DEMONSTRATION OF VULNERABILITIES IN GLOBALLY DISTRIBUTED ADDITIVE MANUFACTURING

Charles Ellis Norwood

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Mechanical Engineering

Jan Helge Bøhn, Chair
Christopher B. Williams
Patrick R. Schaumont

May 11, 2020
Blacksburg, Virginia

Keywords: 3D Printing, Additive Manufacturing, Security, Distributed Manufacturing

DEMONSTRATION OF VULNERABILITIES IN GLOBALLY DISTRIBUTED ADDITIVE MANUFACTURING

Charles Norwood

Abstract

Globally distributed additive manufacturing is a relatively new frontier in the field of product lifecycle management. Designers are independent of additive manufacturing services, often thousands of miles apart. Manufacturing data must be transmitted electronically from designer to manufacturer to realize the benefits of such a system. Unalterable blockchain legers can record transactions between customers, designers, and manufacturers allowing each to trust the other two. Although trust can be established, malicious printers or customers still have the incentive to produce unauthorized parts. To prevent this, machine instructions are encrypted and electronically transmitted to the printing service, where an authorized printer decrypts the data and prints an approved number of parts or products. The encrypted data may include G-Code machine instructions which contain every motion of every motor on a 3D printer. Once these instructions are decrypted, motor drivers send control signals along wires to the printer's stepper motors. The transmission along these wires is no longer encrypted. If the signals along the wires are read, the motion of the motor can be analyzed, and G-Code can be reverse engineered.

This thesis demonstrates such a threat through a simulated attack on a G-Code controlled device. A computer running a numeric controller and G-Code interpreter is connected to standard stepper motors. As G-Code commands are delivered, the magnetic field generated by the transmitted signals is read by a Hall Effect sensor. The rapid oscillation of the magnetic field corresponds to the stepper motor control signals which rhythmically move the motor. The oscillating signals are recorded by a high speed analog to digital converter attached to a second computer. The two systems are completely electronically isolated.

The recorded signals are saved as a string of voltage data with a matching time stamp. The voltage data is processed through a Matlab script which analyzes the direction the motor spins and the number of steps the motor takes. With these two pieces of data, the G-Code instructions which produced the motion can be recreated. The demonstration shows the exposure of previously encrypted data, allowing for the unauthorized production of parts, revealing a security flaw in a distributed additive manufacturing environment.

DEMONSTRATION OF VULNERABILITIES IN GLOBALLY DISTRIBUTED ADDITIVE MANUFACTURING

Charles Norwood

General Audience Abstract

Developed at the end of the 20th century, additive manufacturing, sometimes known as 3D printing, is a relatively new method for the production of physical products. Typically, these have been limited to plastics and a small number of metals. Recently, advances in additive manufacturing technology have allowed an increasing number of industrial and consumer products to be produced on demand. A worldwide industry of additive manufacturing has opened up where product designers and 3D printer operators can work together to deliver products to customers faster and more efficiently. Designers and printers may be on opposite sides of the world, but a customer can go to a local printer and order a part designed by an engineer thousands of miles away. The customer receives a part in as little time as it takes to physically produce the object. To achieve this, the printer needs manufacturing information such as object dimensions, material parameters, and machine settings from the designer. The designer risks unauthorized use and the loss of intellectual property if the information is exposed.

Legal protections on intellectual property only go so far, especially across borders. Technical solutions can help protect valuable IP. In such an industry, essential data may be digitally encrypted for secure transmission around the world. This information may only be read by authorized printers and printing services and is never saved or read by an outside person or computer. The control computers which read the data also control the physical operation of the printer. Most commonly, electric motors are used to move the machine to produce the physical object. These are most often stepper motors which are connected by wires to the controlling computers and move in a predictable rhythmic fashion. The signals transmitted through the wires generate a magnetic field, which can be detected and recorded. The pattern of the magnetic field matches the steps of the motors. Each step can be counted, and the path of the motors can be precisely traced. The path reveals the shape of the object and the encrypted manufacturing instructions used by the printer. This thesis demonstrates the tracking of motors and creation of encrypted machine code in a simulated 3D printing environment, revealing a potential security flaw in a distributed manufacturing system.

ACKNOWLEDGEMENTS

Thank you to my parents, Matthew and Amy Norwood, for a lifetime of unconditional support.

Thank you to my advisor, Dr. Jan Helge Bøhn, committee members Dr. Christopher Williams, and Dr. Patrick Schaumont, and all my professors at Virginia Tech for expanding my understanding of engineering and the world.

Thank you to my friends in Blacksburg and around the country for making life outside the lab relaxing and fulfilling.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Over the past three centuries, product development and manufacturing have undergone several radical changes, revolutionizing economies and standards of living. Before the industrial revolution, goods were created by individual craftsman who often manufactured a single product from start to finish [1]. These craftsmen had to develop manufacturing skills and techniques over years of practice and apprenticeship. Even the most highly skilled craftsmen could not reliably form identical parts or products at a large scale. The first industrial revolution introduced reliable, repeatable manufacturing techniques, interchangeable parts, and the foundation of today's product development and manufacturing systems. The large central factory became a staple of society and drove the development of today's cities and ways of life [1].

Decentralized manufacturing transformed the way products are engineered, produced, and sold. Today it is common for a product to be manufactured on the opposite side of the world from where it was engineered. Manufacturers can find the most cost-effective location to produce a product depending on labor, material, or energy costs. Modern technological advances may allow manufacturing to move out of large factories and may once again revolutionize the industry. Additive manufacturing and the internet can free industry from the limitations of the central factory [1].

With the rapid development of additive manufacturing technologies, increasingly complex products can be produced autonomously and on demand. Additive manufacturing, also known as 3D printing, uses complex machines to repeatedly form thin layers of material. The shape of each layer depends on the topology of the product and as layers are added, the product is formed.

The development process for additive manufacturing starts with the creation of a 3D virtual model of the product in computer aided design (CAD) software. The 3D model is then "sliced" into horizontal layers. The shape of these layers is translated into machine readable code and sent to the additive manufacturing system, which deposits the successive layers, producing a real object.

Industrial parts are increasingly produced by additive manufacturing [2]. Several studies have shown the potential for additive manufacturing to improve supply chains for spare parts[3] [4, 5]. With advances in additive manufacturing and the rise of globally distributed manufacturing, the logical step is to combine the two into globally distributed additive manufacturing. In one potential model of this system, designers and manufacturers would be two different entities in different places in the world. A design firm could engineer and develop a product, then hire an independent manufacturer to produce it. Designers could have as much or as little manufacturing capability as they need, and manufacturers could function solely to produce parts [6].

With such a system, the major advantages of globally distributed manufacturing are realized. If a consumer decides to purchase a product, the original design firm could find the manufacturer closest to the consumer and have it produced. With additive manufacturing, products can be produced on demand, so an ideal system could effectively eliminate the need to keep stocks of parts or products. Consequently, it would reduce all associated costs such as warehouses and logistics for part organization [7]. With local manufacturing, the cost of shipping a part from manufacturer to consumer could be significantly reduced. Transportation costs and emissions are lessened [6]. Also, outdated or very low volume parts could become much cheaper as they can be stored as computer data, not physical products [7]. As additive manufacturing develops, more and more products could utilize such a system.

Many potential drawbacks must be accounted for. If a manufacturer is independent of a product designer, it may have no ownership over the intellectual property (IP) it is printing. A malicious manufacturer could represent a threat to the original IP owner if it steals the designs and manufactures unauthorized parts. Practically speaking, this threat would only be present across borders. An American company giving IP access to an American manufacturer knows that IP theft would be punishable under U.S. law. However, countries like China are notorious for their disregard for intellectual property laws [8]. There could be little legal recourse for international IP theft. Rather than legal solutions, technological solutions could help protect IP when transmitted across international borders.

A key requirement in a globally distributed additive manufacturing system is the transmission of data. All information necessary for successful manufacturing of a part must be transmitted from the creator of that information to the printing service. This information can

include model data ranging from original CAD files to AM machine instructions known as G-code. It can also include manufacturing parameters, materials, or other instructions. To ensure the secure transmission of this data, it may be encrypted by the creator and transmitted over the internet to the printing service, which can decrypt and manufacture the part [9]. However, transmission of data, even if encrypted, presents a potential weakness in the security of a distributed additive manufacturing system. A break in this system could lead to intellectual property theft. If an unauthorized entity gets ahold of CAD files or other manufacturing data, that entity could potentially manufacture pirated copies of a part or product. Such an unauthorized entity could even be the manufacturer.

Several proposals exist to prevent the production of unauthorized or pirated parts. The most common is the use of fingerprinting or watermarking. During the printing process a unique geometry, label, or other identifier is manufactured into the product. Only one specific part can have that specific "fingerprint", so unauthorized parts can be identified more easily. These fingerprints can take the form of unique structures within the part, radio frequency identification (RFID) tags, and patterned materials, among many others [10]. Fingerprints must not interfere with the function of the part, must be manufactured in the same process as the part, and must be difficult to replicate. They also must be readable by someone verifying a legitimate part.

Data encryption and part fingerprinting are two components necessary to secure a distributed additive manufacturing system against unauthorized parts and IP theft. The information flow of all manufacturing data, customers, sellers, fingerprints, and part histories must also be tracked. This information can be recorded and held in what is known as a smart contract [10]. This smart contract would allow a designer, part manufacturer, and customer to work together throughout the product lifecycle. Even though they may not know and may not trust each other, if each trusts the smart contract and the technology that secures it, business can be done.

Ideally, encryption of manufacturing data could be secure enough that it would not be financially viable to steal the data during transmission from the designer to the printer. Modern encryption is effectively un-hackable. However, once the secure printer decrypts the printing information, the information becomes vulnerable. The decrypted data is converted to machine instructions, which control the operation of the printer. The G-Code instructions control the paths of the motors, heaters, and other processes of the printer. Printer software converts the G-

Code to stepper motor control signals, which motor drivers use to operate the stepper motors. Motor drivers send power signals along wires to the stepper motors, which control the physical motion of the printer as it prints parts. These power signals present vulnerability for a secure additive manufacturing system. This thesis will attempt to investigate this vulnerability through a demonstrated attack.

1.1 Problem Statement

The objective of this research is to demonstrate an attack where electrical signals used to control a stepper motor can be measured, recorded, and analyzed to recreate the G-Code needed to manufacture a 3D printed object. If one can extract the G-Code from a secure system, a high value object may be printed without the authorization of the designer or owner of the intellectual property.

1.2 Proposed Solution

Manufacturing data for 3D printing can be securely encrypted up to the point of the 3D printer controller. Once decrypted, the printer controller sends signals to motor drivers which control operation of the 3D printer. Motor drivers send electronic pulses down wires to stepper motors, which turn based on the pulses received. Coordinated rotation of the stepper motors physically moves the print head and other components along the path necessary to create the desired product. If one can independently recreate and synchronize the movement of all motors in a printer, one may be able to reproduce a 3D printed object.

The pulses in the wires connecting the motor drivers to the stepper motors will create an induced magnetic field according to established electromagnetic principles. These pulses are transmitted in a regular, rhythmic pattern due to the fundamental operating principles of a stepper motor. Without removing or destroying any components of the 3D printer, one can detect these induced magnetic fields and record the pattern of pulses being transmitted. Once the pattern of pulses is recorded and interpreted, the motion of the stepper motor can be reverse engineered. With the intended motion known, the G-Code which commanded the motor motion can be recreated. With correctly reverse-engineered G-Code, a manufacturer could subsequently build as many parts as he or she wishes.

This type of attack would typically only be possible in a globally distributed manufacturing setting. A successful attack would require a high level of access to a secure printer. An attacker would need to work in and around a machine to properly detect electromagnetic signals from stepper motor wires.

The attack proposed by this thesis must be demonstrated to be considered a legitimate security threat. The demonstration will use two separate systems: a simulated secure additive manufacturing device, and a hacking device. The AM device will consist of a computer and an X and Y motor. The computer will use software to interpret and deliver G-Code. It will not actually be delivering encrypted data; it will merely produce the motor signals that would be found in a secure additive manufacturing machine. The hacking device will consist of current sensors attached to a data recording device. These current sensors will detect the magnetic field produced by motor pulses. The sensors will output a signal which is filtered and recorded by the hacking device. Finally, the data recorded by the hacking device will be exported to a PC running Matlab which will process the data and return important information such as motor steps and motor direction. The PC will use the motor steps and direction information to reproduce sequences of G-Code instructions. A variety of G-Code commands will be issued and recorded to ensure the hacking device can reproduce G-Code for any distance and direction. A successful demonstration will prove the viability of an attack using stepper motor control signals.

1.3 Thesis Outline

This thesis will demonstrate an attack on a 3D printer. The attack will reverse engineer manufacturing instructions by analyzing electrical signals transmitted between motor drivers and stepper motors. The attack will be verified by comparing the motion of the original motor and the reconstructed signal. The thesis will consist of the following sections.

Chapter 1 provides a background on the history of distributed manufacturing and the foundations of globally distributed additive manufacturing. It also covers proposed security solutions and background for the vulnerability this thesis will attempt to expose.

Chapter 2 reviews work already completed in the field of AM security and demonstrated attacks on 3D printers. This chapter will also provide a background on the technology and computer controls utilized by 3D printers which provide the opening for the demonstrated attack.

Chapter 3 covers the technical aspects of performing the attack as well as the experimental setup to verify a successful attack.

Chapter 4 discusses the results of the performed experiment and analyzes the degree of success a malicious actor in the real world may have with a similar attack.

Chapter 5 concludes the thesis and provides suggestions for further work.

CHAPTER 2

LITERATURE REVIEW

As additive manufacturing technology improves, there are increased opportunities for distributed manufacturing applications. Information security is a primary concern in a distributed manufacturing environment. Section 2.1 will provide an overview of the state of distributed manufacturing as it relates to 3D printing. Section 2.2 will highlight some known threats to information security and proposed solutions. Section 2.3 gives a background on underlying 3D printing technologies. Section 2.4 covers the fundamentals of G-Code. Section 2.5 discusses stepper motors and section 2.6 includes information on electromagnetic principles which will provide a basis for understanding the attack proposed by this thesis.

2.1 Distributed Manufacturing

The scope of distributed manufacturing can be understood through an analysis of possible industries and applications. Srai et al. [11] collected a panel of experts to discuss implications and applications for distributed manufacturing, then compiled and organized the results. The study proposes five key characteristics of distributed manufacturing: digitalization, personalization, localization, new enabling technologies, and enhanced user and producer participation[11]. They then apply these characteristics to specific case studies such as 3D printing, health care, and consumer goods. Srai et al. identify one of the key advantages of distributed manufacturing as production when needed and production closer to the point of consumption. This will ultimately reduce warehousing and transportation costs. The study highlights several advantages of distributed manufacturing, but acknowledges, “glaring IP implications in terms of ownership, necessitating a framework for IP sharing. IP protection will be necessary for the prevention of copyright infringement for design and development work.”

IP protection only becomes a concern when a manufacturer is independent of the intellectual property holder. Durão et al. [12] demonstrate such a distributed manufacturing environment within a laboratory setting. In the study a real part (a section of a pneumatic cylinder) was manufactured in Brazil upon authorization by computers in Germany. The group in Germany can be considered the “owner” of the IP and the part designer, while the

manufacturer in Brazil can be considered an independent manufacturing firm close to the end user. These two entities will henceforth be referred to as the designer and the manufacturer. The study demonstrates four use cases of varying levels of control. They range from nearly all manufacturing control happening in Brazil on the manufacturer's end, to nearly all control performed and monitored in Germany on the designer's end. This study was performed with relatively inexpensive equipment and open source software. It provides a comprehensive list of information that must be tracked and recorded by the designer, such as machine temperature, machine motion, and visual feedback. Successful implementation of a simulated distributed manufacturing environment is the first step to real world implementation.

