2C channels are needed: one for FrED's mockup, one for the fabric machine's mockup, one for WiLMA's stepper motor, and one for WiLMA's tensioner. Although it is technically possible with a single Teensy 3.5 microcontroller to support all of the necessary I2C communications, it is recommended to use two: one for WiLMA (stepper motor and tensioner) and one for the mockups. This way, WiLMA and the mockups maintain their modular functionality and can be easily separated when necessary. In order to use the other I2C channels on a Teensy microcontroller, an additional software package for serial communication must be tested and used since the additional channels are not supported by basic Arduino libraries.

12.2 Further Testing

To better reflect the fabric machine's actual behavior, the square profile should be adjusted to include sinusoidal behavior within the pulses. This behavior is sketched in the following figure (Fig. 68). Each small sine pulse corresponds to a needle in the fabric machine while each square pulse corresponds to each row knitted in the fabric.

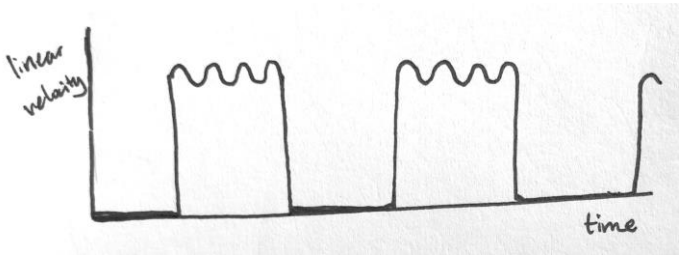


Figure 68: Sketch of modified square wave for a fabric machine profile that more closely aligns with the machine's actual behavior

In order to implement this modified square profile, control of the DC motors in the mockups must be improved. As seen in Fig. 50 and Fig. 56 showing the sample square profiles, there is fluctuation in the amplitude held that should nominally be constant. This is likely partially due to eccentric motion in the belt drive since the trend is also apparent in the constant profiles (Fig. 43 and Fig. 46). Therefore, the design of the mockup module may need to be modified in order to explore the behavior of the modified square profile. For the higher frequency trials at 1 Hz, the mockup motor does not have enough time to reach a steady-state amplitude before receiving the command to stop the motor. Therefore, the acceleration of the stepper motor can be increased using the Tic control interface to try and decrease the system's response time, which would allow the timestep to be shortened and hence increase the upper bound of the frequency range. Improving the system response is also beneficial for additional frequency testing.

In addition to testing a different profile, with the simple square profile, more frequency testing should be done to see if there is a significant trend relating the frequency to the performance of the accumulator. Aside from increasing the stepper motor's acceleration, another possible modification is replacing the DC motors in the mockup module with motors that have a higher gear ratio to slow down the motors further. In the case of the mockup for the fabric machine, replacing the DC motor to one that can run stably at a lower speed would allow more realistic testing since the lower limit in this experiment was approximately 10 mm/s while the actual fabric machine operates closer to 5 mm/s or less.

13. References

[1] 2019, “Industry 4.0 - the Nine Technologies Transforming Industrial Production,” BCG Oper. [Online]. Available: https://www.bcg.com/en-us/capabilities/operations/embracing-industry-4.0-rediscovering-growth.aspx. [Accessed: 08-Jan-2020].

[2] David Donghyun Kim, and Brian Anthony, 2017, “Design and Fabrication of Desktop Fiber Manufacturing Kit for Education,” *ASME 2017 Dynamic Systems and Control Conference - DSCC 2017*, ASME, Tysons, Virginia, USA.

[3] 2014, “Smart Manufacturing Operations Planning and Control Program,” NIST [Online]. Available: https://www.nist.gov/programs-projects/smart-manufacturing-operations-planning-and-control-program. [Accessed: 08-Jan-2020].

[4] 2018, “What’s So Smart About Smart Manufacturing?,” Programs Prof. MIT Prof. Educ. [Online]. Available: https://professional.mit.edu/news/news-listing/whats-so-smart-about-smart-manufacturing. [Accessed: 08-Jan-2020].

[5] 2019, “Smart Manufacturing,” Programs Prof. MIT Prof. Educ. [Online]. Available: https://professional.mit.edu/programs/digital-plus-programs/course-offerings/smart-manufacturing. [Accessed: 08-Jan-2020].

[6] *How to Machine Knit - Cast On*.

[7] “Silver Reed SK280 Standard Gauge Knitting Machine” [Online]. Available: https://yarn-store.com/silver-reed-sk280-knitting-machine/. [Accessed: 07-Apr-2020].

[8] “Closeup.Jpg (1600×1067)” [Online]. Available: http://4.bp.blogspot.com/_MJkIOvRaM9Y/TMIQQZoqUlI/AAAAAAAAAUo/n3kmjELMYWQ/s1600/closeup.jpg. [Accessed: 07-Apr-2020].

[9] Koranne, M., 2013, *Fundamentals of Yarn Winding*, Woodhead Publishing India Pvt Limited.

[10] Vivek Prasad Shankam Narayana, 2005, “A Novel Method for Dynamic Yarn Tension Measurement and Control in Direct Cabling Process,” Doctoral Thesis, North Carolina State University.

[11] Umirov, U., Jung, S. H., Han, C. W., and Park, J.-I., 2008, “Yarn Tension Control of Winding Machine Using Active Tensioner,” J. Inst. Control Robot. Syst., **14**, pp. 956–962.

[12] Wright, I. C., 2013, “The Design of a Yarn Tension Control System for the High Speed Formation of Conical Packages,” thesis, Loughborough University.

[13] *Video 3D High Speed Filament Production with Accumulator and 2 Station Winder Sept 2017*.

[14] 2020, “Linie Produkcyjne - Zamak Mercator,” Zamak Mercat. [Online]. Available: http://www.zamakmercator.pl/pl/oferta/linie-produkcyjne.html#1. [Accessed: 08-Jan-2020].

[15] 2020, “LTCA Accumulator,” Progress. Mach. Co. [Online]. Available: https://www.progressivewinders.com/item/traverse-spoolers/ltca-accumulator/ltca.html. [Accessed: 08-Jan-2020].

[16] “Research | Bioelectronics Group - Professor Polina Anikeeva,” Bioelectron. Res. Group [Online]. Available: https://www.rle.mit.edu/bioelectron/research/, https://www.rle.mit.edu/bioelectron/research/. [Accessed: 08-Jan-2020].

[17] Lauren A Chai, and Brian W Anthony, 2017, “Organization and Compaction of Composite Filler Material Using Acoustic Focusing,” *Proceedings of the ASME 2017 International Mechanical Engineering Congress and Exposition - IMECE2017*, ASME, Tampa, Florida, USA.

[18] Shroyer, B., “Ball Winder Maintenance,” Heavy Duty Ball Wind. NKK [Online]. Available: http://www.ballwinders.nancysknitknacks.com/maintenance.html. [Accessed: 14-Jan-2020].
[19] 2019, “AC-300/500 Accumulator Systems,” REELEX Packag. Solut. Inc [Online]. Available: http://www.reelex.com/machines/accumulators/. [Accessed: 08-Jan-2020].
[20] “PID Theory Explained - National Instruments,” Natl. Instrum. [Online]. Available: https://www.ni.com/en-us/innovations/white-papers/06/pid-theory-explained.html. [Accessed: 05-Mar-2020].
[21] “ROS/Introduction - ROS Wiki” [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Accessed: 29-Jan-2020].
[22] Keenan Wyrobek, 2017, “The Origin Story of ROS, the Linux of Robotics - IEEE Spectrum,” IEEE Spectr. Technol. Eng. Sci. News [Online]. Available: https://spectrum.ieee.org/automaton/robotics/robotics-software/the-origin-story-of-ros-the-linux-of-robotics. [Accessed: 29-Jan-2020].
[23] “ROS/Concepts - ROS Wiki” [Online]. Available: http://wiki.ros.org/ROS/Concepts. [Accessed: 29-Jan-2020].
[24] Kim, S., Kim, D. D., and Anthony, B., 2019, “Dynamic Control of a Fiber Manufacturing Process Using Deep Reinforcement Learning,” ArXiv191110286 Cs Eess.

Appendix A: Bill of Materials

Appendix A.1: Buffer Bill of Materials

Item	Part #	Vendor	Qty	Unit Cost	Cost
Teensy 3.5 (header pins included)	DEV-14056	Sparkfun	1	$28.95	$28.95
100mm Length Travel Linear Stage Actuator with Square Linear Rails with NEMA23 Stepper Motor	654615632646 (UPC)	Amazon	1	$112.15	$112.15
Stainless Steel Ball Bearing, Flanged, Shielded, NO. 686-2Z, for 6 mm Shaft Diameter	7804K143	McMaster	4	$8.74	$34.96
Tic T249 USB Multi-Interface Stepper Motor Controller (Connectors Soldered)	3138	Pololu	1	$41.95	$41.95
Snap-Action Switch with 16.7mm Lever: 3-Pin, SPDT, 5A	1402	Pololu	1	$1.25	$1.25
Filename: ModifiedMotorMount_RevB_09272019.IGS	-	SuNPe	1	$31.00	$31.00
Filename: MovingIdlerAttachment_RevB_09272019.IGS	-	SuNPe	1	$23.00	$23.00
Filename: StationaryIdlerAttachment_RevB_09272019.IGS	-	SuNPe	1	$28.00	$28.00
High-Parallel-Misalignment Flexible Shaft Coupling, Clamping Hub, 1" Overall Length, 3/4" OD, For 8mm Shaft	9889T1	McMaster	2	$15.32	$30.64
93 in.-lbs. Acetal Disc for 3/4" OD High-Parallel-Misalignment Flexible Shaft Coupling	59985K62	McMaster	1	$2.97	$2.97
Filename: GroovedIdler_RevC_10252019.IGS	-	SuNPe	2	$30.75	$61.50

Buffer Total Prototyping Cost*: $396.37

*Fasteners and 80/20 framing system components are not included.