2.2 Distributed Manufacturing Security

A distributed manufacturing system will require a large network of computers sharing large amounts of data. As with any such network, cybersecurity is a major concern. Manufacturing data such as CAD models, G-Code, and machine parameters, as well as customer information, payments, and product lifecycle management (PLM) data may be transmitted. Significant amounts of research have been performed on cybersecurity concerns for additive manufacturing. Yampolskiy et al. [13] provide a comprehensive survey of AM security research and proposes a taxonomy for AM security threats. The paper focuses on the threats of AM sabotage and the theft of technical data. AM sabotage is typically understood as a malicious actor attacking an AM system such that the output of the AM system is improperly produced. Sturm et al. [14] study cyber-attacks on an AM system producing voids, protrusions, or other physical imperfections into printed parts and analyzing the effect of these imperfections on part strength. This is just one of many analyses of AM sabotage. Theft of technical data includes the unauthorized access to 3D model data necessary to produce counterfeit parts. Yampolskiy et al. acknowledge "eavesdropping and side-channel analysis" as a method for reconstructing a 3D model and cites some examples which will be addressed further on.

Theft of technical data from an AM workflow has been demonstrated by attacking the 3D printers themselves to access the data transmitted to them. Miller et al. [15] surveyed the available landscape of desktop 3D printers to understand "residual data" left from 3D printing processes. This involves an analysis of how data is transmitted to 3D printers, and potential avenues for accessing printing data.

Do et al. [16] demonstrate in detail an attack on a desktop 3D printer using relatively inexpensive tools. In this study, an attacker was able to steal printing data and trigger the printer to make incorrect prints if it could access the same wireless network as the printer. An AM system or workflow which encrypts print data would be less susceptible to this type of attack.

Nevertheless, these studies demonstrate the need for a secure additive manufacturing system to prevent print data and intellectual property from being accessed by unauthorized users. The two most apparent approaches to a secure AM system are to secure the system itself and to secure the parts once they leave the system. Systems can be secured by what is often referred to as "smart contracts" where all actions within the AM process workflow, including design, payments, manufacture, and delivery, are securely traced. Parts may be secured by "watermarking" where a unique identifier for each individual part is created and applied to the part without interfering with the part's function. These two approaches will be discussed below.

2.2.1 Smart Contracts and Blockchain

One proposed solution to additive manufacturing security issues is to place the entire AM workflow into a secure system. Because security across borders is a concern, a system could not be fully secured by storing data on a server in a single country. A decentralized security platform on a blockchain network has been proposed and studied. A blockchain is a system in which messages, transactions, and other data are recorded on an unalterable ledger. The ledger is written and assembled by computers around the world. Individual "blocks" containing data are created and added to previous blocks creating a "chain" which holds a record of all previous transactions. A significant amount of computing power is required to create a block and created blocks must be verified by other computers [17]. The collective consensus of large numbers of computers prevents a single entity from creating or altering blocks. Therefore, the ledger is effectively unchangeable and presents a secure way to record AM workflow information.

An additive manufacturing workflow secured by the blockchain has been demonstrated in laboratory settings. Holland et al. [10] have developed a "Secure Additive Manufacturing Platform" or SAMPL which proposes a model for a distributed manufacturing system involving several parties. The parties are labeled as the customer, the printing service provider, and the licenser. Each does not trust the other two. In this model, the customer purchases a license from the licenser (IP owner) and the transaction is recorded in the blockchain. The customer can then

sell the license to a printing service provider of the customer's choosing. The printing service provider then has access to data necessary to print the device. Each step in the transaction is recorded on the blockchain, so that each participant can verify proper transactions. The licenser can verify the customer has purchased a license and has sold it to a printing service provider, the printing service provider can verify the customer has purchased a valid license, and the customer will receive a product with a paper trail of legitimate transactions and correct manufacturing history.

A similar model for secure outsourcing has been proposed by another paper by Yampolskiy et al. [18] but adds another participant to the 3D printing process chain. This participant is defined as a Manufacturing Process Tuning Expert which serves to consult with the original object designer to determine the ideal manufacturing parameters for a given design. The ideal parameters are then used in manufacturing by a separate 3D printing service. Yampolskiy highlights economic advantages for all parties involved. The participants are less bound to each other, allowing each to work with more cost effective or otherwise preferred parties. However, the work acknowledges that with more participants, IP protection becomes a greater challenge.

Yampolskiy then provides a risk assessment for the model, and names several areas which could present security vulnerabilities. The paper acknowledges side channel analysis of 3D printing equipment as a threat, particularly from a malicious 3D printing service provider. Yampolskiy does acknowledge that the 3D printing service would have unrestricted access to the equipment, which leaves it vulnerable to the type of side channel analysis being investigated by this thesis.

2.2.2 Watermarking and Fingerprinting

One of the core security questions arising from distributed manufacturing is how to prevent the unauthorized production of parts. As discussed with respect to smart contracts, customers may purchase licenses which allow production of a specific number of parts. A valid license is needed to print each object and every printed part can be accounted for. If 3D model and other print data is properly protected and only decrypted at the point of printing, unauthorized copies cannot be made. However, if a malicious manufacturer is able to find a flaw in the system and acquire the data necessary for printing, the printer may be able to produce pirated copies of parts. These could be sold at a price lower than a legitimate part, hurting the IP

owner. They could also contain flaws or otherwise not meet requirements, hurting the customer. One proposed solution is the use of watermarks or fingerprints in individual parts. Every part would contain a unique identifier which could be scanned or read to show information such as the manufacturer, original IP owner, or print license. It could also identify all product lifecycle information so users could know exactly how long and where a part had been used. These fingerprints must be legible, must last the life of the part, must not interfere with part performance, and must not be easily reproduced. Extensive research has been done on this topic and several fingerprinting systems have been proposed.

A study by Chen et al. [19] demonstrates a fingerprinting strategy of embedding quick response (QR) codes into 3D printed parts. A QR code is a two dimensional tag consisting of light and dark squares which can be scanned by a camera. In this study a sample QR code was embedded by inserting small voids into a CAD file where the dark squares of the QR code would normally be. The voids were moved axially throughout the part at various depths. Therefore, the QR code can only be viewed by scanning the part along the axis the voids were moved. This essentially turns the two dimensional QR code into a 3D fingerprint. This fingerprint is inserted during the design stage, and relies on proper processing and printing to display a valid code. Specific parameters must be followed when slicing the part to reproduce the correct code in a print. Chen uses this as an advantage and embedded a second, invalid QR code in a different orientation. When default slicing parameters are used, the invalid QR code is displayed and a counterfeit part can be identified.

This proposal assumes incorrect slicing by a malicious user will mark a part as invalid. A valid part could be produced with proper slicing and proper machine code. The attack proposed in this thesis would read the motor controls given by proper machine code. A valid QR code may then be reproduced in two separate parts. One part would be valid, and one part would be counterfeit, but in a large enough industry, part multiples may be difficult to detect. However, this proposal does outline an effective way to tag parts without interfering with mechanical properties.

A few of the same authors propose in Gupta et al. [20] another method for protecting against counterfeiting in additive manufacturing. Gupta proposes the deliberate insertion of discontinuities, voids, or other imperfections into CAD models of parts. If the parts are processed and printed correctly, these imperfections will not appear or not interfere with the

performance of the part. But again, improper slicing will reveal them and potentially compromise the part. With this proposal, counterfeit parts simply will not function as intended, rather than requiring scans to identify invalid parts. However, the proposal suffers from the same issue as Chen's with respect to this thesis. If reverse engineering occurs during the print process, a correctly processed and printed part could potentially be copied and reproduced.

Another proposed solution from Peng, et al. [21] utilizes the inherent instability in the 3D printing process to validate parts. Peng identifies printing noise as slight, non-problematic errors in printed parts resulting from typical inconsistencies in the mechanical performance of the printer or properties of the printing material. This noise can be used advantageously to validate parts. To identify printer noise, a small two dimensional authentication mark in the shape of a quarter of a circle is printed on a part. This shape is then scanned by a microscope and measured. Printing noise creates slight errors in the shape, which can be seen when scanned. These errors are used as features in the generation of a unique QR code. The QR code is then applied to the part after manufacturing to identify the signature of the part. The QR code is merely useful in verifying parts, and is not necessary to validate correct parts. The original authentication mark will remain on the part.

This security scheme is highly resistant to counterfeiting. Peng suggests the printing noise is effectively random, so it cannot be recreated in a counterfeit part. Reverse engineering during the print process could almost certainly not reproduce the exact authentication mark. But as printers become more advanced, printing noise and variability in prints will likely decrease. Therefore, it will take more and more sophisticated scanning equipment to identify small errors in the print. This could be a drawback as an individual wishing to verify a part may not have the capability to authenticate a print. These considerations all depend on the level of security required by the manufacturer, printer, and customer.

2.2.3 Secure Printers

An essential piece in the secure additive manufacturing workflow is the security of a printer itself. Smart contracts and blockchains ensure a record of all the details of an AM part, but to rely on these records, every manufactured part must be accounted for. One approach is to secure manufacturing instructions such as G-Code and machine parameters. Manufacturing instructions can only be authorized and delivered when all other licensing requirements have

been met. Once they are met, the instructions are encrypted so they can be sent along unsecure channels, then decrypted by the secure printer. An architecture proposed by Shaabany et al. [9] outlines how such a system would work. Shaabany et al. describe a Technology Data Marketplace (TDMP) where a technology data provider (TDP) interfaces with a customer through the marketplace and a market manager. A customer purchases technology data from the TDP and it is encrypted and delivered by the TDMP. The customer's machine is equipped with a device known as a Trusted Platform Module (TPM) which verifies the customer and provides the cryptographic key to decrypt the technology data. The TPM is a security standard [22] developed to authorize communication between distributed systems using cryptographic keys.

Carrying this proposed system further, in another paper, Shaabany et al. [23] developed a secure device using a TPM. A secure fluid mixing machine was developed by integrating a TPM into a Raspberry Pi which controlled the device. The TPM served to identify the machine and allow for secure data transmission. The design of the hardware acts as another security layer by attempting to obfuscate the functions of the machine and prevent reverse engineering of product delivery. A similar security infrastructure could be applied to an additive manufacturing machine. In a 3D printing device, a TPM could secure manufacturing information including CAD, G-Code, and manufacturing parameters sent to the customer. Hardware could potentially be designed to further secure the machine as Shaabany did for the fluid mixing device. This thesis attempts to investigate how vulnerable the hardware is once manufacturing information has been decrypted.

2.2.4 Demonstrated Side Channel Analysis

Vulnerability of additive manufacturing machines has already been demonstrated through side channel attacks. Al Faruque et al. [24] demonstrated an acoustic side channel attack on an AM system. Al Faruque asserts that the stepper motors which control an FDM printer emit a characteristic frequency based on motion and direction. The electromagnetic characteristics of the stepper motors and their controllers produce an acoustic signature. The signature can be recorded, and through a series of analysis algorithms, the motion of the motor can be deduced. The motion of the motor is then translated back into machine code. Al Faruque was able to track a motor with an average axis prediction accuracy of 66.29%. This is nowhere near the accuracy necessary to reproduce a high precision part, but it serves as a very impressive benchmark for an

attack using only acoustic information. A similar demonstration by Kubiak et al. [25] used high precision recording equipment and sound isolation techniques to measure the audio signature of stepper motors for a number of distinct motor motions. The acoustic signals can be analyzed to determine the motion of the motors.

This thesis will investigate an attack similar to these types of side channel analysis, but using electromagnetic fields rather than acoustic data. A demonstration by Gatlin et al. [26] measured electromagnetic emissions from a 3D printer, but used the information to secure the printer rather than demonstrate an attack. The expected electromagnetic pattern can be compared to the measured electromagnetic pattern to determine if the printer executed the print correctly. This technique could be used to prevent attack like that demonstrated in Sturm et al. This thesis proposes an attack to steal data using a similar approach. A key difference between these types of attacks is an acoustic attack could be performed by any malicious actor who could place an audio recording device within range of the AM device. Analysis of electromagnetic fields requires closer access to the AM device and would be more likely performed by a malicious machine owner wishing to steal confidential IP.

2.3 Additive Manufacturing

The technology known as additive manufacturing or 3D printing encompasses a wide array of distinct techniques for manufacturing parts. Common among these technologies is that every object is built by adding successive thin layers of material until the final shape is completed. In the standard additive manufacturing workflow, an object is first created in 3D modeling software. The computer model is then "sliced" into several thin horizontal layers. The AM machine is programmed to create each successive layer. The layers are formed, and the physical object is created. Several approaches exist for the deposition of these layers. The earliest AM technology, known as stereolithography, uses a directed laser to harden the surface of a vat of liquid photopolymer. The hardened pattern is lowered into the liquid and another layer is added. Successive layers are hardened and lowered until the desired shape is created. Another technique, classified as discrete particle manufacturing, [27] joins small, disconnected particles together to create the shape of each layer. In some cases, a directed laser can be used to melt small plastic particles together. In others, the particles are joined by selective application of a binding material. This technique allows plastics as well as some metals to be additively

manufactured. One of the most common AM technologies is fused deposition modeling (FDM) where a thermoplastic filament is melted, then deposited through a nozzle onto a build surface. The layers of deposited plastic correspond to the layers created during the slicing of the original CAD model.

A typical FDM printer consists of a build surface, stepper motors, and a heated nozzle. The stepper motors can move both the build surface and the nozzle according to machine instructions created during the AM workflow. These machine instructions are generated in a standardized form known as G-Code. Most slicing software products also function to generate G-Code instructions. The fundamentals of G-Code will be explained further in the next section. As the FDM printer moves through the machine instructions it either deposits material along a prescribed path or moves the tool to a new location without depositing. The stepper motors control every movement of the printer. Typically, a single stepper motor controls motion in the X-direction, and another controls motion in the Y-direction. These two motors work in tandem to create the shape of each layer. A motor or motors moving the build surface or the tool in the Z-direction are only used to create the next layer of material. Another motor is used to control the extrusion of the melted material.

All the information necessary to create a 3D object via additive manufacturing must be transmitted to these stepper motors. They must be able to move freely, and the build surface must be readily accessible to retrieve completed parts. Therefore, anyone who has access to an FDM machine has access to these motors. If the 3D object or the machine instructions are proprietary, motor access could present a potential security flaw.

2.4 G Code

Before an additive manufacturing machine can create an object, it must be given instructions on how to do so. After a 3D model is sliced and the topology of each layer is defined, a toolpath is generated to tell a machine how to create that topology. A standardized method for delivering machine instructions known officially as the NIST RS274GC Interpreter [28] is more commonly referred to as G-Code. The standard was originally developed for numerically controlled traditional manufacturing operations such as milling, but was later applied to additive manufacturing systems. An example of a piece of G-Code can be seen below.

```
G1 F1800 X44.933 Y47.831 E0.02945
```

```
G1 X45.674 Y47.203 E0.07791
G1 X46.301 Y46.723 E0.11731
G1 X46.974 Y46.31 E0.1567
G1 X47.521 Y46.038 E0.18718
G1 X48.407 Y45.64 E0.23564
G1 X49.143 Y45.353 E0.27505
G1 X49.904 Y45.142 E0.31445
G1 X50.47 Y45.038 E0.34316
G1 X51.444 Y44.895 E0.39227
G1 X52.577 Y44.812 E0.44895
```

G-Code instructions for printing a single object follow a typical format:

1. Machine setup
2. Toolpath for depositing each layer
3. Machine shutdown

The NIST standard includes over one hundred specific machine instructions to encompass the entirety of computer numeric control (CNC) machine technology. Commands like spindle speeds, coolant application, drilling, and tapping rarely apply to AM. Machine setup instructions like build plate or extrusion temperature are necessary, but the most commonly used commands in AM applications are G0 and G1. These commands tell the extruder or other processing tool where to move. Each command provides an X coordinate and a Y coordinate. When a machine executes these commands, it merely moves the tool on a direct line to the given coordinates. G0 commands indicate where the tool should move without depositing any material. They are necessary for repositioning to start a new layer or discontinuous area of a layer in process. G1 commands include an instruction for how much material should be extruded over the course of the movement of the tool. G1 commands must also be given a feed rate to define how fast the tool should move, however this rate must only be given at the start of a sequence of G1 commands, or when the feed rate needs to be changed.

Before slicing, 3D objects are represented in a file format called STL, where the shape of the object is approximated by tiny tessellations. True curves are effectively eliminated as they become represented by thousands of small line segments. Therefore, the tool of an AM machine is only required to travel in straight lines. G-Code commands (G2 and G3) for rounded edges or arcs are not necessary in an additive manufacturing environment.

Even a small object can require thousands of lines of G-Code commands, but contained within these commands is the entire shape of an object. Anyone with access to these standard machine instructions can replicate the object. A secure AM platform must protect machine commands as securely as the 3D CAD file.

2.5 Stepper Motors

Most standard fused deposition modeling AM machines use stepper motors to move the machine along the specified tool path. Stepper motors are designed to rotate an output shaft a specific amount or "step". These motors are highly precise and common inexpensive motors are capable of reliably turning a shaft in increments 1.8 degrees or two hundred steps in one revolution. This precision is necessary for additive manufacturing as tools must move along tight tolerances to create fine details of 3D objects. Stepper motors can be attached to belts, gears, lead screws, or any other variety of power transmission devices to move the AM tool as programmed.

The precise control of a stepper motor is created by selectively charging stationary coils of wire inside the motor. These charged coils create a magnetic field which changes the position of a freely rotating permanent magnet. This permanent magnet is attached to the shaft of the motor, so as the magnet moves, the shaft moves. The coils are charged in a prescribed sequential pattern which causes the shaft to rotate. Each step in the pattern of charging the coils creates a step in the motor. The specific orientation of these coils and magnets can vary among motor designs. A wide variety of designs exist for stepper motors and one of the most common is the bipolar stepper motor. A bipolar stepper motor consists of two coils and a permanent magnet attached to the motor shaft. As current is sent through one coil (coil A), a magnetic field is created and the permanent magnet aligns itself with the field. Current is then sent through the other coil (coil B), a field is generated, and the permanent magnet aligns itself, turning the shaft. Then, the direction of the current through coil A is reversed, reversing the polarity of the magnetic field and once again turning the motor shaft. The current direction in coil B is reversed, the motor steps, and the cycle starts over, with the current in coil A returning to its original direction. The design of the permanent magnet attached to the shaft allows for precise 1.8 degree turns each time the current is reversed through each coil.

The specific inner workings of the stepper motor are not relevant to AM process security, but the method of controlling it is. The inner coils of the motor are connected via external wires to the driving circuit controlling the motor. Every time the current in the wire is reversed and the motor makes a step, a "signal" is sent along the wire. If this signal can be read and timed, the

motion of the stepper motor can be known. The precision and simplicity of the stepper motor allows for such signal interpretation.

2.6 Electromagnetics

A foundation of the electromagnetic principles used to achieve the demonstrated attack is established in the following sections. These fundamental properties of electromagnetism must be understood to read the stepper motor control signals.

2.6.1 Magnetic Fields

Any current or moving charge produces a magnetic field around the current conductor. This magnetic field propagates outside the current carrying wire, so no direct electrical connection is necessary to detect the presence of current. Several basic electrical components such as transformers, motors, and inductors function based on this property. Other devices such as current clamps utilize the property to measure the current flowing through a wire. The magnetic field serves as a window into the current in the wire and the sequence of signals being sent to the stepper motor.

The intensity of this magnetic field depends on the magnitude of current as well as the shape, direction, and proximity to the current. It can be determined with two fundamental laws of electromagnetism: Biot Savart's Law and Ampere's Law [29].

Biot Savart's Law governs how a moving charge produces a magnetic field and is defined by the following equation:

$$d\boldsymbol{H} = \frac{I\ d\boldsymbol{l}\ x\ \boldsymbol{R}}{4\pi R^3}$$

Where d**H** is a differential element of the magnetic field, I is current, d**l** is a differential element of the vector along the current path, and **R** is the vector from d**l** to d**H.** Therefore, the magnetic field generated by a line current is given by:

$$\boldsymbol{H} = \int_L \frac{I\ d\boldsymbol{l}\ x\ \boldsymbol{a}_R}{4\pi R^2}$$

Where $\mathbf{a}_R$ is a unit vector from d**l** to the point of interest. The units of **H** are Ampere/meter. Ampere's Law broadly states that the line integral of **H** around a closed path is the same as the net current I_{enc} enclosed by the path. Ampere's Law is a special case of Biot-Savart's Law. It is broadly defined by the following equation:

$$\oint \boldsymbol{H} \cdot d\boldsymbol{l} = I_{enc}$$

It is more usefully defined for an infinite line current by:

$$\boldsymbol{H} = \frac{I}{2\pi\rho}\boldsymbol{a}_{\Phi}$$

Where $\mathbf{a}_\Phi$ is a unit vector along the "Amperian path", which in this case is concentric circles surrounding the infinite line current.

Finally, magnetic flux density, **B**, is given by:

$$\boldsymbol{B} = \mu_o \boldsymbol{H}$$

Here μ_o is a constant known as the permittivity of free space with a value of $\mu_o = 4\pi \times 10^{7}$ H/m. Magnetic flux density defines how magnetic forces act and is more useful for specific applications necessary for current detection. The units of **B** are Tesla (T) which equals kg/s^2A in SI base units.

2.6.2 Inductance and Capacitance

Two passive circuit elements, inductors and capacitors, can alter the current and voltage in a circuit over time affecting the functionality of components like stepper motors and current sensors. In its most basic form, an inductor is a coil of wire. As current flows through the wire, a magnetic field is generated according to the principles previously described. This induced magnetic field influences the current flow through the wire. A capacitor is an element which collects and discharges electrical charge. It also changes the flow of current through a circuit. In conjunction with a resistor, these circuits, known as RL (Resistor-Inductor) and RC (Resistor-Capacitor) circuits, can be used to change the behavior of a larger circuit. They can be used in signal filtering, amplification, and a wide variety of other applications. The coils of wire in a stepper motor behave as an inductor.

The time response of these circuits is defined by a "time constant", or how quickly an RL or RC circuit builds or decays. The time constants are defined by the following equations and correspond to how long it takes for a circuit to increase to 63.2% of its final value [30]. Units for a time constant are in seconds.

$$RC\ Circuit: T_c = RC$$

$$RL\ Circuit: T_c = L/R$$

These circuit elements will be useful in reading the stepper motor signals. The other foundational electromagnetic principles provide a starting point for the demonstrated attack. They can be applied to any motor or system vulnerable to an attack.

CHAPTER 3

EXPERIMENTAL SETUP

Chapter 3 will demonstrate the setup of the attack through the following sections. Section 3.1 introduces the general simulation. Section 3.2 covers the simulation of the 3D printer. Section 3.3 describes the attacking device. Section 3.4 outlines the technique for interpreting voltage data.

3.1 Simulation and Demonstration of an Attack

The focus of this thesis is to determine the possibility of an attack on a secure additive manufacturing machine by detecting and measuring control signals sent to the stepper motors of the machine. The possibility will be demonstrated through a simulated attack on an additive manufacturing type system. If this simulated attack is successful it will provide a proof of concept for attacks on more secure machines which are closer to real production systems. This will allow designers and manufacturers to find vulnerabilities and openings within their secure system. They can then find technical solutions to close these gaps and improve security.

The demonstration will consist of two systems: the simulated AM machine and the attacking device. The simulated AM machine will be able to take standard G-Code instructions and control stepper motors accordingly. In a successful attack, the attacking device will be able to read the current passing through the stepper motor wires and translate the current signals back into G-Code. This experiment will only demonstrate G-Code recreation on one motor. However, because G-Code instructions operate in two dimensions independently, translation to two dimensions would not require significant changes.

3.2 3D Printer Simulation

Standard state of the art 3D printers use several stepper motors to translate a build area in the X, Y, and Z directions. Stepper motors are also used to extrude the build material. Rather than assemble an entire 3D printer, this demonstration will replicate just the Y-motor of a 3D printer. The Y-motor controls motion of the build area in the Y direction. The X and Y motors of a 3D printer typically work in conjunction. A single line or block of G-Code typically contains instructions for motion in both X and Y. It can contain instructions in the Z direction,

but standard 3D printers only move in the Z direction when adding a new layer. Complex motion in the Z direction is possible and is an area of study within additive manufacturing, but is not a focus of this simulation. The Y motor was chosen for analysis to leave the X motor available as a control for analysis of the simulated AM machine.

The simulated AM machine consists of four main components: the controlling computer, the motor drivers, motors, and an external 24V power supply. There is no set standard for specific components of a 3D printer. Different components may require slightly different approaches or techniques to attack, but the components selected in the demonstration are common components similar to those found in a wide array of AM systems.

3.2.1 3D Printer Control

The central controller of the 3D printer is a Raspberry Pi 3 Model B V1.2. The Raspberry Pi is a small, single board computer capable of performing a wide variety of computational tasks. It costs less than $50 and is easily found in several online electronics stores. It is an easily configurable, low cost solution which provides external power, general purpose input-output (GPIO) pins, a user interface, programming capabilities and a large open source user community. It can interface with monitors, keyboards, mice, and any other standard computer peripherals and can control computer numeric control machines and programs.

Attached to the Raspberry Pi is an add-on board called the Raspberry Pi CNC board version 2.58 from Protoneer. The Protoneer CNC board runs an on board CNC microcontroller running GRBL [31], an open-source G-Code and CNC control program. The Protoneer board includes sockets for four separate stepper motor drivers and screw terminals for each stepper motor. Screw terminals for power, spindle control, and other CNC features like end stops are also included. A CNC control board functions in place of a standard off-the-shelf 3D printer board. Functionally, these devices work identically for reading G-Code instructions and turning motors as commanded. This board does not include stepper drivers which must be purchased additionally. The CNC board gives more control and a better interface than would be found in an off-the-shelf 3D printer. The CNC board will allow testing of specific motor movements and single lines of G-Code. This will help verify the success of the attacking device.

The interface of the CNC control is through the standard Raspberry Pi desktop. Protoneer supplies a user interface SD Card image V4.10 [32] which includes bCNC [33], a user

interface for controlling CNC operations which can be seen below. BCNC can receive and send single G-Code commands and can also be used to create and execute entire blocks of G-Code.

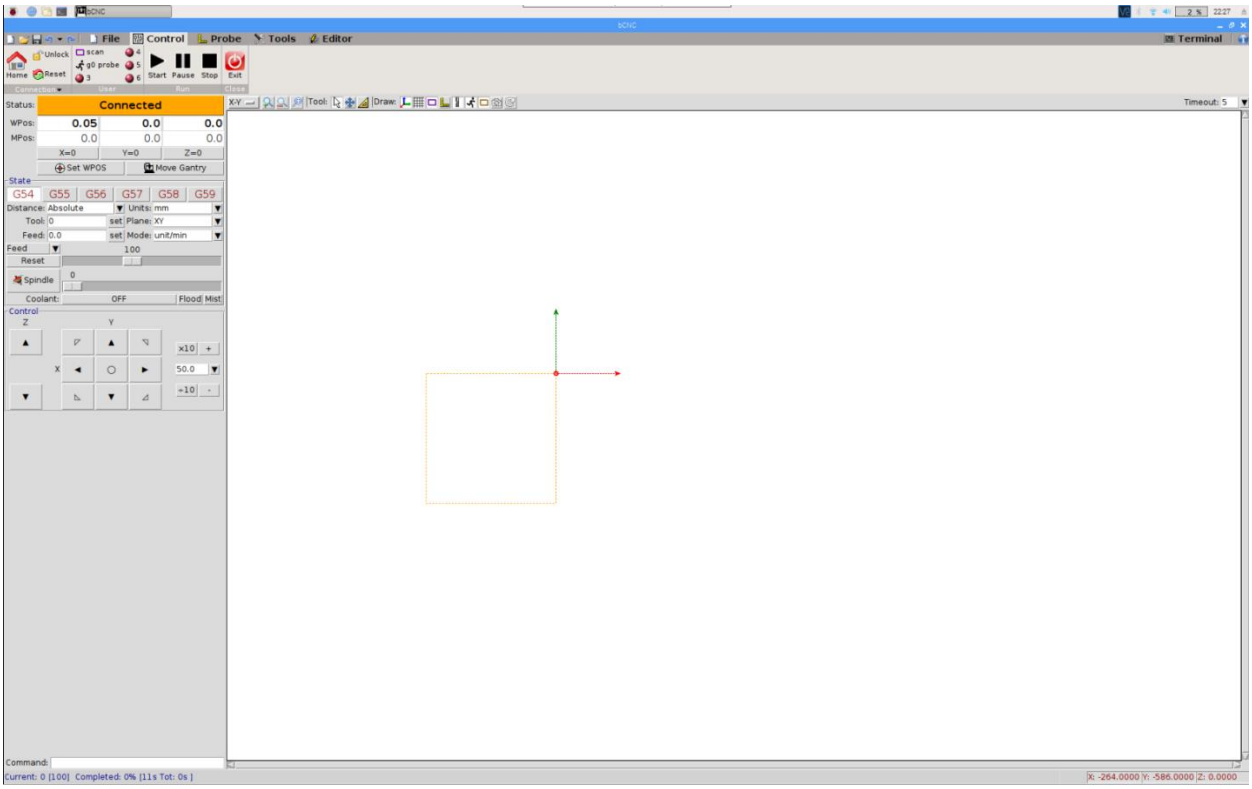


Figure 3.1: bCNC user interface for sending G-Code commands to stepper motors

3.2.2 Motors and Drivers

The motors used in this demonstration are model 42HB34F08AB stepper motors made by Changzhou Bo Hong Electric Appliance Co. They are bipolar stepper motors with two coils, and four wire leads, the same type of stepper motors found in a typical 3D printer. The motors travel a standard 200 steps per revolution, or 1.8 degrees per step. They have a phase resistance of 6.2 Ω, and a phase inductance of 10 mH.

The motors are driven by external stepper drivers mounted to the Protoneer controller. The drivers are Pololu DRV 8825 high current stepper motor drivers. These drivers are known as “chopping drivers”. The most basic motor control is to send current through the motor coils, charging each coil in the pattern necessary to achieve the desired speed and direction. The alternating charges are typically visualized as a square wave. Rather than producing these full alternating charges, a chopping driver splits each charge period into several much smaller spikes in voltage. These particular drivers chop at a frequency of 30 kHz. A visualization of the

chopping signal can be seen below. Figure 3.2 shows the chopping signal at slow speed. Figure 3.3 shows the chopping signal at a higher motor speed. The lower voltage in yellow is the signal delivered to the motor. One can see the characteristic spikes in voltage produced by the chopping driver. The upper purple signal is the output of the current sensor filtered through the RC circuit.