Appendix A.2: Tensioner Bill of Materials

Item	Part #	Vendor	Qty	Unit Cost	Cost
Teensy 3.5 (header pins included)	DEV-14056	Sparkfun	1	$28.95	$28.95
Torsion Spring, 90 Degree Left-Hand Wound, 0.386" OD	9271K583	McMaster	1	$4.63	$4.63
Adafruit VL6180X Time of Flight Distance Ranging Sensor (VL6180)	3316	Adafruit	1	$13.95	$13.95
High-Load Oil-Embedded SAE 863 Bronze Sleeve Bearing, for 1/4" Shaft Diameter and 5/16" Housing ID, 3/4" Long	2868T44	McMaster	1	$0.81	$0.81

Tensioner Total Prototyping Cost*: $48.34

*Fasteners and 80/20 framing system components are not included.

Appendix A.3: Mockup Bill of Materials

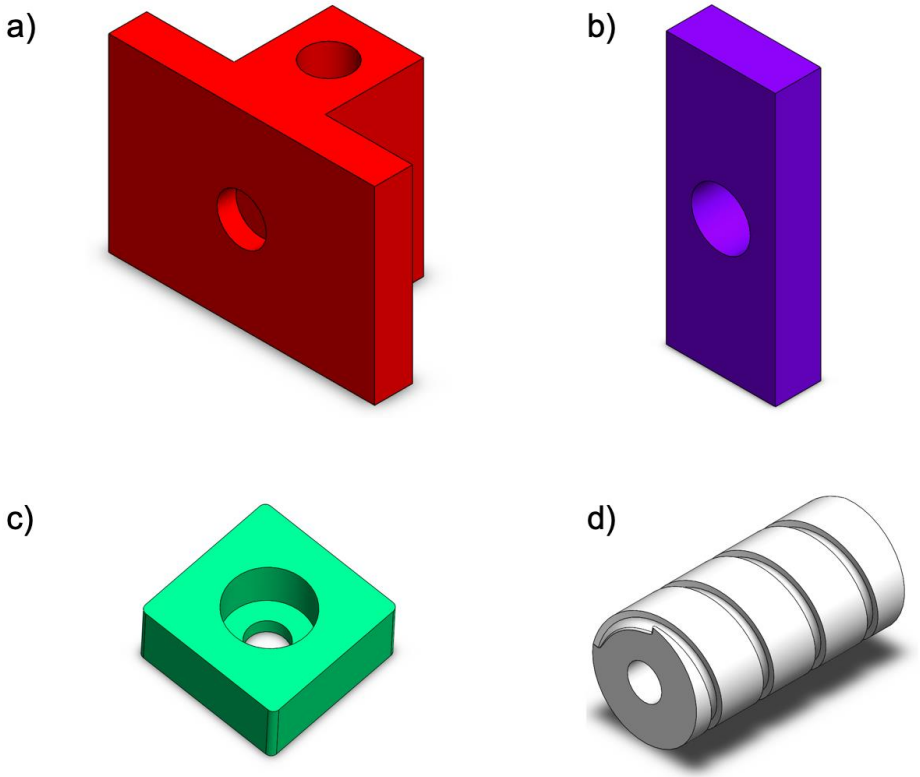
Item	Part #	Vendor	Qty	Unit Cost	Cost
Pololu Stamped Aluminum L-Bracket Pair for 37D mm Metal Gearmotors	1084	Pololu	2	$7.95	$15.90
Teensy 3.5 (header pins included)	DEV-14056	Sparkfun	1	$28.95	$28.95
High Torque Timing Pulleys MR2 Type - 2mm PowerGrip® GT®3	GPB32MR2060-B-P6	Misumi	4	$11.41	$45.64
Rotary Shafts - D Cut	NSFRV6-100-F30-T30-W5	Misumi	2	$7.75	$15.50
High Torque Timing Belts - MR2 (GT)	GBN142MR2-060	Misumi	2	$5.85	$11.70
VNH5019 Motor Driver Carrier	1451	Pololu	2	$24.95	$49.90
Stainless Steel Ball Bearing, Flanged, Shielded, NO. 686-2Z, for 6 mm Shaft Diameter	7804K143	McMaster	4	$8.74	$34.96
Multipurpose 6061 Aluminum, 1/4" Thick x 8" Wide, 1/2' feet	8975K443	McMaster	1	$12.69	$12.69
Filename: BearingBlock_A_09112019.IGS	-	SuNPe	4	$14.63	$58.52
Filename: MockupSpool_A_09112019.IGS	-	SuNPe	2	$36.25	$72.50
131:1 Metal Gearmotor 37Dx73L mm 12V with 64 CPR Encoder (Helical Pinion)	4756	Pololu	2	$39.95	$79.90

Mockup Total Prototyping Cost*: $426.16

*Fasteners and 80/20 framing system components are not included.

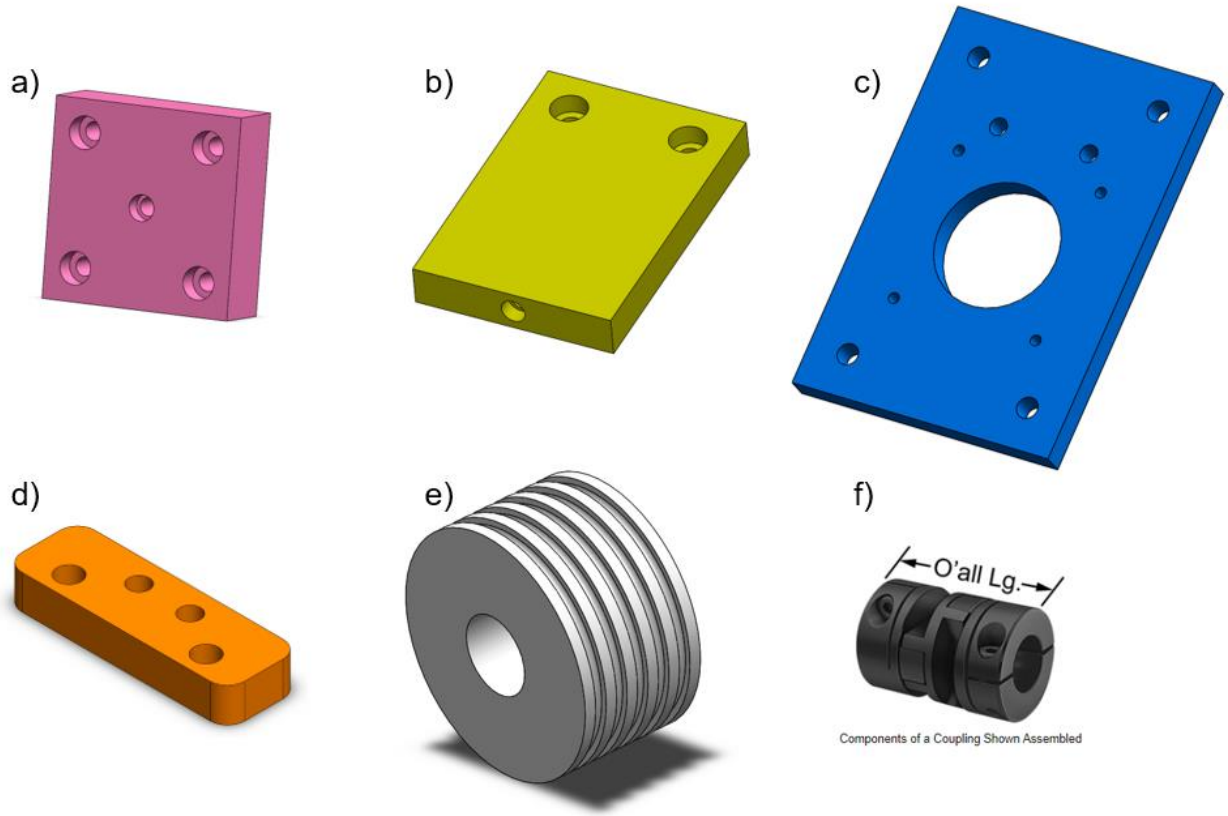
Appendix B: CAD of Individual Designed Components