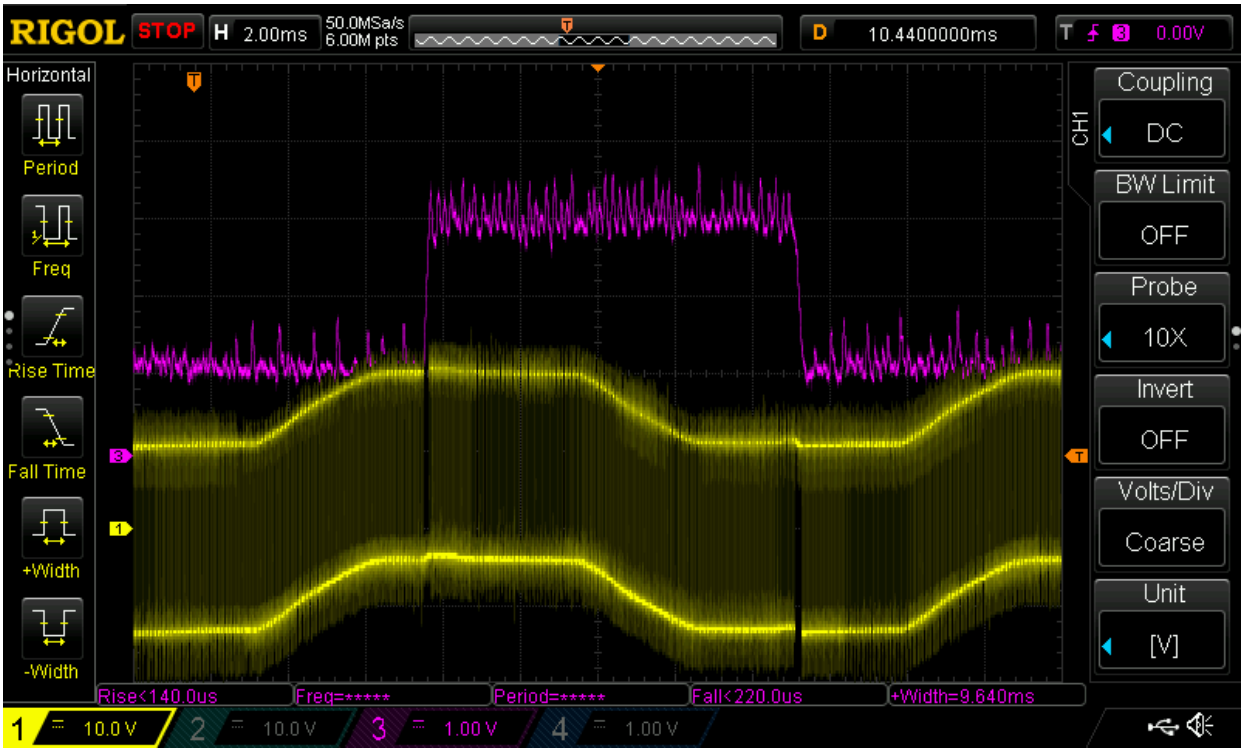


Figure 3.2: Comparison of stepper motor signal and current sensor signal at low speed

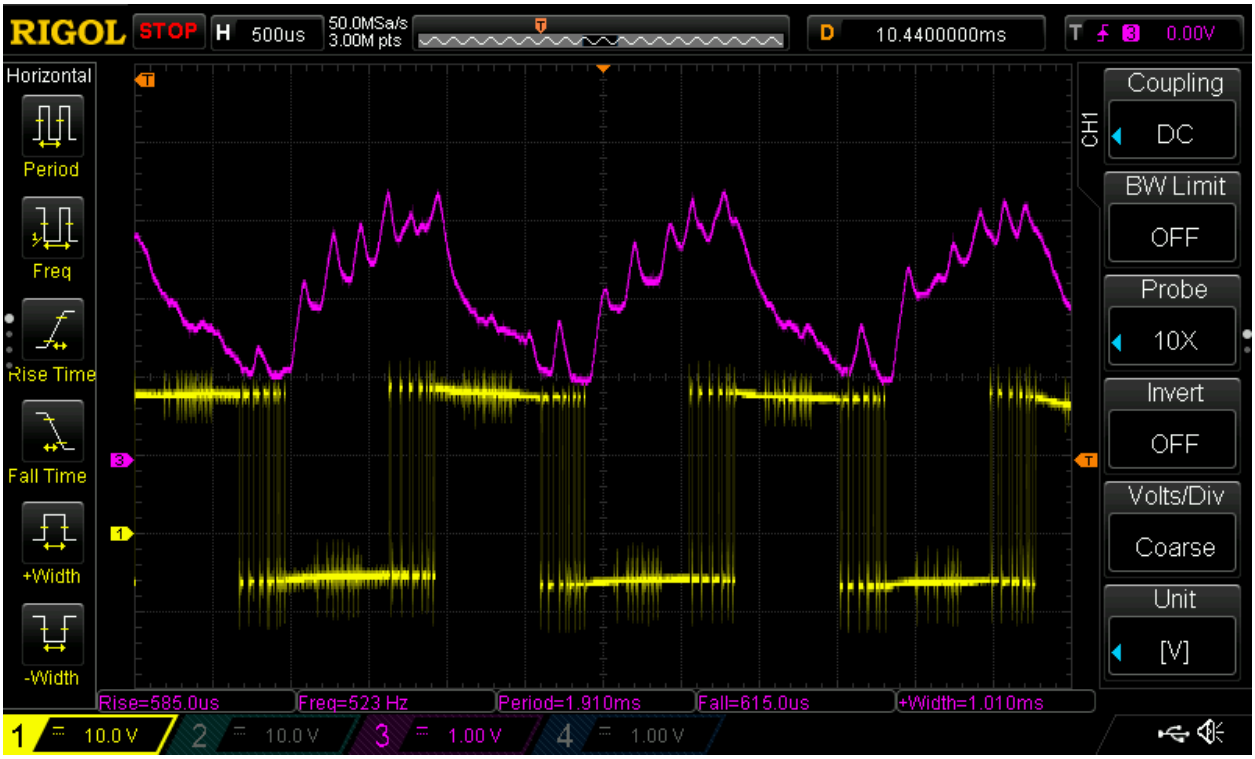


Figure 3.3: Comparison of stepper motor signal and current sensor signal at high speed

The chopping signal allows a high voltage to be applied to the motor coils while keeping the induced current in the coils low. The high voltage is necessary for faster steps and higher torque, while the current must be kept low to prevent damage to the motor. The chopping technique relies on the inductance of the motor coils to resist the frequent change in current. With the motor inductance, the voltage spikes effectively become one continuous signal, similar to a basic motor control pattern.

The stepper drivers can be adjusted to allow a maximum amount of current to pass through the coils of the stepper motor. The stepper motors have a current rating of 0.8 Amperes, so the drivers are adjusted to only deliver this maximum current. The current is further limited by the driver. In full step mode, the current windings are only set to receive 71% of the limited current. This is strictly due to the construction of the driver and cannot be adjusted. Therefore, the current in the motor coils and wires is 0.57 A alternating at a frequency of 30 kHz.

The drivers, the motors, and the wires which connect them are the key pieces in reverse engineering the stepper motor control signals. This thesis assumes that a signal may be securely transmitted up until the control computer of the 3D printer, here represented by the

Protoneer+Raspberry Pi system. Everything beyond the control computer is the target of an attack. The motors and the attached wires must be free to move in an AM system. They cannot be deeply hidden within the machine or easily obscured. That leaves them vulnerable.

3.3 Attacking Device

The demonstrated attack will be carried out by an external device capable of reading control signals sent to the stepper motors, recording them, then externally processing them to translate the recorded signals back into G-Code. The control signals are read by current sensors detecting current passing through the stepper motor wires. The current sensor data is filtered through a passive circuit, then read by an analog to digital converter (ADC). The ADC is connected to a Raspberry Pi, which records and saves the digital signals.

3.3.1 Current Sensors

As explained in the previous chapter, the current passing through the stepper motor wires creates a magnetic field around the wire. The presence of a magnetic field indicates current, and the lack of a magnetic field indicates no current. The magnitude of the magnetic field corresponds to the magnitude of the current. These characteristics are the key to reading the stepper motor control signals without interrupting the stepper motor circuit.

The current sensors used in this application are ACS723 High Accuracy, Galvanically Isolated Current Sensor ICs integrated into a chip called the Sparkfun Current Sensor Breakout (Low Current) produced by Sparkfun. The ACS723 is a Hall Effect sensor, a very common device used to detect the magnitude of a magnetic field. This particular sensor is designed specifically as a current sensor, utilizing the known relationship between current and magnetic field. The sensor has a listed sensitivity of 400 mV/A, meaning for every ampere passing through the sensor, the output signal will increase by 400 mV. Therefore, for 0.57 A, the current sensor will output approximately 0.23 V above the reference voltage. The reference voltage at zero current is listed as half of the supply voltage of the sensor. Supply voltage is 5V, so the reference voltage is 2.5V. However, the sensor IC is integrated into a breakout chip with more advanced features. Two integrated potentiometers can adjust the reference voltage and the sensitivity of the output.

The current sensor must be directly integrated into the circuit of the stepper motor signal. Realistically, this means cutting wires or removing them from the screw terminals of the Protoneer controller. While a malicious machine operator with unlimited access to a machine could potentially do this, it may not be possible in a true attack scenario. However, there is still electrical isolation between current being sensed (the stepper motor wires) and the current sensor power supply and output signal. This setup still proves the concept of an external sensor detecting the presence of a magnetic field to measure current. Expensive, high precision sensors such as current clamps are capable of measuring current without cutting the wires. Construction of such a sensor may be an avenue for future research.

3.3.2 Raspberry Pi and AD Converter

The main computer for powering the current sensors and reading and storing the data is a second Raspberry Pi with an integrated analog to digital converter. It runs on the default Raspbian image. Programming is possible in multiple languages. For this specific application, the control code was written in C++ to handle the speed required to rapidly read voltage signals from the current sensors.

The analog to digital converter used is a Raspberry Pi High-Precision AD/DA Expansion Board from Waveshare. It serves to translate the analog output of the current sensors to digital signals readable by the Raspberry Pi. The expansion board directly mounts to the pins of the Raspberry Pi and provides external pins and screw terminals for sensor integration. It has eight separate channels for analog input, an output voltage of 5V and a ground terminal. The current sensors can directly connect to the analog input of the board and can run off the supplied 5V. The board primarily serves as a usable interface for the mounted ADS1256 analog to digital converter integrated circuit from Texas Instruments. The ADS1256 is an eight channel delta sigma analog to digital converter which can output 24 bits of data at speeds up to 30,000 samples per second. The ADS1256 cannot read all eight channels at 30,000 samples per second due to time delays when cycling through channels. When reading one channel, the sampling rate can be increased as the converter does not need to use the time required to switch between channels. The high precision and speed of this ADC are necessary to read the rapidly changing signals of the current sensors and deliver that data to the Raspberry Pi.

The code used to run the ADC through the Raspberry Pi was modified code produced by Waveshare [34] to accompany the board. The default code starts up the board, then enters a function which instructs the ADC to cycle to a channel, issuing the necessary commands to retrieve one conversion, then returns that data to the interface. It then moves on to the next channel and does the same until all eight channels have been read. It then restarts. A new data retrieval function was written which issues the necessary channel commands to a single channel. The continuous data command is issued, and the code continuously outputs the converted data into a text file with a corresponding time stamp. The board is programmed to operate at the maximum 30,000 samples per second. However, as instructed by the datasheet of the ADS1256, a settling time of 5 "DRDY" periods is necessary for continuous conversion of data. One DRDY period is equal to the inverse of the programmed sampling rate, so the settling time required is approximately $5 * \left(\frac{1}{30{,}000}\right) = 1.67\mu s$. This programmed settling time significantly slows the sampling rate of the board to around 5,500 samples per second. However, this is still fast enough to read the stepper motor steps at full speed. The main execution file and new continuous data read command can be found in Appendix B.

3.3.3 RC Circuit

The current sensors are integrated directly into the wires of the stepper motor and detect every fluctuation in current through those wires. The chopping driver charges the stepper coils at 30 kHz which means current is sent through the current sensor at that rate. As previously stated, the inductive properties of the stepper motors smooth out this high frequency, but the output of the current sensor appears as a highly noisy signal. The current sensor signal is passed through a filtering RC circuit to remove spikes in the output voltage.

The resistor and capacitor values required for the RC circuit were determined through analysis and experimentation. As a starting point, the stepper motors were treated as an RL circuit with a time constant determined by $T_c = \frac{L}{R} = \frac{10\ mH}{6.2\ \Omega} = 1.6\ ms$. An ideal RC circuit will approximate the behavior of the RL circuit and the filtered output of the current sensor will appear as a traditional stepper motor control signal. The values found to be most effective for proper filtering were a resistor of R = 330 Ω and a capacitance of C = 1 µF for a time constant given by $T_c = RC = 0.33\ ms$. A smaller time constant ensures that the RC circuit used to filter

the current sensor signal will respond faster than the RL circuit of the stepper motor. Some of the noise from the chopping drivers is still present, but it is reduced to a level where it is no longer an issue for later data processing.

3.3.4 Integration

Combining these components into one system is the final step in developing the hardware for the hacking device. The thesis demonstrates an attack on one motor. One motor has two motor coils and each change in the current direction in a motor coil corresponds to one step. To count each step of the motor, a current sensor must be integrated with the circuit for each coil. This means two separate sensor signals must be read simultaneously. The ADC integrated with the Raspberry Pi cannot cycle between channels fast enough to read the sensor voltages for the motor at high speed, but it can read one channel fast enough. The solution is to combine the two motor signals into one, using a passive differential amplifier circuit. In a differential amplifier, the two signals are fed into the positive and negative inputs of an operational amplifier or "op amp" and all the inputs and outputs are connected via a specific resistor configuration. The resulting output is equal to the voltage on the positive input minus the voltage on the negative input. The configuration can be seen in Figure 3.4.

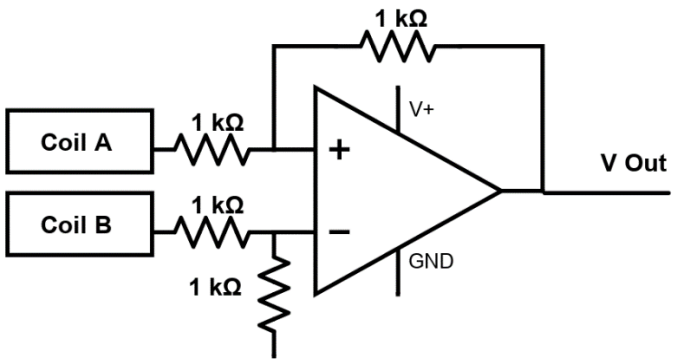


Figure 3.4: Differential amplifier circuit [35]

The voltage outputs of the current sensors must be adjusted, so that the difference between the two sensors displays four distinct levels. The integrated reference voltage and gain potentiometers of the current sensors allow easy adjustment and provide the desired output of the difference amplifier. The voltage data from Coil A is fed into the positive terminal of the op amp and has a reference voltage of approximately 2.8 V and a gain of 0.8 V when the coil is energized. The data from Coil B is fed into the negative terminal of the op amp and has a

reference voltage of 0.8 V and a gain of 0.4 V when the coil is energized. The input and output voltages are summarized in the following table. High and low indicate whether the voltage passing through the stepper coils is positive (high) or negative (low). The sequence of coil charging is controlled by the stepper motor driver. The "state" column refers to the state of the stepper motor. As the motor turns, it progresses through four distinct states depending on the voltage of the coils. The numbers for the state value have been chosen to match the progression of states defined by the stepper driver datasheet.

Table 3.1: Voltage Levels for Current Sensors and Differential Amplifier Output

State	Coil A		Coil B		Output
Off	Off	2.8 V	Off	0.8 V	2.0 V
1	High	3.6 V	High	1.2 V	2.4 V
2	Low	2.0 V	High	1.2 V	0.8 V
3	Low	2.0 V	Low	0.4 V	1.6 V
4	High	3.6 V	Low	0.4 V	3.2 V

The resulting data from the differential amplifier circuit can be read into the single channel of the analog to digital converter. The difference in voltage and gain gives four distinct voltage levels corresponding to the four states of the stepper motor. Each state corresponds to a single step, so each time a new state is reached, a step can be counted. It is important to note that these reference levels and gains are relatively arbitrary. All that matters is that the reference voltage and gains of the two sensors are different from each other so that the output of the op amp gives the four distinct voltage levels. A full diagram of the attacking device can be seen in Figure 3.5.

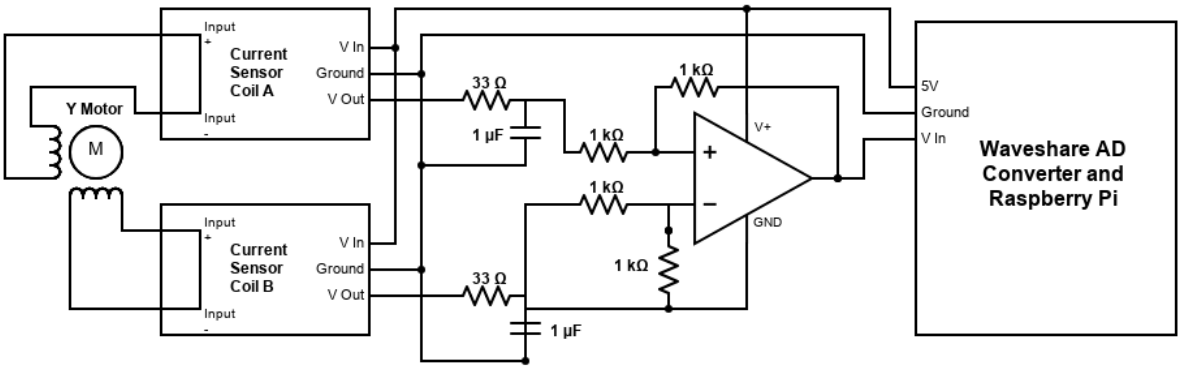


Figure 3.5: Full attacking device diagram [35]

The voltage output by the sensors and the voltage output by the op amp for a step sequence can be seen in Figure 3.6. When Coil A and B are high, state 1 is output. When Coil B stays high but Coil A goes low, the output enters state 2, etc.

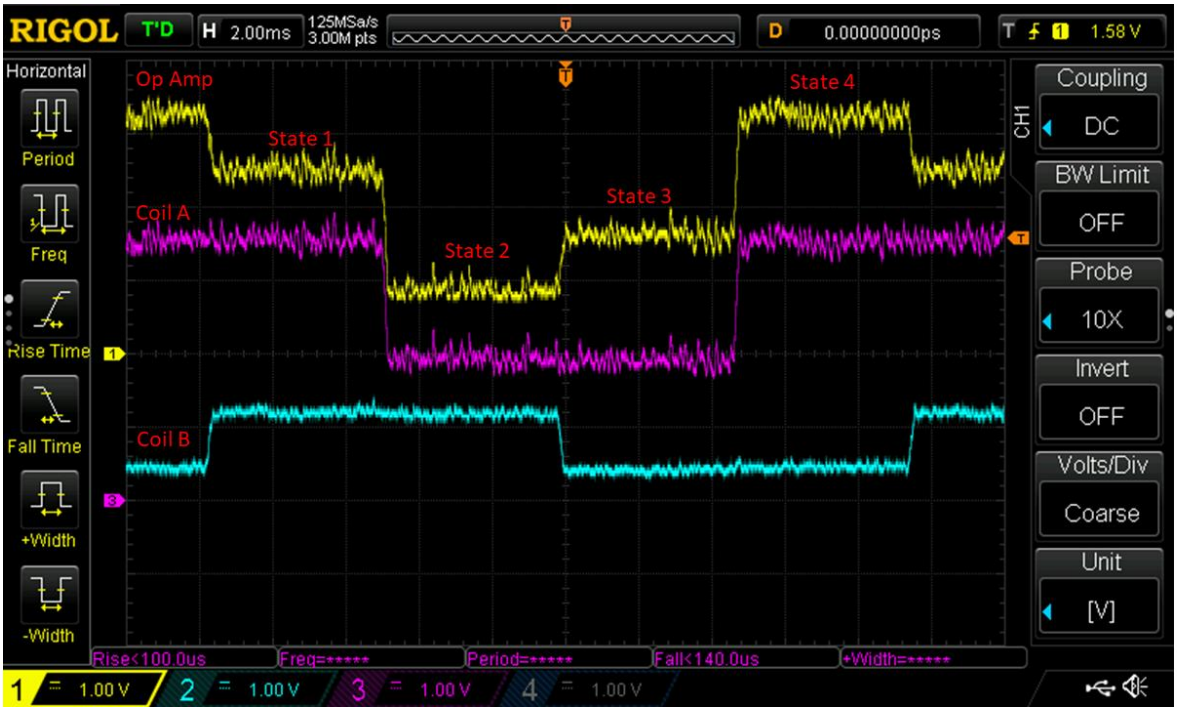


Figure 3.6: Current sensor outputs compared to output of differential amplifier

The physical layout of the entire setup can be seen in Figure 3.7.

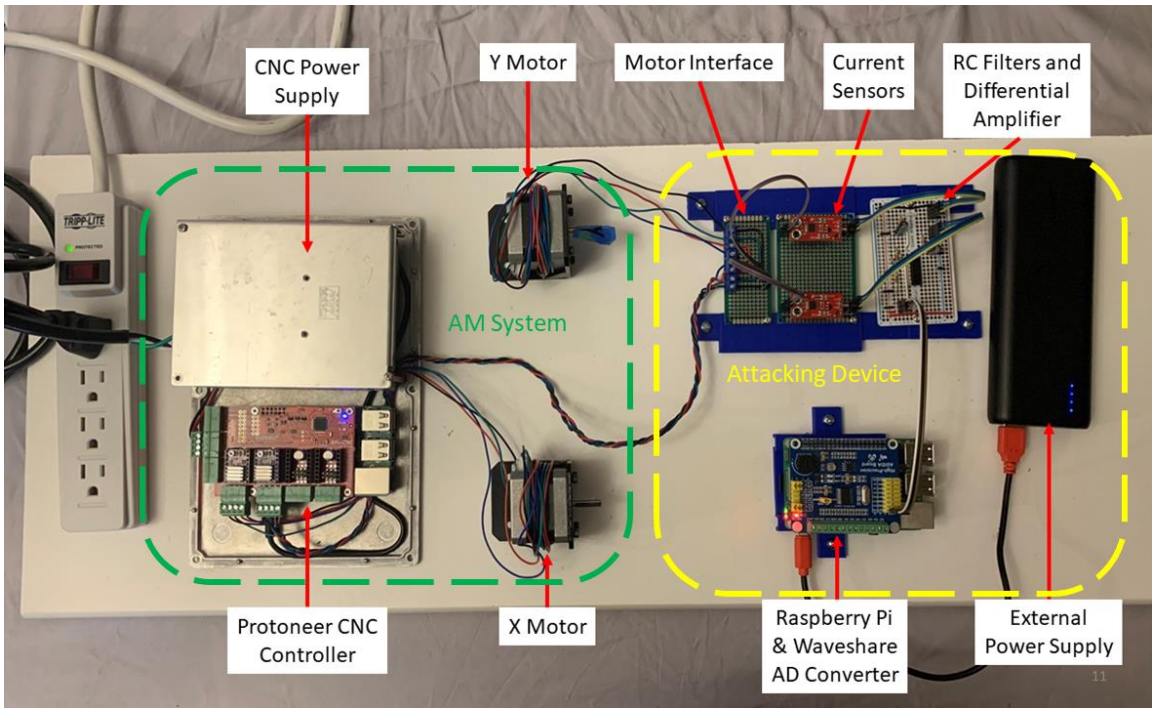


Figure 3.7: Full experimental setup

3.4 Data Interpretation

The final step in demonstrating the attack is to process the voltage data collected by the Raspberry Pi. The Raspberry Pi program outputs a column of voltage data into a text file alongside the time each data point is recorded. The clock starts when the program to record data begins, it does not correspond to global time. This voltage data represents the output of the differential amplifier as it is read by the ADC integrated with the Raspberry Pi. The data is saved to an external drive and moved to a PC where it is processed using Matlab. The data is processed in Matlab to use the increased computing capability of a PC and minimize the computing required by the Raspberry Pi so that it can run the ADC as fast as possible. A sample of the output code for a motor translation of 0.1 mm plotted in Excel can be seen in Figure 3.8. An equal motion in reverse is seen in Figure 3.9. The four voltage levels corresponding to motor states 1, 2, 3, and 4 as well as no motor motion can be seen.

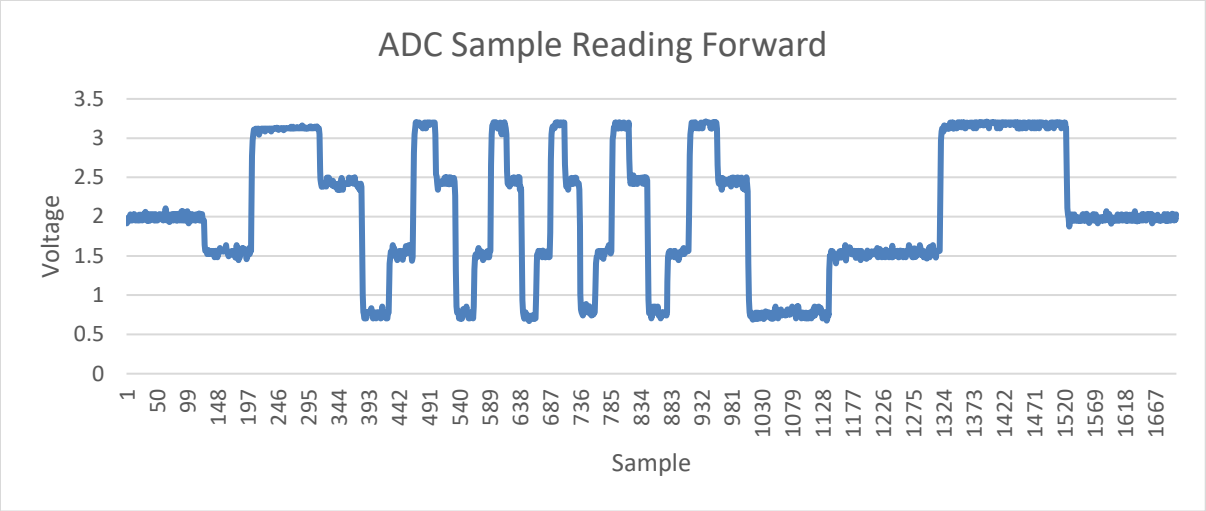


Figure 3.8: Sample reading of 0.1 mm forward motion

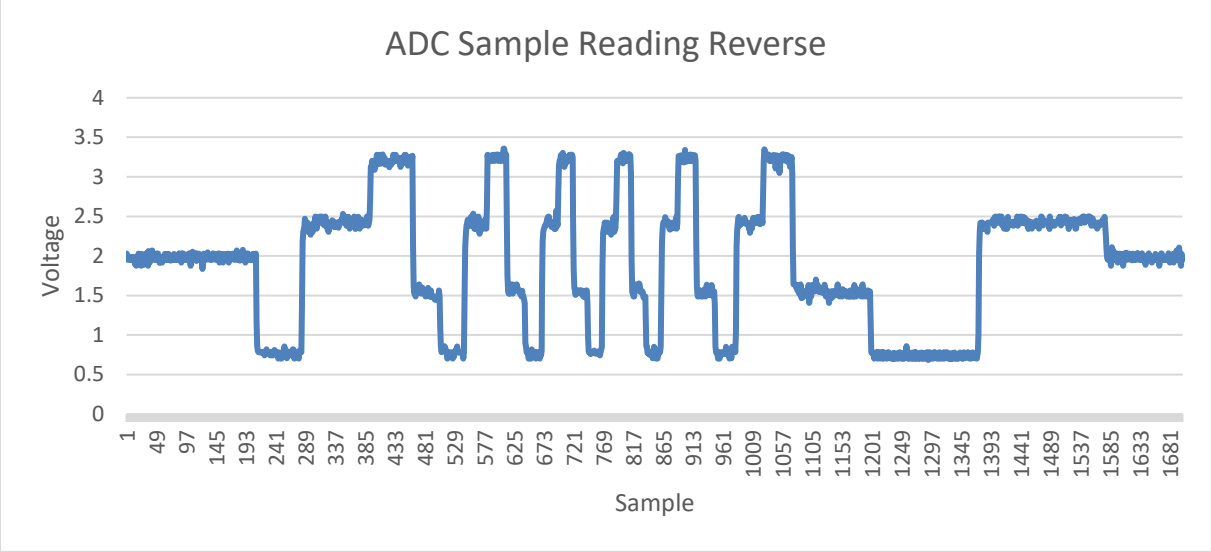


Figure 3.9: Sample reading of 0.1 mm reverse motion

3.4.1 Interpretation sequence

Each line of G-Code corresponds to one continuous movement in one direction. It is delivered to the machine as a destination, and the interpreter on board the CNC machine (GRBL) takes the current location of the machine, and translates this new destination into a direction and a distance. Therefore, the code interprets the voltage data through two main loops: direction determination and step counting. Each loop begins by reading each voltage point and determining the state of the motor (1 – 4, or no motion). The motor states are processed until two consecutive states are known and direction can be determined. The code then restarts at the

beginning of the data and counts each step. When the end of the data is reached, or the direction changes, the line of G-Code which commanded that movement is printed.

The code relies on the acceleration and deceleration of the motors to properly interpret the data. The GRBL G-Code processor accelerates and decelerates the motor according to its internal programming. An additive manufacturing machine cannot change direction instantaneously, the motors must be accelerated and decelerated for accurate material deposition. The same is true for CNC machines. At the beginning and end of a single motor movement, the motor is moving relatively slowly. However, it moves much faster in the middle, as seen in Figures 3.8 and 3.9. The effect is much more pronounced in longer motions, as the motor has more distance to accelerate. It can be fast enough that the four distinct states cannot be read accurately. Voltage data at high speed is seen in Figure 3.10. The voltage signal effectively becomes a sinusoidal wave with a peak and a trough. The RC circuit which filters the noise of the current sensors causes this loss of information. The time constant of the RC circuit is too low for the rapid change in current to charge and discharge the capacitor. This loss of information is accounted for in the step counting technique.

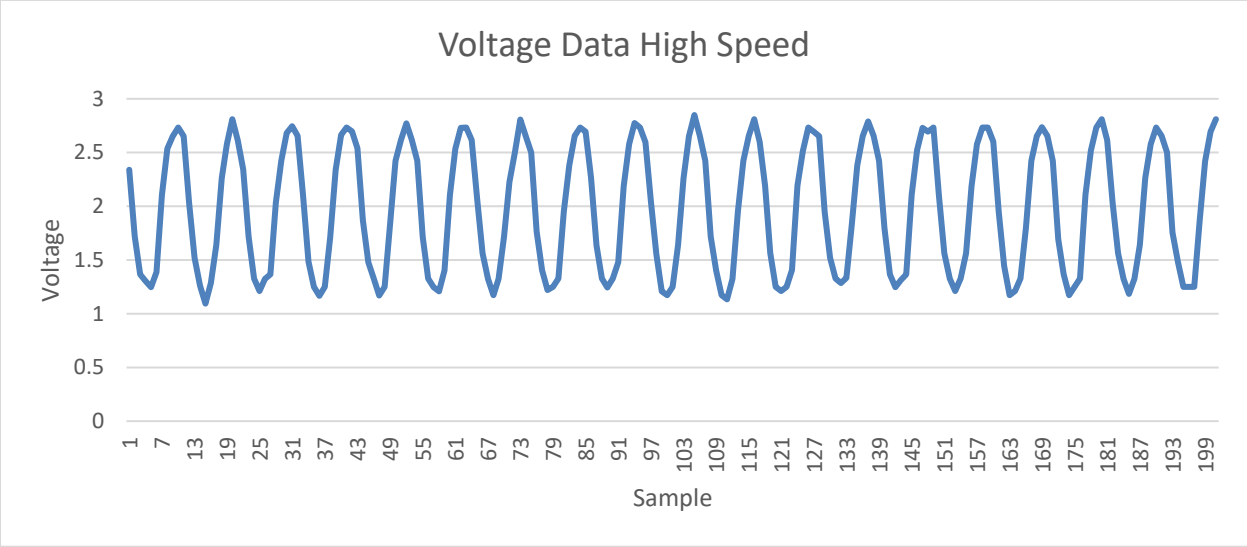


Figure 3.10: Sample of data recorded at high motor speed

3.4.2 Direction Processing

The first step in analyzing the data is to determine the direction the motor is moving. The sequence of motor steps changes based on the direction of the motor. As seen in Figure 3.8, a forward motion of 0.1 mm, the first state after no motion is state 3, followed by state 4, and then

state 1. The subsequent figure, Figure 3.9 shows the motor progressing through states 2, 1, 4, 3, which corresponds to a reverse motion. The entire sequence of data does not need to be read to determine direction. Only two consecutive motion states are needed. When direction is determined, the loop is exited.

3.4.3 Step Counting

Step counting is the key to determining the distance traveled by the stepper motor. As shown in the voltage data table, the four distinct voltage levels correspond to each motor step. At high speed the four distinct voltage levels can no longer be read, but the voltage data is still useful for counting steps. The slow steps at the beginning of the motion are used for state determination. Then, the state information is used to determine a separate "step state". The step state still has four distinct states, but is not determined entirely by voltage ranges. Instead, the program contains an upper and a lower voltage threshold. When the voltage is greater than the upper threshold, the step state is advanced, and a step is counted. When the voltage then drops below the upper threshold, the step state is advanced, and another step is counted. The same occurs for the lower threshold. For example, state 3 (1.6 V) is the first state recorded in Figure 3.8. The code then sets the step state to 3 and begins looking for step state 4. Step state 4 is reached when the voltage crosses above the upper threshold. The code begins looking for step state 1, which is reached when the voltage drops below the upper threshold. If the direction is reversed, the sequence of step states is reversed. The voltage thresholds are set so the step counting works at high and low speed. The upper voltage threshold is 2.7 V and the lower is 1.3 V.