Appendix B.1: CAD of Individual Designed Components in Buffer Revision A



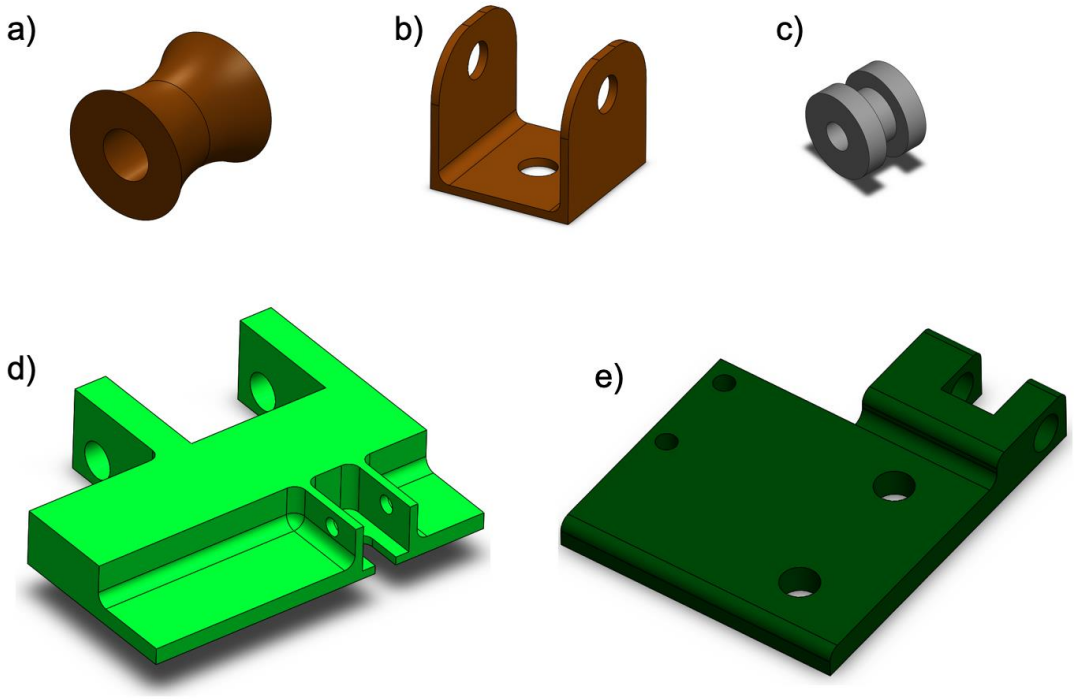
a) Nut adapter to which off-the-shelf lead screw nut is attached
b) End constraint for idler pulley; opposite the nut adapter in the assembly
c) Bearing block to attach to 80/20 framing system
d) Idler pulley

Appendix B.2: CAD of Individual Designed Components for Buffer Revision B



a) Nut adapter that mounts to an off-the-shelf linear carriage; includes a threaded hole in the center to mount the moving idler pulley
b) Cap for the off-the-shelf linear stage that includes a threaded hole to mount the stationary idler pulley
c) Motor plate to mount the stepper motor as well as 80/20 legs
d) Mount for the limit switch
e) Idler pulley
f) Off-the-shelf high-parallel-misalignment flexible shaft coupling (from McMaster, Part #: 9889T1 and 59985K62); image courtesy of McMaster-Carr

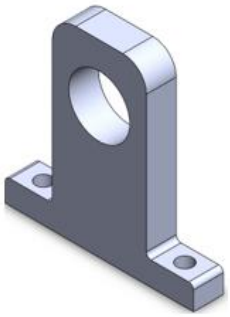
Appendix B.3: CAD of Individual Designed Components in Tensioner Revision B



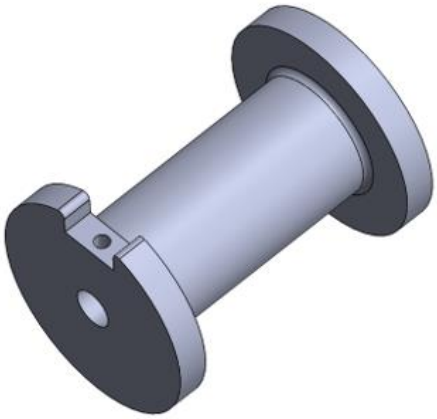
a) Positioning idler to guide fiber into the dancer
b) Mount for positioning idler; attaches to 80/20 framing system
c) Grooved bearing that contacts the fiber directly; mounts to the top half of the dancer
d) Top half of the dancer; spring-loaded against the fiber
e) Bottom half of the dancer; secures the time-of-flight sensor and mounts to the 80/20 framing system

Appendix B.4: CAD of Individual Designed Components in Mockup Module

a)



b)



a) Bearing block

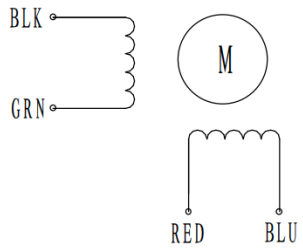
b) Spool with set screw in flange for installing shaft

Appendix C: NEMA 23 Stepper Motor Specifications

HIGH TORQUE HYBRID STEPPING MOTOR SPECIFICATIONS

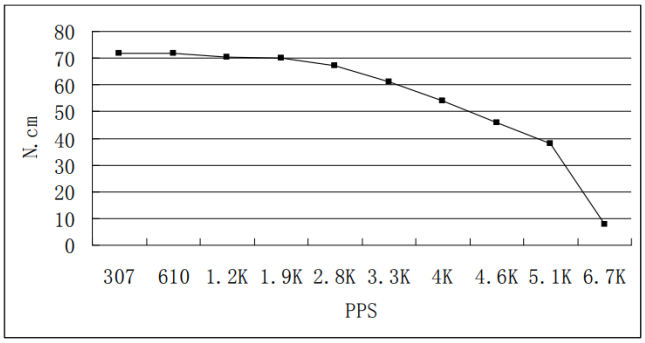
General specifications		Electrical specifications	
Step Angle (°)	1.8	Rated Voltage (V)	2.52
Temperature Rise (℃)	80 Max (rated current、2 phase on)	Rated Current (A)	2.8
Ambient Temperature (℃)	-20~+50	Resistance Per Phase (±10% Ω)	0.9 (25℃)
Number of Phase	2	Inductance Per Phase (±20% mH)	2.5
Insulation Resistance (MΩ)	100 Min (500VDC)	Holding Torque (N.cm)	126
Insulation Class	Class B		
Max.radial force (N)	28 (20mm from the flange)		
Max.axial force (N)	10		

● Wiring Diagram :



● Pull out torque curve :

VOLTAGE: 30VAC, CONSTANT CURRENT: 2.8A, HALF STEP



●

Dimensions:

(unit=mm)

Excerpt from motor documentation as provided by Pololu for Pololu Item #1474

Appendix D: Mockup DC Motor Mapping

Appendix D.1: Arduino Script for DC Motor Mapping

dcmotor_constant

```
#include <Encoder.h>

// Machine 1 = M1 = FrED; initialization of which pins are used
const int m1_dir_0 = 2;
const int m1_dir_1 = 3;
const int m1_pwm = 4;
const int m1_enc_0 = 5;
const int m1_enc_1 = 6;

Encoder m1Enc(m1_enc_0,m1_enc_1);
long m1_oldPos = 0;
float m1_now = 0;


// Machine 3 = M3 = Fabric Machine; initialization of which pins are used
const int m3_dir_0 = 7;
const int m3_dir_1 = 8;
const int m3_pwm = 9;
const int m3_enc_0 = 29;
const int m3_enc_1 = 30;

Encoder m3Enc(m3_enc_0,m3_enc_1);
long m3_oldPos = 0;
float m3_now = 0;

void setup() {
  pinMode(m1_dir_0, OUTPUT);
  pinMode(m1_dir_1, OUTPUT);
  pinMode(m1_pwm, OUTPUT);
  pinMode(m3_dir_0, OUTPUT);
  pinMode(m3_dir_1, OUTPUT);
  pinMode(m3_pwm, OUTPUT);
  Serial.begin(9600);
}


void loop() {

  // True direction is FrED reeling fiber in; false is dispensing
  bool windup = true;

  if (windup){
    digitalWrite(m1_dir_0, LOW);
    digitalWrite(m1_dir_1, HIGH);
    digitalWrite(m3_dir_0, LOW);
    digitalWrite(m3_dir_1, HIGH);
    }
  else {
    digitalWrite(m1_dir_0, HIGH);
    digitalWrite(m1_dir_1, LOW);
    digitalWrite(m3_dir_0, HIGH);
    digitalWrite(m3_dir_1, LOW);
  }
```

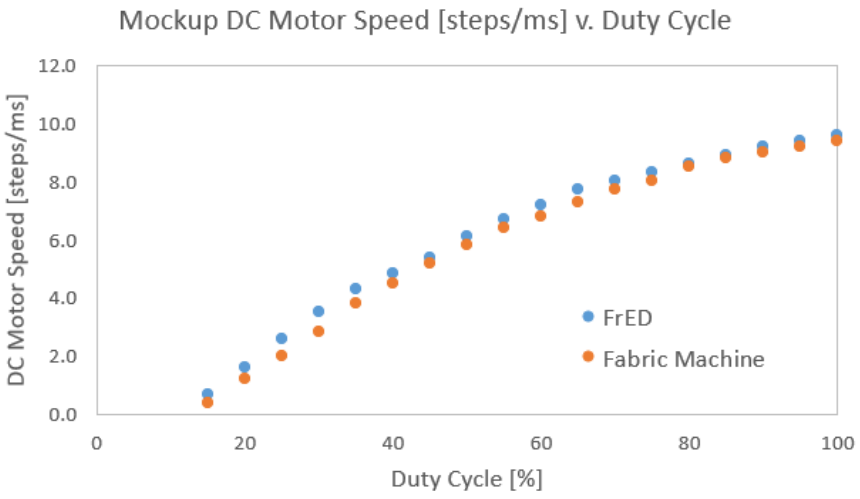
(Arduino Script for DC Motor Mapping continued)

```
  // Modify PWM/duty cycle one at a time for duty cycle mapping
  analogWrite(m1_pwm, 0);
  analogWrite(m3_pwm, 0);

  // Calculate motor speed
  long m3_newPos = m3Enc.read();
  float m3_ds = m3_newPos - m3_oldPos;
  float m3_dt = millis() - m3_now;
  float m3_vel = m3_ds/m3_dt;
  m3_now = millis();
  m3_oldPos = m3_newPos;

  long m1_newPos = m1Enc.read();
  float m1_ds = m1_newPos - m1_oldPos;
  float m1_dt = millis() - m1_now;
  float m1_vel = m1_ds/m1_dt;
  m1_now = millis();
  m1_oldPos = m1_newPos;

  // Serial Monitor to print motor speed [steps/time]
  Serial.println(m1_vel);
  delay(10);
}
```

Appendix D.2: DC Motor Mapping Data

Black represents raw values. Red represents calculated value, using spool diameter = 20 mm and steps per revolution = 8400.