3.4.4 Interpretation Techniques

State determination is inaccurate when the motor is traveling quickly or transitioning between states. The state is only determined when an accurate reading can be ensured. This is performed by calculating the standard deviation of 20 points surrounding the current data point being read. If the standard deviation is above a threshold, the motor must be traveling quickly or in a transition between two states. If so, the state is not determined and the code progresses until it finds a point where state can be read accurately.

The code can also process transitions between multiple lines of G-Code. As the machine transitions to the next line of G-Code instruction, the motor must change direction and decelerate, which means the motor state can once again be determined accurately. Motor state is checked continuously with the next state in the progression always known. If a state different than the anticipated state is reached, a direction change has occurred. For example, as seen in Figure 3.11, if the motor is moving forward and the state is 1, state 2 will be expected. If state 4 is recorded, the motor has changed direction. If this occurs, the script processes a line of G-Code and resets the step counter for the next line. Direction determination does not need to happen again because it is merely switched from the known previous direction.

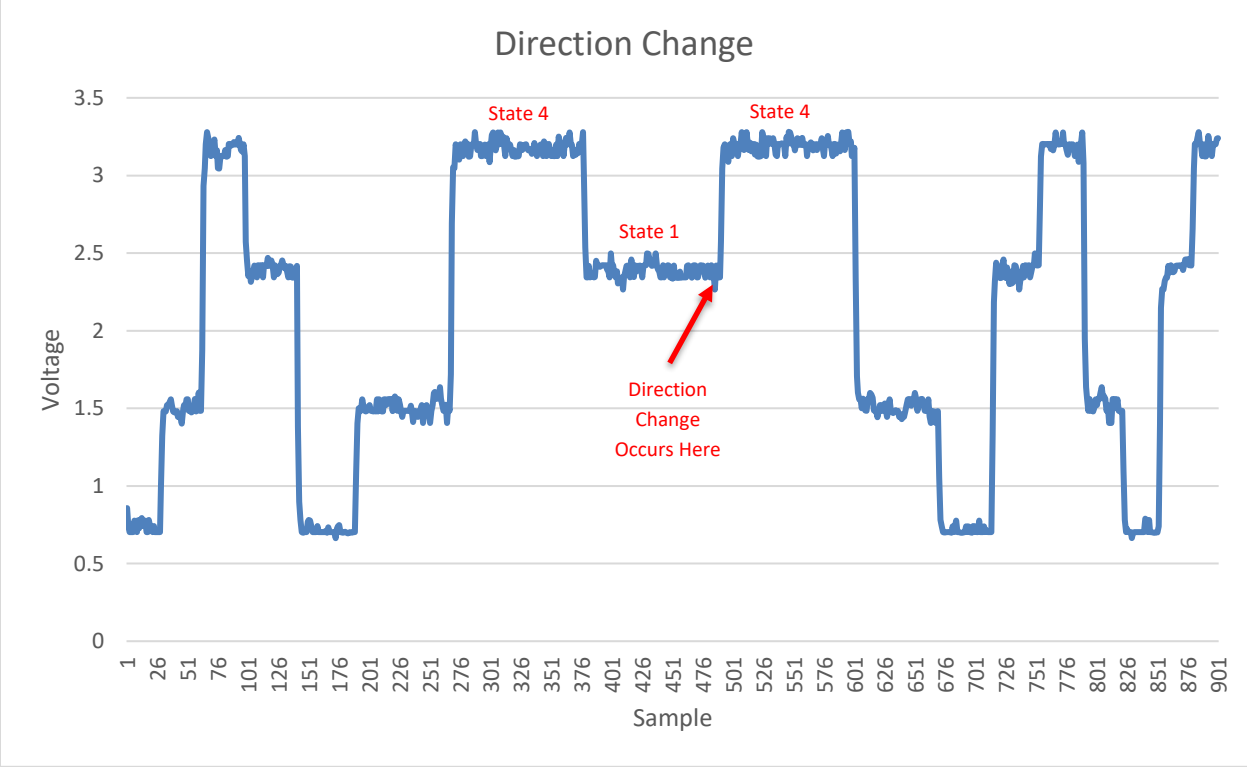


Figure 3.11: Sample of voltage data for motor directional change

3.4.5 Translation to G-Code

Three pieces of information are needed to reverse engineer the G-Code instructions: 1) direction, 2) distance, and 3) previous motor location. A single line of G-Code gives a target location and sometimes a speed or feed rate. If feed rate is not given, the G-Code interpreter just uses the last commanded feed rate. If every step from a single line of G-Code is recorded, the exact distance and direction traveled by the motor can be determined if machine parameters are

known. For example, in Figure 3.8, 26 discrete changes in voltage are visible, corresponding to 25 steps. The first change in voltage is the initial state of the motor when it is powered on. It does not move for this first step. This state matches the final state from the motor's previous motion. The machine is set to translate 0.004 mm for one step, so 25 steps corresponds to a translation of 0.1 mm. Therefore, the G-Code instruction for a 0.1 mm translation in the positive direction would be G1 Y0.1, assuming the machine was started from Y = 0. The original position must be known to recover the target position from distance and direction traveled. In a full G-Code program for an entire part each line must be correctly processed to track the location of the motor and process the new successive line. When a direction change is processed as described previously, the current line of G-Code is printed, and the new Y location is recorded. The Y location is updated each time a line of G-Code is printed and processed. Lines of G-Code are printed continuously until the end of the voltage data is reached.

CHAPTER 4

RESULTS AND DISCUSSION

To validate the hacking system and prove the possibility of this type of attack, several different tests are conducted. Step counting and direction determination are the keys to reconstructing G-Code, so the tests will validate both factors. The system must properly recognize direction and count steps in any scenario a 3D printer may run its motors. Chapter 4 is divided into the following sections. Section 4.1 covers the validation of the device for all types of movement. 4.2 describes the success of G-Code reconstruction. Finally, section 4.3 addresses G-Code information that cannot be recovered.

4.1 Validation

The primary variable one could encounter is the start and end state of a programmed movement. If the motor is turned to state 1, it will begin at state 1. The same could occur for states 2-4. The motor can also end on any state. Therefore, the hacking system must correctly determine steps and direction beginning on each state and ending on each state traveling in each direction. For example, a motor can begin on state 1 and end on state 1. The motor can begin on state 1 and end on state 2, etc. Four start and end states in two directions requires a total of 32 tests. The testing procedure uses the following sequence:

1. The simulated 3D printer is set to location Y = 0
2. The hacking device begins recording data to a unique text file.
3. A G-Code command is issued to the machine.
4. The command is executed
5. The hacking device stops recording.

After each test, the machine is reset to the starting point and the next test is run. The starting point for each test depends on the start state desired. For example, at a starting point of Y = 0, the start state recorded for this validation was state 3. It could be any state depending on previous motor motion. As long as all four states are verified, matching state 1 to Y = 0 is not necessary. The G-Code command issued is intended to have the machine travel beyond 25 steps or 0.100 mm. The G-Code commands issued instruct the motor to travel 0.104 mm, 0.108 mm,

0.112 mm, and 0.116 mm, so each end state can be validated. Once all four end states are validated for a single start state, the start state is moved by 0.004 mm and all four end states are tested. The G-Code command distances must be increased by 0.004 mm to achieve the desired travel. The voltage data for each test is recorded and transferred to the Matlab interpreter to validate the steps counted and direction. The data is also plotted in Excel so the start and end states can be verified visually. An example of the plotted data is seen in Figure 4.1. The plot for every test can be found in Appendix A.

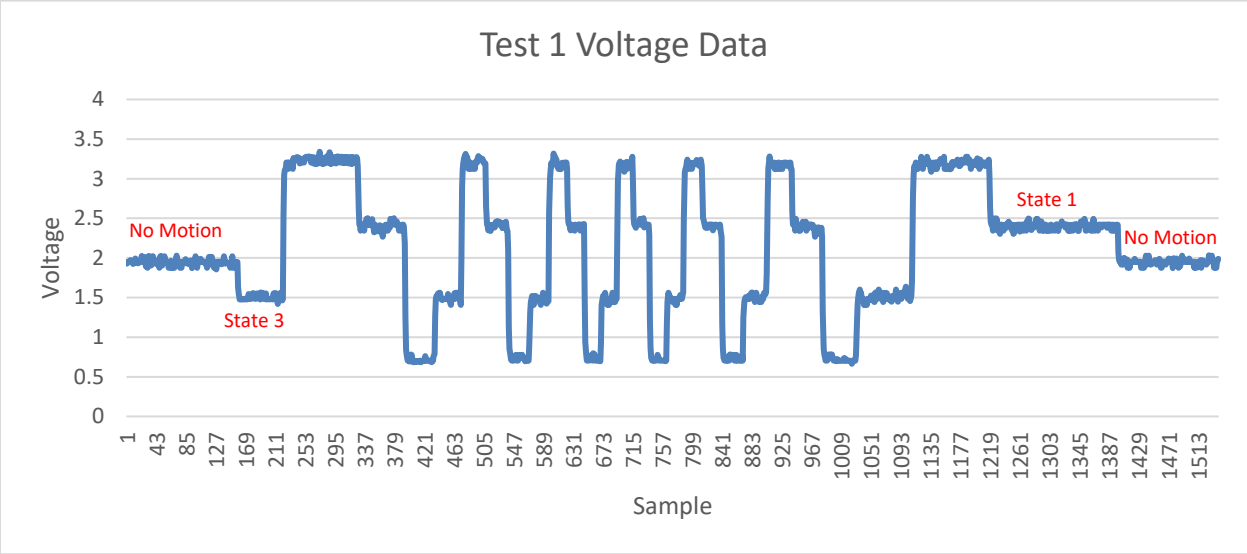


Figure 4.1: Sample of voltage data for testing

The test parameters and results are summarized in the following two tables showing the parameters for each test conducted. There are eight total tables for each starting state in each direction. Each is labeled with a starting state and a starting location. Each table displays the following information for each of the four end states: the G-Code command issued, expected distance traveled, expected steps traveled, the steps counted, and the G-Code command returned. The steps counted should match the expected steps traveled, and the G-Code command returned should match the distance traveled. It does not match the G-Code command issued because the G-Code command necessary to cover the intended distance varies based on the starting point.

Forward:

Start State: 3

Start Distance: 0

Table 4.1: Tests 1-4

Test	End State	G-Code Command	Steps Expected	Steps Returned	Distance Traveled	G-Code Returned
1	1	Y.104	26	26	0.104	Y0.104
2	2	Y.108	27	27	0.108	Y0.108
3	3	Y.112	28	28	0.112	Y0.112
4	4	Y.116	29	29	0.116	Y0.116

Start State: 4

Start Distance: 0.004

Table 4.2: Tests 5-8

Test	End State	G-Code Command	Steps Traveled	Steps Returned	Distance Traveled	G-Code Returned
5	1	Y.120	29	29	0.116	Y0.116
6	2	Y.108	26	26	0.104	Y0.104
7	3	Y.112	27	27	0.108	Y0.108
8	4	Y.116	28	28	0.112	Y0.112

Start State: 1

Start Distance: 0.008

Table 4.3: Tests 9-12

Test	End State	G-Code Command	Steps Traveled	Steps Returned	Distance Traveled	G-Code Returned
9	1	Y.120	28	28	0.112	Y0.112
10	2	Y.124	29	29	0.116	Y0.116
11	3	Y.112	26	26	0.104	Y0.104
12	4	Y.116	27	27	0.108	Y0.108

Start State: 2

Start Distance: 0.012

Table 4.4: Tests 13-16

Test	End State	G-Code Command	Steps Traveled	Steps Returned	Distance Traveled	G-Code Returned
13	1	Y.120	27	27	0.108	Y0.108
14	2	Y.124	28	28	0.112	Y0.112
15	3	Y.128	29	29	0.116	Y0.116
16	4	Y.116	26	26	0.104	Y0.104

Reverse:

Start State: 3

Start Distance: 0

Table 4.5: Tests 17-20

Test	End State	G-Code Command	Steps Expected	Steps Returned	Distance Traveled	G-Code Returned
17	1	Y.-104	26	26	-0.104	Y-0.104
18	2	Y.-116	29	29	-0.116	Y-0.116
19	3	Y.-112	28	28	-0.112	Y-0.112
20	4	Y.-108	27	27	-0.108	Y-0.108

Start State: 2

Start Distance: 0.004

Table 4.6: Tests 21-24

Test	End State	G-Code Command	Steps Traveled	Steps Returned	Distance Traveled	G-Code Returned
21	3	Y-.112	27	27	-0.108	Y-0.108
22	4	Y-.108	26	26	-0.104	Y-0.104
23	1	Y-.120	29	29	-0.116	Y-0.116
24	2	Y-.116	28	28	-0.112	Y-0.112

Start State: 1

Start Distance: 0.008

Table 4.7: Tests 25-28

Test	End State	G-Code Command	Steps Traveled	Steps Returned	Distance Traveled	G-Code Returned
25	1	Y-.120	28	28	-0.112	Y-0.112
26	2	Y-.116	27	27	-0.108	Y-0.108
27	3	Y-.112	26	26	-0.104	Y-0.104
28	4	Y-.124	29	29	-0.116	Y-0.116

Start State: 4

Start Distance: 0.012

Table 4.8: Tests 29-32

Test	End State	G-Code Command	Steps Traveled	Steps Returned	Distance Traveled	G-Code Returned
29	3	Y-.128	29	29	-0.116	Y-0.116
30	4	Y-.124	28	28	-0.112	Y-0.112
31	1	Y-.120	27	27	-0.108	Y-0.108
32	2	Y-.116	26	26	-0.104	Y-0.104

For each test, all steps are counted properly and the expected G-Code command is returned. The only errors encountered in testing were human errors like sending the command before the hacking device had begun recording. If the test is run properly, the hacking device and the code worked flawlessly. Therefore, the hacking device and code can be expected to work for any motor start and end state in either direction.

4.2 G-Code Reconstruction

The true purpose of this demonstration is to recreate sequences of G-Code commands like those found in 3D printing data. A secure AM system would encrypt these G-Code commands and they would not be accessible by the printer operator. Therefore, if they can be recreated from stepper motor signals, they could potentially be stolen by a malicious printer operator. As shown in the previous validation section, recreation of single lines of G-Code is successful. Multiple consecutive commands must also be validated.

4.2.1 Multiple Commands

The strategy for decoding multiple commands is illustrated in section 3.4.4 and 3.4.5 and relies on changes in motor direction to identify the ends of G-Code commands. When an unexpected step is recorded, a direction change has occurred and the line of G-Code is calculated based on direction, steps, and previous Y location. The steps are then reset and the direction is changed. A similar testing strategy as before has been taken for these transitions. A transition can occur on any motor state, so a transition on each must be recorded and verified. This must be performed in both directions to ensure any type of motor motion is decryptable. A similar testing procedure was performed as in section 4.1. Each test began at Y = 0 and two G-Code commands were sent to the motors. The first was a command to travel to a point, the second to return to the origin at Y = 0. The point each traveled to was increased by 0.004 mm to verify that a transition on each motor state was possible. Tables summarizing the testing follow.

Forward:

Table 4.9: Tests 33-36

Test	Transition Step	Direction	Steps Traveled	Steps Returned	G-Code Command	G-Code Returned
33	1	Forward	26	26	Y.104	Y0.104
		Reverse	26	26	Y0	Y0.000
34	2	Forward	27	27	Y.108	Y0.108
		Reverse	27	27	Y0	Y0.000
35	3	Forward	28	28	Y.112	Y1.112
		Reverse	28	28	Y0	Y0.000
36	4	Forward	29	29	Y.116	Y1.116
		Reverse	29	29	Y0	Y0.000

Reverse:

Table 4.10: Tests 37-40

Test	Transition Step	Direction	Steps Traveled	Steps Returned	G-Code Command	G-Code Returned
37	1	Reverse	26	26	Y.-104	Y-0.104
		Forward	26	26	Y0	Y0.000
38	4	Reverse	27	27	Y.-108	Y-0.108
		Forward	27	27	Y0	Y0.000
39	3	Reverse	28	28	Y.-112	Y-0.112
		Forward	28	28	Y0	Y0.000
40	2	Reverse	29	29	Y-.116	Y-0.116
		Forward	29	29	Y0	Y0.000

Each contains the transition state, direction of travel, expected steps traveled, steps counted and returned, the G-Code commands issued, and the G-code commands returned. With this testing procedure the returned G-Code matches the delivered G-Code because each test begins from Y = 0, the default start for the interpretation code. Again, each test is successful in returning the exact number of steps traveled and the expected G-Code, so the system accurately tracks any direction transition.