Duty Cycle [%]	DC Motor Speed [steps/ms]		DC Motor Speed [rev/s]		DC Motor Speed [mm/s]	
	FrED	Fabric Machine	FrED	Fabric Machine	FrED	Fabric Machine
0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0
15	0.7	0.4	0.1	0.0	5.2	3.0
20	1.6	1.2	0.2	0.1	12.0	9.0
25	2.6	2.0	0.3	0.2	19.4	15.0
30	3.5	2.9	0.4	0.3	26.2	21.3
35	4.3	3.8	0.5	0.5	32.2	28.4
40	4.9	4.5	0.6	0.5	36.3	33.7
45	5.4	5.2	0.6	0.6	40.4	38.9
50	6.1	5.8	0.7	0.7	45.6	43.4
55	6.7	6.4	0.8	0.8	50.1	47.9
60	7.2	6.8	0.9	0.8	53.9	50.9
65	7.7	7.3	0.9	0.9	57.6	54.6
70	8.0	7.7	1.0	0.9	59.8	57.6
75	8.3	8.0	1.0	1.0	62.1	59.8
80	8.6	8.5	1.0	1.0	64.3	63.6
85	8.9	8.8	1.1	1.0	66.6	65.8
90	9.2	9.0	1.1	1.1	68.8	67.3
95	9.4	9.2	1.1	1.1	70.3	68.8
100	9.6	9.4	1.1	1.1	71.8	70.3



Appendix E: WiLMA Experiment Code

Appendix E.1: Brain Node Python Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: sslu

Filename: brain_node_v7.py

Purpose: control buffer, mockup FrED and mockup fabric machine with feedback
            control model with only one tensioner on the input to the buffer
"""

import sys
sys.path.extend(['/usr/lib/python2.7','/usr/lib/python2.7/plat-x86_64-linux-gnu','/usr/lib/python2.7/lib-tk','/usr/lib/python2.7/li
                 'usr/lib/python2.7/lib-dynload','/usr/local/lib/python2.7/dist-packages','/usr/lib/python2.7/dist-packages',
                 '/usr/lib/python2.7/dist-packages/wx-3.0-gtk3'])
import math
import time
import datetime
import numpy as np
import rospy
import csv
import pandas as pd
import io
import matplotlib.pyplot as plt
import serial
from std_msgs.msg import String
from std_msgs.msg import Int16
from std_msgs.msg import Int32
from std_msgs.msg import Float32
from mockup_dcmotor.msg import mockup_duty
from stepper_motor.msg import stepper_cmd


""" Testing Parameters """
testing = False
trialnum = 20
testspeed = 'highspeed'
testratio = '1Hz'


""" Controller Gain Parameters """
kp = 7.0e7
ki = 2.5e4
ks = 4.0e6


""" Manually Selected Parameters """
fred_revps = 0.70 # [rev/s]

## Input mockup fabric machine profile ##
mockupmode = "square"
# constant: set below; linear velocity, mm/s
# sine: sine wave; input amplitude and frequency below
# square: square wave; input amplitude and frequency below

# constant
fabric_linvel = 30 # [mm/s]

# time-varying
amp = 33 # [mm/s]
freq = 1 # Hz


""" Timing """
timestep = 100 # milliseconds; time of each loop
numtimesteps = 200
n = 5
```

(Brain Node Python Code continued)

```
""" Initializing Variables """
mode = 0
fabcounter = 0
wilma_pos = np.int32(0)
wilma_vel = np.int32(0)
calc_pulses = np.int32(0)
fred_spr = 35
prevfredf = 35
fredf = 0.5
pulses = 0
dfredf = 0
efredf = 0
sumefredf = 0
allreadings = ''
prevmode = 0
stepcounter = 0
ratio_vel = 0
fred_vel = 0
fab_vel = 0
fab_cmd = 0
pfab_cmd = 0
downcount = 0

setforce = 0.75
# 0.70 lbs for PMMA solid
# 0.75 lbs for nylon monofilament
buffersize = 13500 # for step mode 0.25


""" Data Logging """
datalog = pd.DataFrame(columns=['fredspringpos','fredspringforce','errfredf','sumerrfredf','wilma_cmd','wilma_vel','wilma_pos',
                                'mockup_deltav','calc_wilma','fred_vel','fab_vel','ratio_vel','fab_cmd','derrfredf'])
### fredspringpos = distance sensor reading for torsional spring [mm]
### fredspringforce = calculated force based on distance sensor reading [lbs]
### errfredf = setpointforce - fredspringforce [lbs]
### sumerrfredf = cumulative error [lbs]
### wilma_cmd = commanded pulse count for wilma stepper motor [pulses]
### wilma_vel = actual pulse count for wilma stepper motor [pulses]
### wilma_pos = reported position for wilma stepper motor
### mockup_deltav = actual difference in measured linear velocity of mockup fred and mockup fabric machine DC motor [mm/s]
### calc_wilma = model value of wilma speed to meet given deltav [pulse]
### ratio_vel = measured ratio of FrED to fabric machine velocity


# helper function
def enote(floatnum):
    # returns string of scientific notation
    temp = str(floatnum)
    temp = temp[:-2]
    out = ''
    zeros = 0
    for i in temp:
        if i != '0':
            out += i
        else:
            zeros += 1
    if len(out) == 1:
        res = out[0]+'e'+str(int(zeros))
    else:
        res = out[0]+'.'+out[1:]+'e'+str(int(zeros)+len(out)-1)
    return res


# data log identifier
now = datetime.datetime.now()
now = str(now)
if testing:
    # identify based on gain parameters for optimization
    identifier = '/testing/'+now+'_'+str(timestep)+'ms_kp'+enote(kp)+'_ki'+enote(ki)+'_ks'+enote(ks)
else:
    identifier = testspeed+'/'+testratio+'/'+'trial'+str(trialnum)
```

(Brain Node Python Code continued)

```
""" Serial Port Stuff for Tensioner's Adafruit VL6108X Time of Flight Sensor """

ser1port = '/dev/ttyACM0' # port for tensioner on WiLMA input
serbaud = 57600

ser1 = serial.Serial(ser1port) # tensioner for WiLMA input
ser1.port = ser1port
ser1.baudrate = serbaud


def readserial(port):
    toread = port.inWaiting()
    msg = port.read(toread)
    temp = str(msg)
    result = ''.join([i for i in temp if i.isdigit()])
    if result == '':
        result = 0
    result = int(result)
    return result


""" Helper Functions for Fabric Machine Profile """
def rpstolinvel(rps,spool_dia=20):
    # convert revolutions per second to tangential linear velocity
    out = rps*(math.pi*spool_dia)
    return out

def gensine(amp,freq,duration=100000,step=timestep):
    # generates a sine wave
    # duration default 100 seconds
    t = np.arange(duration)
    temp = 2*math.pi*freq*(t/timestep)
    out = np.sin(temp)
    return out

def genrect(amp,freq,duration=100000,step=timestep):
    # generates a square wave
    # duration default 100 seconds
    count = 0
    out = []
    period = (1000/freq)/timestep
    for i in range(duration):
        if count <= period:
            out.append(amp)
        else:
            out.append(0)
        count += 1
        if count == (2*period+1):
            count = 0
    return out


fred_linvel = rpstolinvel(fred_revps)

if mockupmode == "constant":
    fabric_linvel = fabric_linvel # [mm/s], set in line 57
elif mockupmode == "sine":
    fabprofile = gensine(amp,freq) # set in line 60, 61
elif mockupmode == "square":
    fabprofile = genrect(amp,freq) # set in line 60, 61
eff_linvel = float(fred_linvel) - float(fabric_linvel)
deltav = eff_linvel


""" Read DC Motor Mapping CSV """
df_fredmotormap = pd.read_csv("/home/sslu/catkin_ws/src/wilma/dcmotormaps/fred_dcmotormap_20mmdiaspool.csv",header=0)
df_fabmotormap = pd.read_csv("/home/sslu/catkin_ws/src/wilma/dcmotormaps/fab_dcmotormap_20mmdiaspool.csv",header=0)
```

(Brain Node Python Code continued)