4.2.2 Several Commands

The final demonstration of the attack is to recreate several consecutive lines of G-Code. To do so, a sequence of multiple G-Code instructions are programmed to run on bCNC. The instructions range from long to short traverses in alternating directions. Like the previous test, the hacking device starts recording to a text file, and the G-Code program is run. The data is then processed in Matlab to recreate the G-Code commands. The G-Code programmed to run uses the following ten instructions:

Table 4.11: Test G-Code Instructions

G90
G21
G1 Y6
G1 Y4
G1 Y5
G1 Y1
G1 Y2.5
G1 Y0
G1 Y0.004
G1 Y0
G1 Y0.1
G1 Y0

G90 was included to set the dimensions to absolute rather than incremental. G21 was added to set the units to mm. These are purely for formatting and will not be read by the device. The test was conducted 20 times and in all but four, the G-Code was output with 100% accuracy. The four incorrect readings produced the following lines with the incorrect reading highlighted in red:

Table 4.12: Test G-Code Results

Test 2	**Test 3**	**Test 6**	**Test 10**
G1 Y6.000	G1 Y6.000	G1 Y5.984	G1 Y6.000
G1 Y4.000	G1 Y4.048	G1 Y3.984	G1 Y4.000
G1 Y5.000	G1 Y5.048	G1 Y4.984	G1 Y5.000
G1 Y1.016	G1 Y1.048	G1 Y0.984	G1 Y1.032
G1 Y2.516	G1 Y2.548	G1 Y2.484	G1 Y2.532
G1 Y0.016	G1 Y0.048	G1 Y-0.016	G1 Y0.032
G1 Y0.020	G1 Y0.052	G1 Y-0.012	G1 Y0.036
G1 Y0.016	G1 Y0.048	G1 Y-0.016	G1 Y0.032
G1 Y0.116	G1 Y0.148	G1 Y0.084	G1 Y0.132
G1 Y0.016	G1 Y0.048	G1 Y-0.016	G1 Y0.032

In each case, only one error occurred, but it caused the subsequent G-Code commands to all be incorrect as well. This is because the location derived from each step count is based on the old location, so the error in location remains throughout the data processing.

The errors all occurred because too few steps were counted for the intended motion. Test 2 missed four steps (0.016 mm), test 3 missed twelve, test 6 missed four, and test 10 missed eight. Notably these are all multiples of four. The motor cycles through four states and the counting cycle looks for each progressive state when counting steps. Due to this, if a step state is not counted, the counting function will not look for the next state in the cycle. For example, if the counting function is looking for state 1, and state 1 is missed, the motor must cycle through states 2, 3, and 4 before it finds state 1 again. That means all four of these steps will be missed.

The raw data recorded by the hacking device can be inspected to understand why these errors occur. The data is plotted in Excel, where individual steps can be counted. In test 2, around data point 24,793, the voltage data goes from step state 2 to step state 1, then back to step state 2 without ever crossing the upper threshold of state 3. This missed step triggers the four missed steps seen in line four of test 2. The hacking device also records a time stamp for each data point and the time stamp of data point 24793 occurs 0.000443 seconds after data point 24792. The typical time between data points is around 0.000182 seconds. This indicates that data points were not recorded during the time delay and the time the motor entered state 3 was not recorded. It is unclear why this extra time delay occurred. A delay of this magnitude is extremely rare. In test two, a delay twice the typical 0.000182 seconds happened only three times, and a delay 1.5 times the standard 0.000182 seconds happened eight times in 50,000 samples. Unfortunately, this delay can have a significant effect on the final outcome as errors due to missed steps are never corrected. Perhaps the relatively simple, low cost hacking device is to blame. More time spent ensuring a consistent sample rate could eliminate this error.

The other three failed tests were inspected by counting the individual steps in each incorrectly calculated line of G-Code. A hand count gives the exact same tally of steps counted by the Matlab program. Occasional time delays occur, but inspection of the data shows these did not have an effect on step counting and the proper progression of step states occurs. The GRBL program used to send the G-Code commands contains a complex algorithm for accelerating and decelerating motors to reach the desired locations. It is possible that GRBL is subtracting steps

to compensate for the accelerations and decelerations. There could also be unknown issues with the hacking device causing the improper recording of voltage data.

Despite these errors, a total of four lines of G-Code out of 200 total lines (10 per test for 20 tests) had errors for an error rate of 2%. Each test includes 4302 steps, for a total of 86040 steps across all twenty tests. In total 28 steps were not counted. This is an error rate of 0.03 %. The largest error was a miscount of twelve steps corresponding to a linear error of 0.048 mm. A linear error of this magnitude could be significant depending on the application.

Even basic 3D printed parts require thousands of lines of G-Code, so this error rate may be unacceptable. More time spent investigating these errors may eliminate them. More expensive and complex voltage recording devices may also help. However, this demonstration is a promising start to the ultimate goal of stealing an entire part from motor signals.

4.3 Missing Information

In addition to the errors in recording, some pieces of manufacturing information cannot be determined from the recorded data. A feed rate must be defined for G1 commands. Typically, the feed rate is given in the first G1 command in a sequence and is only given again if it changes or a repositioning G0 move occurs. It is not completely necessary for each line. One approach to determining feed rate could be to measure the frequency at which a certain step state is triggered. For example, each time a peak at state 3 is reached, the time stamp accompanying the data point would be recorded. When the peak occurs again, the time difference between the two peaks can be used to determine the speed of the motor. However, depending on the controller, the fastest speed recorded may not match the desired feed rate. If the motor is accelerated like it in this example, the max speed may not be reached if the traverse distance is too short. A careful analysis of GRBL internal programming may reveal that feed rates can be determined from acceleration rates and travel distances, but this is beyond the scope of this thesis.

Furthermore, knowledge of feed rate is not entirely necessary. Normally, the maximum possible feed rate is used to maximize throughput of a 3D printer. The limiting factor in many cases is the material being deposited. A 3D printing service would typically know the feed rate for a particular material, so the machine's feed rate is not essential information.

This method also fails to detect consecutive G-Code commands in the same direction. For example, a command of G1 Y3 followed by G1 Y6 would be read only as G1 Y6. A reversal in motor direction is needed to detect the end of a line of G-Code. This typically would not be an issue as the final destination of the motor would still be recorded. In a more advanced demonstration of an attack, two motors would be tracked to outline two dimensional shapes. A reversal in one motor would indicate the end of a command in two motors. For example, consider two commands issued in succession starting from X0 Y0: G1 X3 Y3, G1 X0 Y6. The direction reversal in X to go from 0 to 3 back to 0 would indicate a new line of G-Code. Therefore, the Y data recorded at the same time of the reversal would indicate the end of the line of code.

Recording multiple motors is necessary to recreate the entirety of the G-Code. The same technique for one motor could be repeated for any number of motors. This experiment was limited to the recording capability of a low-cost AD converter. However, a more sophisticated and expensive data acquisition device would be able to record the streams of data from several motors. Time synchronization of all the data would be essential. In a typical FDM system, the X, Y, and extruder motors all work simultaneously, so their coordination would need to be perfect.

This demonstration is just a start, but these experiments and the analysis of the hacking device and interpretation code suggest that a side channel attack of stepper motor signals is possible. Simulated encrypted machine instructions have been hacked by reading the magnetic field generated by the current passing through stepper motor wire.

CHAPTER 5

CONCLUSION

This thesis has outlined the state of the industry for a distributed manufacturing system and the potential advantages of such a system as additive manufacturing technology advances. It has also highlighted some of the current security threats to a distributed manufacturing environment, specifically unauthorized reproduction of parts. It then demonstrated the possibility of a side channel attack to access encrypted manufacturing information for an additive manufacturing machine. The demonstration presented within this thesis circumvents one approach to preventing unauthorized reproduction: the encryption of manufacturing instructions.

5.1 Concluding Remarks

The attack demonstrated in this thesis was to determine motor motion based on magnetic fields generated by stepper motor control signals. This motor motion was then translated into G-Code manufacturing instructions. The system of sensors, data recorders, and processers successfully recreated a comprehensive series of tests to ensure the system will work for any forward and reverse motor motion. It will also work for any transition between forward and reverse motion. Finally, a sequence of G-Code instructions similar to what is used for a real 3D printed part was reproduced. While not 100% successful, the success rate was high enough to conclude that this method could be used for a long sequence of data. Chapter 4 presented some suggestions for why these errors occurred and future refinement of the system may lead to a 100% success rate. Investigation of the errors did not necessarily show that all the errors were caused by the hacking device. The errors could be caused by processes within the simulated 3D printer itself.

5.2 Contributions

A fully secure distributed manufacturing system must prevent any type of attack. Data theft is one of many types of attacks that could occur. Prevention of data theft is the only way to ensure widespread adoption of distributed manufacturing in advanced technology areas. Intellectual property is too valuable to be left vulnerable. The attack demonstrated in this thesis

is very far from a full theft of advanced technology data. However, it proves that encryption of manufacturing data up to the computer of a 3D printer is not fully secure. Therefore, those wishing to build fully secure systems must account for the information contained in wire signals. An obvious approach would be to encase the wires so that their external magnetic fields cannot be read. It may be difficult to do so, as the wires must still freely translate along with their motors. Repairs would also be difficult in an encasement. Perhaps a motor controller with a trusted platform module could be integrated directly into the motors themselves. This would eliminate the long stretches of wire needed to detect the signals.

However engineers decide to solve this problem, they must be aware that a system with hardware costs totaling less than $100 was able to present a successful rudimentary attack on a simulated secure system. In theory, every system can be hacked with enough resources. A system is only secure when the costs to hack it exceed the financial benefit. More advanced sensors, data recorders, and processors would not be prohibitively expensive. Advanced technology intellectual property can be worth millions of dollars, so it must be protected.

5.3 Recommendations for Future Work

This demonstration required motor control wires to be cut and run through the current sensors. The circuit to the motor coils was not interrupted, but an attack like this could leave evidence of tampering. An ideal attack would leave the wires as they are, with a sensor merely attached to the exterior of the wire. This would require more sensitive, expensive sensors, but it would more accurately demonstrate the type of attack a secure system is vulnerable to.

The motors used in the demonstration were not physically connected to any load. In a real FDM system, they would drive the motion of a print head, print bed, and the object being printed. A load on a stepper motor can alter the current draw of that motor. Since the attack relies on detecting motor current, an attack in a real scenario may be affected by these loads. Implementing this attack on a motor carrying a load could give more insight into this question. The current measurement technique here does not rely on extremely precise current measurements. It only needs to detect whether a motor coil is in a binary high or low state. While significant loading may alter the current draw of a motor, minor fluctuations in loading conditions would likely produce only a minor fluctuation in current. Experiments would need to be run to determine how much loading would be necessary to move current outside of its binary

ranges. If this loading is within the realm of possibility for a 3D printer this type of attack may not be possible. Variable motor loading could also be used to thwart such an attack.

Another avenue for exploration would be the effect of micro stepping. Micro stepping is a motor control technique in which stepper motor coils are powered to increments of their full value to turn the motor shaft to a state between full steps. More precise motor control can be achieved, but at the cost of motor torque. An attack on a motor using micro stepping would require a more precise current detection technique. However, if the variable current could be properly recorded, it is likely that steps could be counted just as demonstrated in this thesis. This could pose a problem to the "binary" interpretation technique explained in the previous paragraph. Smaller steps and smaller variations in current could be more susceptible to motor loading. A fully robust system would have to account for micro stepping and motor loading.

Also, in this demonstration, the motion of only one motor was reconstructed. In reality, the motion of one motor is useless in an additive manufacturing environment. A standard FDM system uses several motors to translate the print head and extrude material. The X and Y motors work together to move the print head in a standard cartesian coordinate system while the Z motor moves the print head to deposit each new layer. Extruder motors push filament through the extruder nozzle along the path the X and Y motors generate. All of these motors would need to be recorded to properly reconstruct an entire part. Typically, these are all similar stepper motors, so the approach to hacking any one motor is the same as hacking the others. More sensors and a more complex data processing system would be necessary but would not present a significantly different problem. Time synchronization of the motors would be essential, so a hacking device capable of recording four high speed channels simultaneously would be required. To perform a full system test, one would not need to perform significant research and development of the attacking system. The approach for one motor could be repeated for all other printer motors, the data synchronized, and a full G-Code program recovered. The techniques presented in this thesis merely need to be repeated on more precise, advanced equipment to demonstrate a full system attack.

REFERENCES

[1] C. Roser, *"Faster, Better, Cheaper" in the History of Manufacturing*. Boca Raton, FL: CRC Press, 2017.

[2] S. H. Khajavi, J. Partanen, and J. Holmström, "Additive manufacturing in the spare parts supply chain," *Computers in Industry,* vol. 65, no. 1, pp. 50-63, 2014/01/01/ 2014, doi: https://doi.org/10.1016/j.compind.2013.07.008.

[3] M. Ballardini Rosa, I. Flores Ituarte, and E. Pei, "Printing spare parts through additive manufacturing: legal and digital business challenges," *Journal of Manufacturing Technology Management,* vol. 29, no. 6, pp. 958-982, 2018, doi: 10.1108/JMTM-12-2017-0270.

[4] M. Muir and A. Haddud, "Additive manufacturing in the mechanical engineering and medical industries spare parts supply chain," *Journal of Manufacturing Technology Management,* vol. 29, no. 2, pp. 372-397, 2018, doi: 10.1108/JMTM-01-2017-0004.

[5] Y. Li, G. Jia, Y. Cheng, and Y. Hu, "Additive manufacturing technology in spare parts supply chain: a comparative study," *International Journal of Production Research,* vol. 55, no. 5, pp. 1498-1515, 2017/03/04 2017, doi: 10.1080/00207543.2016.1231433.

[6] E. Rauch, M. Dallinger, P. Dallasega, and D. T. Matt, "Sustainability in Manufacturing through Distributed Manufacturing Systems (DMS)," *Procedia CIRP,* vol. 29, pp. 544-549, 2015/01/01/ 2015, doi: https://doi.org/10.1016/j.procir.2015.01.069.

[7] M. P. Salmi, Jouni; Tuomi, Jukka; Chekurov, Sergei; Björkstrand, Roy; Huotilainen, Eero; Kukko, Kirsi; Kretzschmar, Niklas; Akmal, Jan; Jalava, Kalle; Koivisto, Satu; Vartiainen, Matti; Metsä-Kortelainen, Sini; Puukko, Pasi; Jussila, Ari; Riipinen, Tuomas; Reijonen, Joni; Tanner, Hannu; Mikkola, Markku, "Digital Spare Parts," Aalto University, VTT Technical Research Centre of Finland, Aalto, Finland, March 2018 2018. [Online]. Available: http://urn.fi/URN:ISBN:978-952-60-3746-2

[8] E. Rosenbaum, "1 in 5 corporations say China has stolen their IP within the last year: CNBC CFO survey," *CNBC*. [Online]. Available: https://www.cnbc.com/2019/02/28/1-in-5-companies-say-china-stole-their-ip-within-the-last-year-cnbc.html

[9] G. Shaabany, M. Grimm, and R. Anderl, "Secure Information Model for Data Marketplaces Enabling Global Distributed Manufacturing," *Procedia CIRP,* vol. 50, pp. 360-365, 2016/01/01/ 2016, doi: https://doi.org/10.1016/j.procir.2016.05.003.

[10] M. Holland, J. Stjepandić, and C. Nigischer, "Intellectual Property Protection of 3D Print Supply Chain with Blockchain Technology," in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 17-20 June 2018 2018, pp. 1-8, doi: 10.1109/ICE.2018.8436315.

[11] J. S. Srai *et al.*, "Distributed manufacturing: scope, challenges and opportunities," *International Journal of Production Research*, vol. 54, no. 23, pp. 6917-6935, 2016/12/01 2016, doi: 10.1080/00207543.2016.1192302.