```
""" Helper Functions """
def interp(x1,x2,y1,y2,y_only=True):
    # linear interpolation
    x_avg = (x1+x2)/2.
    y_avg = (y1+y2)/2.
    if y_only == True:
        return y_avg
    else:
        return [x_avg,y_avg]


def dc_quadratic(a,b,c,speed):
    # quadratic solver returning real roots only
    c = c-speed
    d = (b**2)-(4*a*c)
    if d >= 0:
        x1 = (-b+math.sqrt(d))/(2*a)
        x2 = (-b-math.sqrt(d))/(2*a)
        out = [x1,x2]
    else:
        print('No real roots.')

    for each in out:
        if each < 0 or each > 100:
            out.remove(each)
    return out


def dcmotor(velocity,machine):
    # return duty cycle to achieve a given velocty for a given machine
    if velocity == 0:
        return 0
    if machine == 'fred':
        # fitted curve for 20 mm spool diameter (R**2 = 0.99897); units: rev/s
        a = -0.00012
        b = 0.02566
        c = -0.26319
    elif machine == 'fab':
        # fitted curve for 20 mm spool diameter (R**2 = 0.99935); units: mm/s
        a = -0.00724
        b = 1.62601
        c = -20.27035
    else:
        print("Machine input incorrect. Options are 'fred' or 'fab'.")
        return None

    duty = dc_quadratic(a,b,c,velocity)[0]
    duty = int(round(duty))
    return duty


def wilma_model(velocity_diff,n=n,step_mode=0.25,stepsprev=200,pitch=5.0,dia_idler=20):
    # geometric model of WiLMA
    omega = (deltav/(2*pitch*(n-1)))
    pulses = (omega*stepsprev*10000)/step_mode
    return pulses


def spring_calc(height,leg=1.25,dancer=45.25,deg=90,maxtorque=1.28):
    # spring force calculation based on distance sensor reading
    # units: inches, lbs, degrees
    k = maxtorque/leg/float(deg)
    angle = np.arcsin(height/dancer) # radians
    angle = angle*180./math.pi # degrees
    dangle = deg-angle
    force = k*dangle
    return force
```

(Brain Node Python Code continued)

```
def getlatestspr(ser):
    # serial read tensioner's time-of-flight sensor
    global allreadings
    reading = readserial(ser)

    if reading == 0 or reading == 90:
        allreadings = allreadings + '90'
    else:
        allreadings = allreadings + str(reading)

    if len(allreadings) < 2:
        latest = 0
    elif len(allreadings) % 2 == 0:
        allreadings = allreadings[len(allreadings)-4:]
        latest = int(allreadings[-2:])
    else:
        allreadings = allreadings[len(allreadings)-3:]
        latest = int(allreadings[-3:-1])

    if latest > 90:
        latest = latest - 90
    elif latest == 90:
        latest = 0
    elif latest > 45:
        # because dancer leg length is 45.25 mm; the largest possible real reading is 45.25
        latest = 45
    return latest


""" ROS Callback Functions """
def feedback_status(data):
    global wilma_pos, wilma_vel
    wilma_pos = data.pos
    wilma_vel = data.vel


def feedback_deltavel(data):
    global deltav
    deltav = data.data
    if deltav < 0:
        deltav = 0


def feedback_ratio(data):
    global ratio_vel
    ratio_vel = data.data


def feedback_fredvel(data):
    global fred_vel
    fred_vel = data.data


def feedback_fabvel(data):
    global fab_vel
    fab_vel = data.data
```

(Brain Node Python Code continued)

```
def callback_mode(data):
    # interprets manual control from computer terminal

    global mode, prevmode
    prevmode = mode
    mode = data.data

    if mode == 0: # starting up, wait for buffer to finish homing automatically
        print('Mode 0: Start-up Homing')
    elif mode == 1: # re-home to start new experiment w/o resetting Teensy
        print('Mode 1: Re-homing')
        pub_home.publish(0)
    elif mode == 2: # main loop; runs line
        pass
    elif mode == 3: # STOP
        print('Mode 3: Stop Commanded')
        stop()
    elif mode == 4: # debugging mode
        print('Mode 4: Debugging')
    elif mode == 5: # rewind/reset sequence
        print('Mode 5: Rewind/Reset Sequence')
        rewind()
    elif mode == 6: # preload/tighten FrED only
        print('Mode 6: Tighten w/ FrED')
        preload(tight=True)
    elif mode == 7: # loosen FrED only
        print('Mode 7: Loosen w/ FrED')
        preload(tight=False)
    else:
        print('Mode Unknown: '+str(mode))
        stop()


def mockup_model():
    global fabric_linvel, fabcounter, fab_cmd, pfab_cmd
    msg = mockup_duty()
    msg.fred_duty = dcmotor(fred_revps,'fred')
    msg.timestep = timestep

    if not mockupmode == "constant":
        fabric_linvel = fabprofile[fabcounter]
        fabcounter += 1

    msg.fabric_duty = dcmotor(fabric_linvel,'fab')

    pfab_cmd = fab_cmd
    fab_cmd = dcmotor(fabric_linvel,'fab')

    return msg

""" Mode Helper Functions """
def stop():
    # stop mockup and accumulator
    global mode
    mode = 3
    mockup_msg = mockup_duty()
    mockup_msg.fred_duty = 0
    mockup_msg.fabric_duty = 0
    pub_mockup.publish(mockup_msg)

    pulses = np.int32(0)
    pub_wilma.publish(pulses)


def rewind():
    mockupmsg = mockup_duty()
    mockupmsg.fred_duty = -1*dcmotor(0.8,'fred')
    mockupmsg.fabric_duty = -1*dcmotor(44,'fab')
    pub_mockup.publish(mockupmsg)


def preload(tight=False):
    mockupmsg = mockup_duty()
    mockupmsg.fabric_duty = 0
    if tight:
        mockupmsg.fred_duty = -1*dcmotor(0.17,'fred')
        pub_mockup.publish(mockupmsg)
    else:
        mockupmsg.fred_duty = dcmotor(0.5,'fred')
    pub_mockup.publish(mockupmsg)
```

(Brain Node Python Code continued)

```
""" PI Controller """
def wilma_controller(step_mode=0.25,ethreshold=0.1,setforce=setforce,kp=kp,ki=ki,ks=ks):
    # units for force, setforce, and threshold are lbs
    global pulses, wilma_pos, deltav, fred_spr, prevfredf, fredf, dfredf, efredf, buffersize, sumefredf, fab_cmd, pfab_cmd, downcou

    if mode == 3:
        stop()

    # get tensioner/force feedback
    if (fred_spr == 0):
        print("Error: No signal from tensioner.")
        stop()

    prevfredf = fredf
    fredf = spring_calc(fred_spr)
    dfredf = prevfredf - fredf
    efredf = setforce-fredf
    sumefredf += efredf
    print("spr_range: "+str(fred_spr)+'; spr_f: '+str(fredf))

    if wilma_pos < (buffersize/step_mode):
        # WiLMA Tension Feedback Model - PI Controller
        pterm = kp*efredf # proportional term
        iterm = ki*sumefredf # integral term
        pulses = pterm+iterm
        pulses = int(pulses)
        ##### Only invoked if fabric machine has square profile
        if ((fab_cmd == 0) and (pfab_cmd != 0)) or ((fab_cmd == 0) and downcount < 5):
            pulses = int(pulses + ks*(pfab_cmd-fab_cmd))
            downcount += 1

        if downcount >= 5:
            downcount = 0
        #####

    elif wilma_pos <= 0:
        pulses = np.int32(0)
        print("Warning: Buffer minimum reached.")

    elif wilma_pos > (buffersize):
        print("Warning: Buffer is full.")
        stop()

    else:
        print("Warning: Error in WiLMA Controller.")
        stop()

    # print('wilma_pulses: '+ str(pulses))
    # print('wilma_actual: '+str(wilma_vel))
    # print('error_force: '+str(efredf))
    print('fred_vel: '+str(fred_vel))
    print('fab_vel: '+str(fab_vel))
    print('ratio: '+str(ratio_vel))
    print('pulses: '+str(pulses))

    return pulses
```

(Brain Node Python Code continued)

```
""" Timed Callback Setup """
class TimedCallback:
    def callback_main(self, event=None):
        global datalog, mode, fred_spr, stepcounter
        if mode == 2:
            if stepcounter < numtimesteps:
                print('----------------------')
                mockupmsg = mockup_model()
                pub_mode.publish(2)
                fred_spr = getlatestspr(ser1)
                wilmamsg = wilma_controller()
                pub_wilma.publish(pulses)
                pub_mockup.publish(mockupmsg)
                calc_pulses = np.int32(wilma_model(deltav))
                datalog = datalog.append({'fredspringpos':fred_spr,'fredspringforce':fredf,'errfredf':efredf,'sumerrfredf':sumefred
                    'wilma_cmd':pulses,'wilma_vel':wilma_vel,'wilma_pos':wilma_pos,'mockup_deltav':deltav,'calc_wilma':calc_pulses,
                    'fred_vel':fred_vel,'fab_vel':fab_vel,'ratio_vel':ratio_vel,'fab_cmd':fabric_linvel,'derrfredf':dfredf},ignore_
                datalog.to_csv('/home/sslu/data/simple_rect,nylon/'+identifier+'_datalog.csv')
                stepcounter += 1
                print('stepcounter: '+str(stepcounter))
            else:
                print('Done with data collection. Datalog saved.')
                stop()
                exit(0)
        elif mode == 0:
            pass
        elif mode == 4:
            # debug mode
            fred_spr = getlatestspr(ser1)
            print(allreadings)
            print('latest: '+str(fred_spr))
        elif mode == 5:
            # rewind and reset mockup
            mode = 5
            rewind()
        elif mode == 6:
            # tighten
            mode = 6
            preload(tight=True)
        elif mode == 7:
            # loosen
            mode = 7
            preload(tight=False)
        elif mode == 1:
            # re-homing buffer
            pub_home.publish(0)
        else:
            mode = 3
            stop()
```

(Brain Node Python Code continued)

```
""" ROS Setup """

# Publishers
pub_wilma = rospy.Publisher('wilma_cmd',Int32,queue_size=10)
pub_mockup = rospy.Publisher('mockup_profile',mockup_duty,queue_size=10)
pub_mode = rospy.Publisher('mode',Int16,queue_size=10)
pub_home = rospy.Publisher('home',Int16,queue_size=10)
print("Publishers initialized.")


def brain():
    global timestep

    rospy.init_node('brain', anonymous=True)

    sub_mode = rospy.Subscriber('mode',Int16,callback_mode)
    sub_pos = rospy.Subscriber('wilma_status',stepper_cmd,feedback_status)
    sub_dv = rospy.Subscriber('actual_diff',Float32,feedback_deltavel)
    sub_rv = rospy.Subscriber('ratio_vel',Float32,feedback_ratio)
    sub_m1 = rospy.Subscriber('fred_vel',Float32,feedback_fredvel)
    sub_m3 = rospy.Subscriber('fab_vel',Float32,feedback_fabvel)
    print("Subscribers initialized.")

    cb = TimedCallback()
    rospy.Timer(rospy.Duration(float(timestep)/1000.),cb.callback_main)

    # keeps Python from exiting until this node is stopped
    rospy.spin()


if __name__ == '__main__':
    try:
        brain()
    except rospy.ROSInterruptException:
        pass
```

Appendix E.2: Tensioner Arduino Code

vl6180x_mod

```
#include <Wire.h>
#include "Adafruit_VL6180X.h"

// Modified from vl6180x time-of-flight sensor example code

Adafruit_VL6180X vl = Adafruit_VL6180X();

void setup() {
  Serial.begin(57600);

  // Wait for serial port to open on native usb devices
  while (!Serial) {
    delay(1);
  }
  if (! vl.begin()) {
    while (1);
  }
}

void loop() {
  uint8_t range = vl.readRange();
  uint8_t stat = vl.readRangeStatus();

  if (stat == VL6180X_ERROR_NONE) {
    if (range < 10){
      range = range + 90;
    }
    Serial.print(range);
  }
  else {
    Serial.print(90);
  }
}
```

Appendix E.3: Buffer Arduino Code

buffer_v6

```
#include <ros.h>
#include <std_msgs/Int16.h>
#include <std_msgs/Int32.h>
#include <Tic.h>
#include <stepper_motor/stepper_cmd.h>

// Buffer ROS node - controls WiLMA's stepper motor

// ROS General and Publishers Setup
ros::NodeHandle  nh;
stepper_motor::stepper_cmd wilma_status;
ros::Publisher pub_status("wilma_status", &wilma_status);

int mode = 0;
int timestep = 100;
int homing = 0;

// Machine 2 = M2 = Buffer; initialization of which pins are used
TicI2C tic;
const int limswitch = 32;
int limswitch_st = 0;
long pulses = 0;
const int led = 13;
int pos = 0;
float stepmode = 0.25;


// Sends a "Reset command timeout" command to the Tic.  We must
// call this at least once per second, or else a command timeout
// error will happen.  The Tic's default command timeout period
// is 1000 ms, but it can be changed or disabled in the Tic
// Control Center.
void resetCommandTimeout()
{
  tic.resetCommandTimeout();
}

// Delays for the specified number of milliseconds while
// resetting the Tic's command timeout so that its movement does
// not get interrupted by errors.
void delayWhileResettingCommandTimeout(uint32_t ms)
{
  uint32_t start = millis();
  do
  {
    resetCommandTimeout();
  } while ((uint32_t)(millis() - start) <= ms);
}

// Update mode if it changes
void mode_change(const std_msgs::Int16& msg){
  mode = msg.data;
}
```

(Buffer Arduino Code continued)

```
// Main callback function for stepper motor movement
void stepper_ctrl(const std_msgs::Int32& msg){
  pulses = msg.data;

  if (mode == 3){ // Stop commanded
    pulses = 0;
  }

  // Main operation: calculates velocity based on position change;
  // Publishes velocity and position
  else if (mode == 2){
    int ipos = tic.getCurrentPosition();
    float itime = millis();
    tic.setTargetVelocity(pulses);
    delay(timestep/2);
    int fpos = tic.getCurrentPosition();
    float ftime = millis();
    float dt = ftime - itime;
    int dpos = fpos - ipos;
    float tempv = dpos*1000*(pow(10,4))/dt;
    int vel = round(tempv);
    wilma_status.pos = fpos;
    wilma_status.vel = vel;
    pub_status.publish(&wilma_status);
  }
}


// Callback when homing sequence is commanded
void home_callback(const std_msgs::Int16& msg){
  homing = 0;
}
```

(Buffer Arduino Code continued)

```
// ROS Subscriber Setup
ros::Subscriber<std_msgs::Int32> sub_wilma("wilma_cmd", &stepper_ctrl);
ros::Subscriber<std_msgs::Int16> sub_mode("mode", &mode_change);
ros::Subscriber<std_msgs::Int16> sub_home("home", &home_callback);

void setup() {
  nh.initNode();
  nh.advertise(pub_status);
  nh.subscribe(sub_wilma);
  nh.subscribe(sub_mode);

  pinMode(limswitch,INPUT_PULLDOWN);

  Wire.begin(); //set up I2C
  delay(10); //give Tic some time to start up (in example code)
  tic.exitSafeStart();

  Serial.begin(57600);
  pinMode(led, OUTPUT);
}

void loop() {
  if (homing == 0){ // Run homing sequence
    delay(1000);
    // Keep driving carriage up until limit switch is triggered
    while (limswitch_st==0) {
      tic.setTargetVelocity(-2000000);
      delayWhileResettingCommandTimeout(200);
      limswitch_st = digitalRead(limswitch);
      delay(1);
    }
    tic.haltAndSetPosition(0);
    homing = 1;
  }
  delay(1);
  nh.spinOnce();
}
```

Appendix E.4: Mockup Arduino Code

mockup_v3

```
#include <ros.h>
#include <std_msgs/Float32.h>
#include <mockup_dcmotor/mockup_duty.h>
#include <Encoder.h>

// Mockup ROS Node - Controls DC motors in mockup module

// ROS General and Publisher Setup
ros::NodeHandle  nh;
std_msgs::Float32 deltav_msg;
std_msgs::Float32 ratiov_msg;
std_msgs::Float32 fredv_msg;
std_msgs::Float32 fabv_msg;
ros::Publisher pub_dv("actual_diff", &deltav_msg);
ros::Publisher pub_rv("ratio_vel", &ratiov_msg);
ros::Publisher pub_m1("fred_vel", &fredv_msg);
ros::Publisher pub_m3("fab_vel", &fabv_msg);

float spool_dia = 20; // [mm]
float j = 8400; // counts per revolution for 131:1 gear ratio DC motor

// Machine 1 = M1 = FrED; initialization of which pins are used
const int m1_dir_0 = 2;
const int m1_dir_1 = 3;
const int m1_pwm = 4;
const int m1_enc_0 = 5;
const int m1_enc_1 = 6;
Encoder m1Enc(m1_enc_0,m1_enc_1);
long m1_oldPos = 0;
float m1_now = 0;
int m1_f = 1;
int fred_pwm = 0;

// Machine 3 = M3 = Fabric Machine; initialization of which pins are used
const int m3_dir_0 = 7;
const int m3_dir_1 = 8;
const int m3_pwm = 9;
const int m3_enc_0 = 29;
const int m3_enc_1 = 30;
Encoder m3Enc(m3_enc_0,m3_enc_1);
long m3_oldPos = 0;
float m3_now = 0;
int m3_f = 1;
int fabric_pwm = 0;
```

(Mockup Arduino Code continued)

```
// Main callback function: command motor movement
// Publishes each motor's actual velocity
void mockup_ctrl(const mockup_dcmotor::mockup_duty& msg){
  if (msg.fred_duty >= 0){
    digitalWrite(m1_dir_0, HIGH);
    digitalWrite(m1_dir_1, LOW);
    digitalWrite(m3_dir_0, HIGH);
    digitalWrite(m3_dir_1, LOW);

    fred_pwm = msg.fred_duty;
    fabric_pwm = msg.fabric_duty;
  }

  else if (msg.fred_duty < 0){
    digitalWrite(m1_dir_0, LOW);
    digitalWrite(m1_dir_1, HIGH);
    digitalWrite(m3_dir_0, LOW);
    digitalWrite(m3_dir_1, HIGH);

    fred_pwm = -1*msg.fred_duty;
    fabric_pwm = -1*msg.fabric_duty;
  }

  if (m1_dir_0 == HIGH){
    m1_f = 1;
  }
  else {
    m1_f = -1;
  }
  if (m3_dir_0 == HIGH){
    m3_f = 1;
  }
  else {
    m3_f = -1;
  }

  analogWrite(m1_pwm, fred_pwm);
  analogWrite(m3_pwm, fabric_pwm);

  long m1_newPos = m1Enc.read();
  float m1_ds = m1_newPos - m1_oldPos;
  float m1_dt = millis() - m1_now;
  float m1_angvel = m1_f*m1_ds/m1_dt;
  m1_now = millis();
  m1_oldPos = m1_newPos;
```

(Mockup Arduino Code continued)

```
  long m3_newPos = m3Enc.read();
  float m3_ds = m3_newPos - m3_oldPos;
  float m3_dt = millis() - m3_now;
  float m3_angvel = m3_f*m3_ds/m3_dt;
  m3_now = millis();
  m3_oldPos = m3_newPos;

  float m1_vel = m1_angvel*1000.0*spool_dia*PI/j;
  fredv_msg.data = m1_vel;
  pub_m1.publish(&fredv_msg);
  float m3_vel = m3_angvel*1000.0*spool_dia*PI/j;
  fabv_msg.data = m3_vel;
  pub_m3.publish(&fabv_msg);
  float delta_vel = m1_vel - m3_vel;
  deltav_msg.data = delta_vel;
  pub_dv.publish(&deltav_msg);
  ratiov_msg.data = m1_vel/m3_vel;
  pub_rv.publish(&ratiov_msg);
}

// ROS Subscribers Setup
ros::Subscriber<mockup_dcmotor::mockup_duty> sub_main("mockup_profile",mockup_ctrl);

void setup() {
  nh.initNode();
  nh.advertise(pub_dv);
  nh.advertise(pub_rv);
  nh.advertise(pub_m1);
  nh.advertise(pub_m3);
  nh.subscribe(sub_main);

  pinMode(m1_dir_0, OUTPUT);
  pinMode(m1_dir_1, OUTPUT);
  pinMode(m1_pwm, OUTPUT);

  pinMode(m3_dir_0, OUTPUT);
  pinMode(m3_dir_1, OUTPUT);
  pinMode(m3_pwm, OUTPUT);

  Serial.begin(57600);
}

void loop() {
    nh.spinOnce();
    delay(1);
}
```

Appendix F: WiLMA Operating Procedure

Appendix F.1: Initial Setup

1. Plug in the 12V power supply.
2. Open Tic stepper motor control software (on the PC) and connect the correct USB (may need to disconnect that USB port from the VM). To begin, de-energize the stepper motor.
3. Start VM if not already started. Open terminal and run this command: `roscore`
4. Open four additional terminals and change the source in each of them with this command:
   ```
   source ~/catkin_ws/devel/setup.bash
   ```
5. Turn on the USB port labeled "Tens" to connect the tensioner's Teensy. In Arduino, check that the port selected is /dev/ttyACM0. Upload the tensioner code (vl6180x_mod.ino).
6. Turn on the USB port labeled "Mock" to connect the mockup's Teensy. In Arduino, check that the port selected is /dev/ttyACM1. Upload the mockup code (mockup_v3.ino).
7. In one of the terminals, start the mockup node by running this command:
   ```
   rosrun rosserial_python mockup_node.py /dev/ttyACM1 _baud:=57600
   ```
8. Turn on the USB port labeled "Buffer" to connect the buffer's Teensy. If the motor begins moving, check that the motor is de-energized in the stepper motor control software to pause the homing sequence until setup is complete.
9. In Arduino, check that the port selected is /dev/ttyACM2. Upload the buffer code (buffer_v6.ino).
10. In the brain node Python code, input the desired settings, parameters, etc. at the beginning of the file.
11. In another terminal, start the brain node by running this command:
    ```
    rosrun wilma brain_node_v7.py
    ```
12. In another terminal, start the buffer node by running this command:
    ```
    rosrun rosserial_python buffer_node.py /dev/ttyACM2 _baud:=57600
    ```
13. From the Tic stepper motor control software, re-energize the stepper motor to begin the homing sequence. Wait for the homing sequence to finish.
14. If the fiber is not threaded through the accumulator at this point, unload the fiber from the FrED mockup, wind it through the accumulator, noting the number of grooves used (set as n in the brain node Python code), and tape it to the fabric machine mockup's spool. To unload fiber from the FrED mockup, run this command in the remaining unused terminal:
    ```
    rostopic pub mode std_msgs/Int16 7 --once
    ```

15. If the fiber is already threaded through the accumulator, preload the fiber using the FrED mockup spool so that the fiber is in contact with the tensioner. To preload the fiber then stop the preloading, run these commands in the terminal dedicated for rostopic publishing:

```
rostopic pub mode std_msgs/Int16 6 --once
rostopic pub mode std_msgs/Int16 3 --once
```

At this point, the setup is complete.

Appendix F.2: Normal Operation

1. After the initial setup is complete, double-check the desired input settings, parameters, etc. in the brain node Python code. If there is a mistake, `ctrl+z` to kill the brain node code in the terminal, correct the mistake, and restart the brain node code.
2. To start the production line, publish the mode message 2 by running this command:

```
rostopic pub mode std_msgs/Int16 2 --once
```

3. Monitor the print statements. If this is the first run for a certain setting, check that the print statements of the mockups' velocities and ratios match the desired nominal values. If not, stop the production line by publishing the mode message 3. Stop the brain node code and adjust the nominal values as necessary before restarting.
4. Once the values are adjusted, data collection may begin. To start the production line, publish mode message 2. All the machines will stop automatically when the trial is complete or if the buffer becomes full prematurely. The data is saved and outputted to CSV at every timestep, so it can be accessed even if the trial cannot run to completion.
5. To restart the machines for the next trial, press the reset button on the buffer's Teensy and wind up the slack from the buffer using the FrED mockup spool by publishing the mode message 6.
6. If the FrED mockup spool is close to empty, rewind the fiber back from the fabric machine mockup by publishing the mode message 5.

Occasionally, the buffer node will show a message that says, "Unable to sync with device" or "Lost sync." Simply, kill the buffer node and restart it. To fully kill the node and associated serial ports, use the following command `kill %` after exiting the node with `ctrl+z`. Then, re-upload the Arduino code, checking that the Serial port is /dev/ttyACM2.

Appendix G: Data Analysis Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: sslu

Filename: analyze.py

Purpose: extract grand statistics from 20 trials of specified setting
"""

import math
import numpy as np
import csv
import pandas as pd
import matplotlib.pyplot as plt

testspeed = 'highspeed'
testtype = 'square,nylon'

# constant only
testratio = '1,25x'

# square only
testfreq = '1Hz'
testfred = '1,5'


if 'constant' in testtype:
    # constant
    filepath = testtype+'/'+testspeed+'/'+testratio
else:
    # square
    if testspeed == 'lowspeed':
        filepath = testtype+'/'+testspeed+'/'+testfreq+'/FrED'+testfred+'x'
    else:
        filepath = testtype+'/'+testspeed+'/'+testfreq


# aggregate data frame for results of each trial
aggdf = pd.DataFrame(columns=['avg_error','std_error','summed_error',
                              'avg_force','std_force','avg_ratio','std_ratio'])
### avg_error = average error value in spring force of tensioner
### std_error = standard deviation of error (tensioner)
### summer_error = cumulative/summed error for the whole trial
### avg_force = average spring force applied to fiber
### std_force = standard deviation of spring force applied to fiber
### avg_ratio = average ratio of FrED to fabric machine velocity
### std_ratio = standard deviation of ratio of FrED to fabric machine velocity


```

(Data Analysis Code continued)

```
filenum = 1
for i in range(1,21):
    filenum = i
    trialpath = filepath + '/'+'trial'+str(filenum)+'_datalog.csv'
    tempdf = pd.read_csv(trialpath,header=0)

    forcelist = tempdf['fredspringforce'].values.tolist()
    errorlist = tempdf['errfredf'].values.tolist()
    ratiolist = tempdf['ratio_vel'].values.tolist()
    summedlist = tempdf['sumerrfredf'].values.tolist()

    ratiolist_clean = []
    for x in ratiolist:
        if x == '#NAME?':
            x = float('inf')
        else:
            x = float(x)

        if x == float('inf') or x == float('-inf') or math.isnan(x):
            pass
        else:
            ratiolist_clean.append(x)

    if 'square' in testtype:
        fredratio = testfred
        fredratio = float(fredratio.replace(',','.'))
        templist = ratiolist_clean
        ratiolist_clean = []

        for x in templist:
            if x < 2*fredratio and x > fredratio/2:
                ratiolist_clean.append(x)


    avg_error = np.average(errorlist)
    std_error = np.std(errorlist)
    summed_error = summedlist[-1]
    avg_force = np.average(forcelist)
    std_force = np.std(forcelist)
    avg_ratio = np.average(ratiolist_clean)
    std_ratio = np.std(ratiolist_clean)

    aggdf = aggdf.append({'avg_error':avg_error,'std_error':std_error,
                          'summed_error':summed_error,'avg_force':avg_force,
                          'std_force':std_force,'avg_ratio':avg_ratio,
                          'std_ratio':std_ratio},ignore_index=True)

    print('Processing Trial: '+str(i))

aggdf.to_csv(filepath+'/'+'aggregate.csv')
print('Done: aggregate.csv saved.')
```

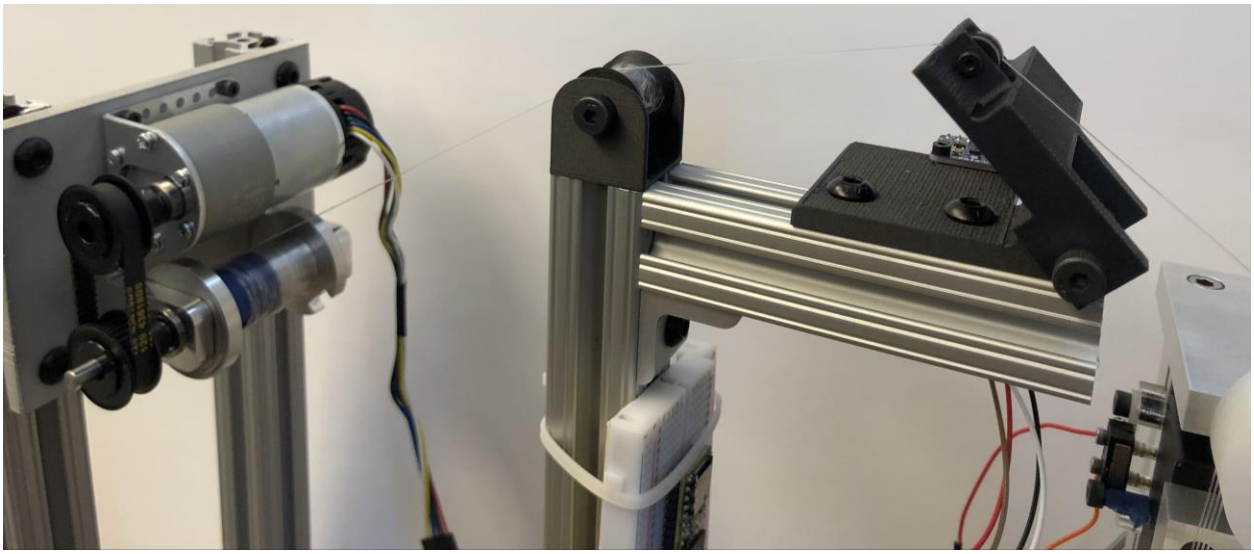
Appendix H: Additional Videos and Pictures of WiLMA

Video Documentation:

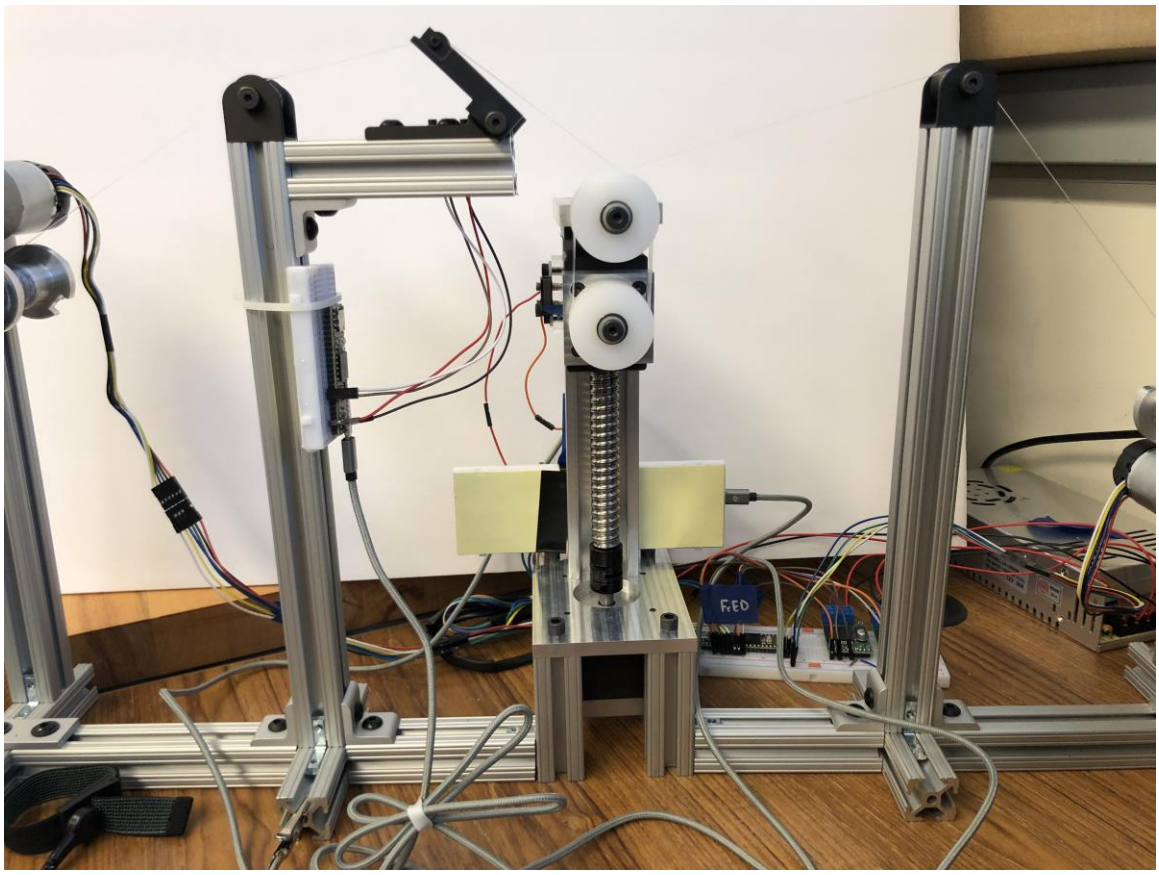
- Walkaround, showing WiLMA's components in operation
- Demonstration of WiLMA operation with constant fabric machine profile
- Demonstration of WiLMA operation with square fabric machine profile
- Demonstration of reset and preload sequence
- Demonstration of commercial yarn ball winder winding PMMA fiber
- Demonstration of commercial yarn ball winder with low friction fiber slipping

Photo Documentation:

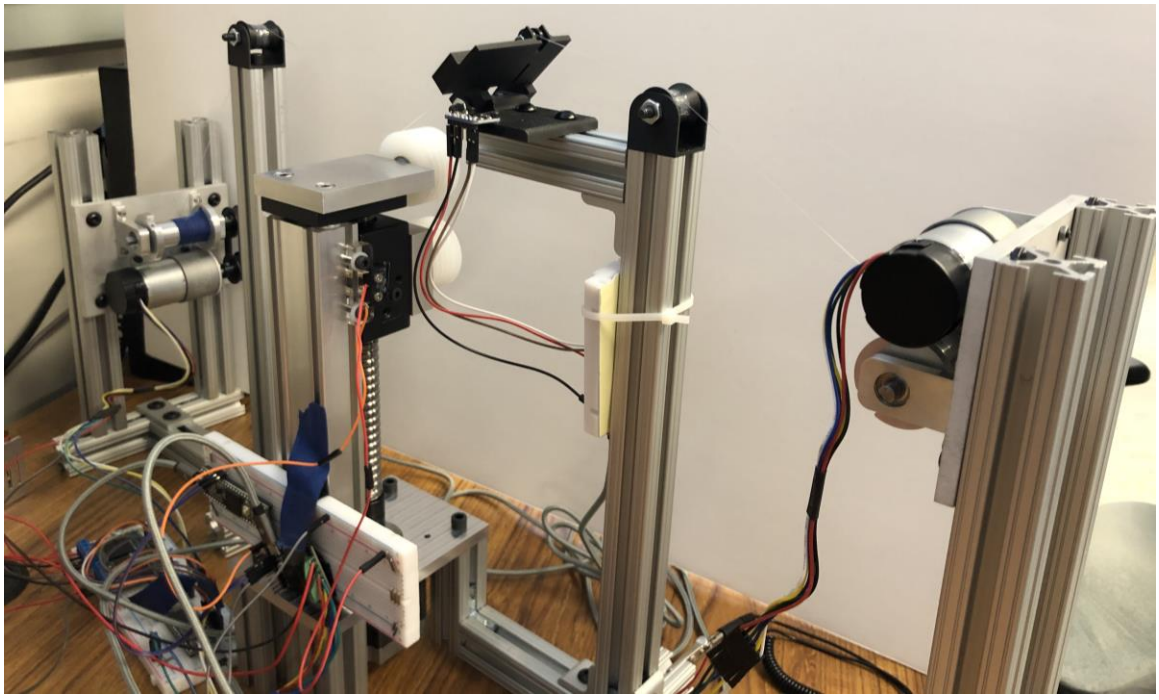
a) Close-up of FrED mockup and tensioner modules



b) Front view of WiLMA (buffer and tensioner)



c) Back view of WiLMA (buffer and tensioner)



d) Close-up of limit switch attachment and idler pulleys