[12] L. F. C. S. Durão, A. Christ, R. Anderl, K. Schützer, and E. Zancul, "Distributed Manufacturing of Spare Parts Based on Additive Manufacturing: Use Cases and Technical Aspects," *Procedia CIRP*, vol. 57, pp. 704-709, 2016/01/01/ 2016, doi: https://doi.org/10.1016/j.procir.2016.11.122.

[13] M. Yampolskiy *et al.*, "Security of additive manufacturing: Attack taxonomy and survey," *Additive Manufacturing*, vol. 21, pp. 431-457, 2018/05/01/ 2018, doi: https://doi.org/10.1016/j.addma.2018.03.015.

[14] L. D. Sturm, C. B. Williams, J. A. Camelio, J. White, and R. Parker, "Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the .STL file with human subjects," *Journal of Manufacturing Systems*, vol. 44, pp. 154-164, 2017/07/01/ 2017, doi: https://doi.org/10.1016/j.jmsy.2017.05.007.

[15] D. B. Miller, W. B. Glisson, M. Yampolskiy, and K.-K. R. Choo, "Identifying 3D printer residual data via open-source documentation," *Computers & Security*, vol. 75, pp. 10-23, 2018/06/01/ 2018, doi: https://doi.org/10.1016/j.cose.2018.01.011.

[16] Q. Do, B. Martini, and K. R. Choo, "A Data Exfiltration and Remote Exploitation Attack on Consumer 3D Printers," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2174-2186, 2016, doi: 10.1109/TIFS.2016.2578285.

[17] A. Angrish, B. Craver, M. Hasan, and B. Starly, "A Case Study for Blockchain in Manufacturing: “FabRec”: A Prototype for Peer-to-Peer Network of Manufacturing Nodes," *Procedia Manufacturing*, vol. 26, pp. 1180-1192, 2018/01/01/ 2018, doi: https://doi.org/10.1016/j.promfg.2018.07.154.

[18] M. Yampolskiy, T. R. Andel, J. T. McDonald, W. B. Glisson, and A. Yasinsac, "Intellectual Property Protection in Additive Layer Manufacturing: Requirements for Secure Outsourcing," presented at the Proceedings of the 4th Program Protection and

Reverse Engineering Workshop, New Orleans, LA, USA, 2014. [Online]. Available: https://doi.org/10.1145/2689702.2689709.

[19] F. Chen, J. H. Yu, and N. Gupta, "Obfuscation of Embedded Codes in Additive Manufactured Components for Product Authentication," *Advanced Engineering Materials,* vol. 21, no. 8, p. 1900146, 2019/08/01 2019, doi: 10.1002/adem.201900146.

[20] N. Gupta, C. Fei, N. G. Tsoutsos, and M. Maniatakos, "INVITED: ObfusCADe: Obfuscating additive manufacturing CAD models against counterfeiting," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 18-22 June 2017 2017, pp. 1-6, doi: 10:1145/3061639:3079847.

[21] F. Peng, J. Yang, and M. Long, "3-D Printed Object Authentication Based on Printing Noise and Digital Signature," *IEEE Transactions on Reliability,* vol. 68, no. 1, pp. 342-353, 2019, doi: 10.1109/TR.2018.2869303.

[22] *Information Technology-Trusted Platform Module*, ISO, Switzerland, 05-15-2009 2009.

[23] G. Shaabany and R. Anderl, "Security by Design as an Approach to Design a Secure Industry 4.0-Capable Machine Enabling Online-Trading of Technology Data," in *2018 International Conference on System Science and Engineering (ICSSE)*, 28-30 June 2018 2018, pp. 1-5, doi: 10.1109/ICSSE.2018.8520195.

[24] M. A. A. Faruque, S. R. Chhetri, A. Canedo, and J. Wan, "Acoustic Side-Channel Attacks on Additive Manufacturing Systems," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, 11-14 April 2016 2016, pp. 1-10, doi: 10.1109/ICCPS.2016.7479068.

[25] I. P. Kubiak, A.; Stańczak, A., "Usefulness of Acoustic Sounds from 3D Printers in an Eavesdropping Process and Reconstruction of Printed Shapes," *Electronics,* vol. 9(2), 9 February 2020 2020, doi: https://doi.org/10.3390/electronics9020297.

[26] J. Gatlin, S. Belikovetsky, S. B. Moore, Y. Solewicz, Y. Elovici, and M. Yampolskiy, "Detecting Sabotage Attacks in Additive Manufacturing Using Actuator Power Signatures," *IEEE Access,* vol. 7, pp. 133421-133432, 2019, doi: 10.1109/ACCESS.2019.2928005.

[27] I. R. Gibson, David; Stucker, Brent, *Additive Manufacturing Technologies*, 2 ed. Springer-Verlag (in English), 2015, p. 498.

[28] T. Kramer, F. Proctor, and E. Messina, "The NIST RS274NGC Interpreter - Version 3," National Institute of Standards and Technology, Gaithersburg, Maryland, 2000. Accessed: 07/25/2019. [Online]. Available: https://www.nist.gov/publications/nist-rs274ngc-interpreter-version-3

[29] M. N. O. Sadiku, *Elements of Electromagnetics*, Fourth Edition ed. New York, NY: Oxford University Press, 2007.

[30] R. E. Thomas, A. J. Rosa, and G. J. Toussaint, *The Analysis and Design of Linear Circuits*, Sixth ed. Hoboken, NJ: John Wiley & Sons, 2009.

[31] *GRBL*. (2019). Accessed: 04/12/2020. [Online]. Available: https://github.com/grbl/grbl

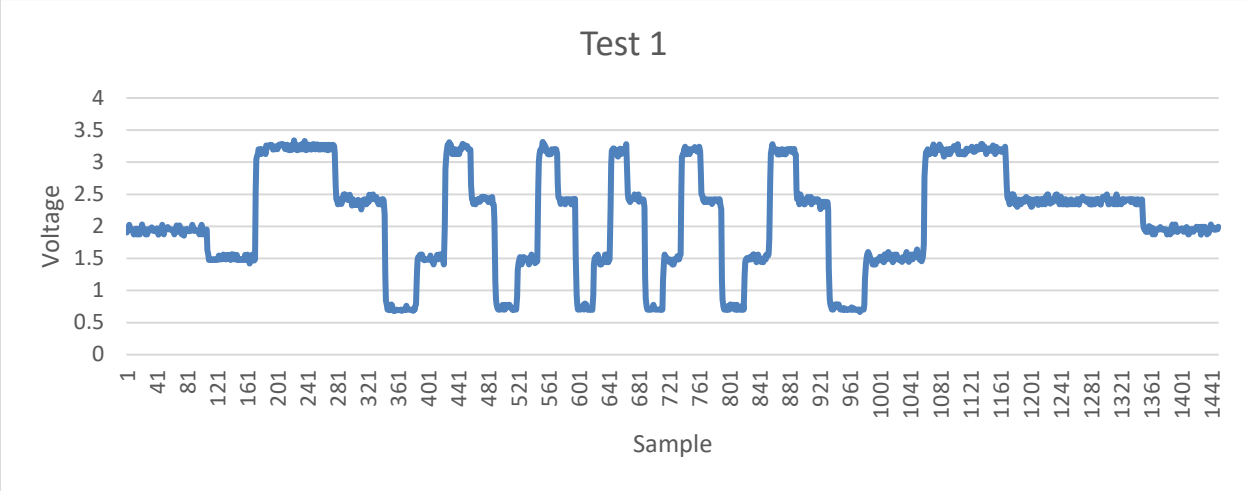
[32] *Raspberry Pi CNC User Interface SD Card Image V4.10*. (2018). Protoneer. Accessed: 04/12/2020. [Online]. Available: https://wiki.protoneer.co.nz/Raspberry_Pi_CNC_User_Interface_SD_Card_Image_V4.10

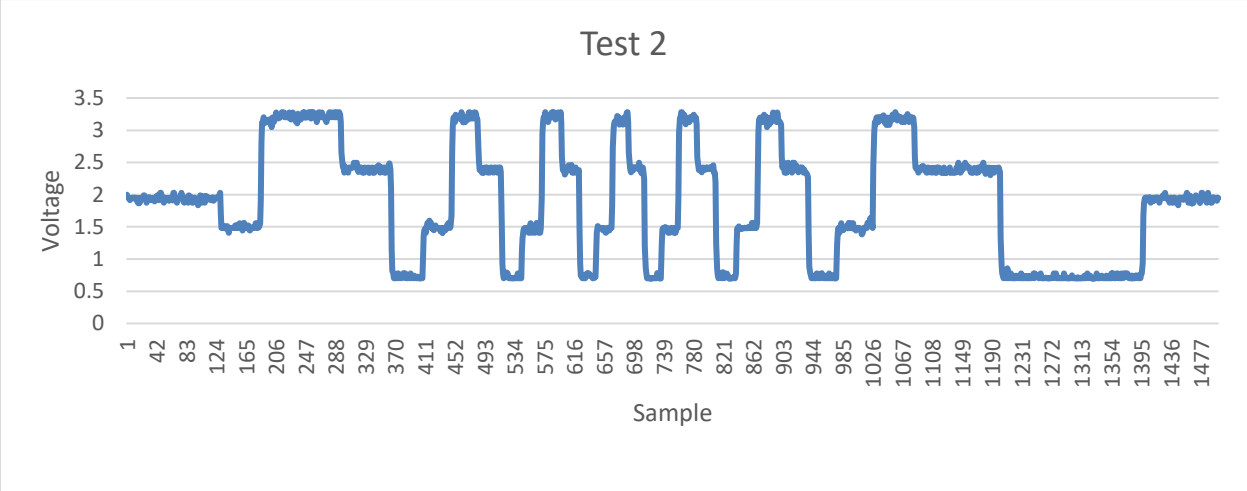
[33] *bCNC*. [Online]. Available: https://github.com/vlachoudis/bCNC/wiki

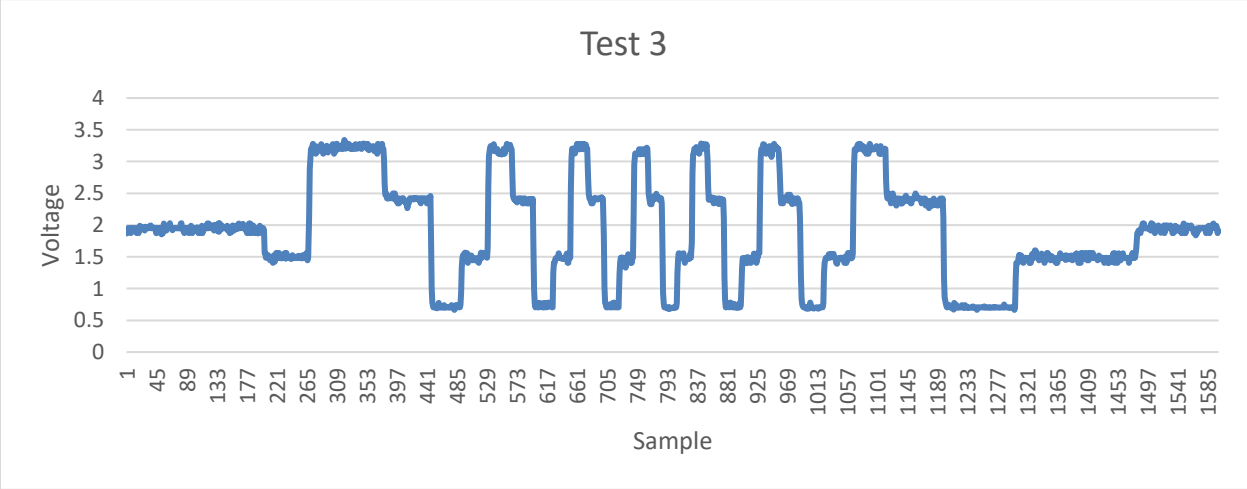
[34] *High-Precision-AD-DA-Board-Code*. (2019). Waveshare. [Online]. Available: https://www.waveshare.com/wiki/File:High-Precision-AD-DA-Board-Code.7z

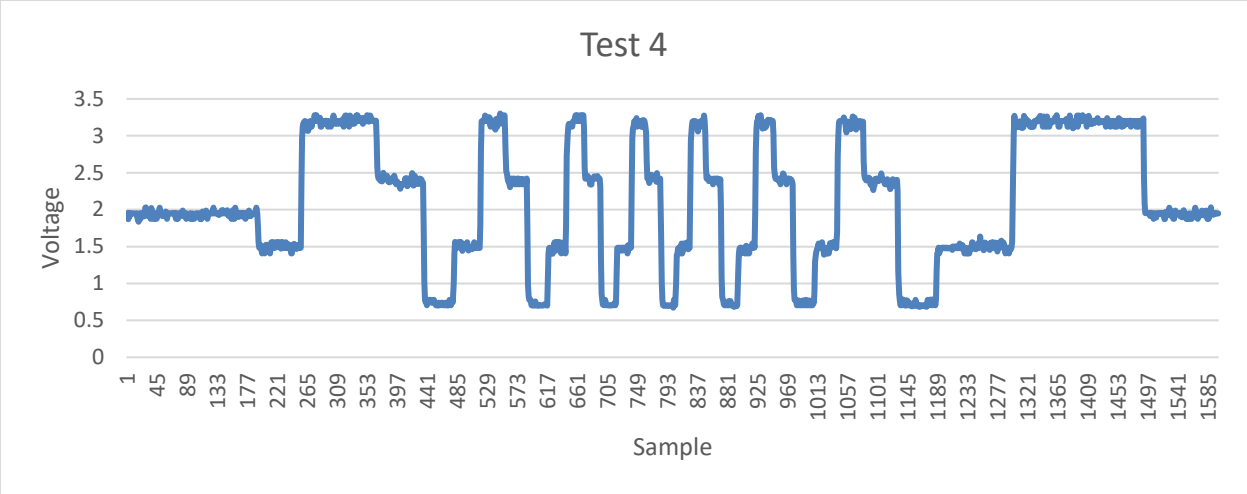
[35] *Scheme-It*. Digikey. Accessed: 04/10/2020. [Online]. Available: https://www.digikey.com/schemeit/project/

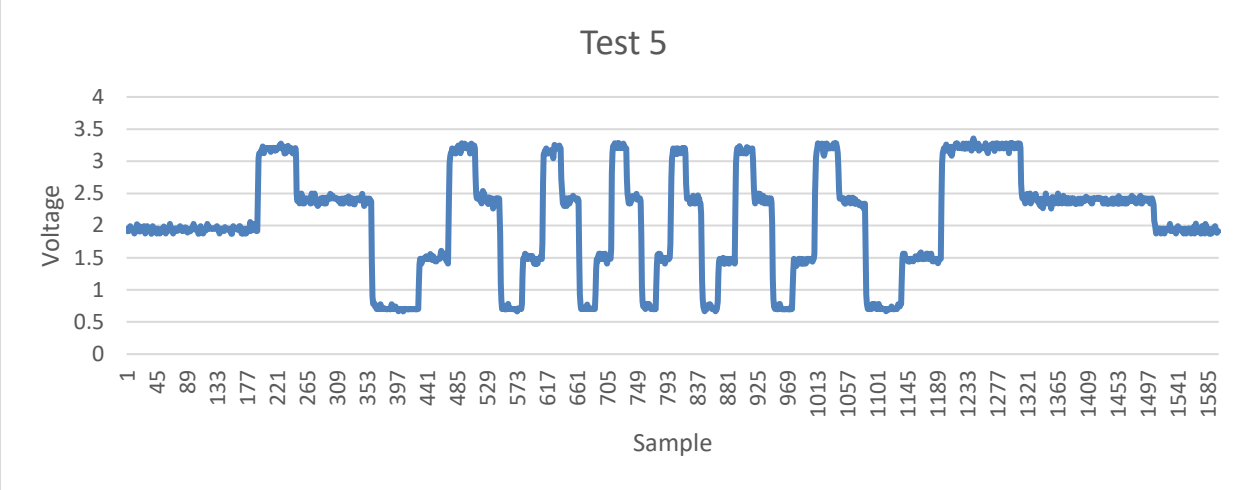
APPENDIX A: EXPERIMENTAL DATA PLOTS

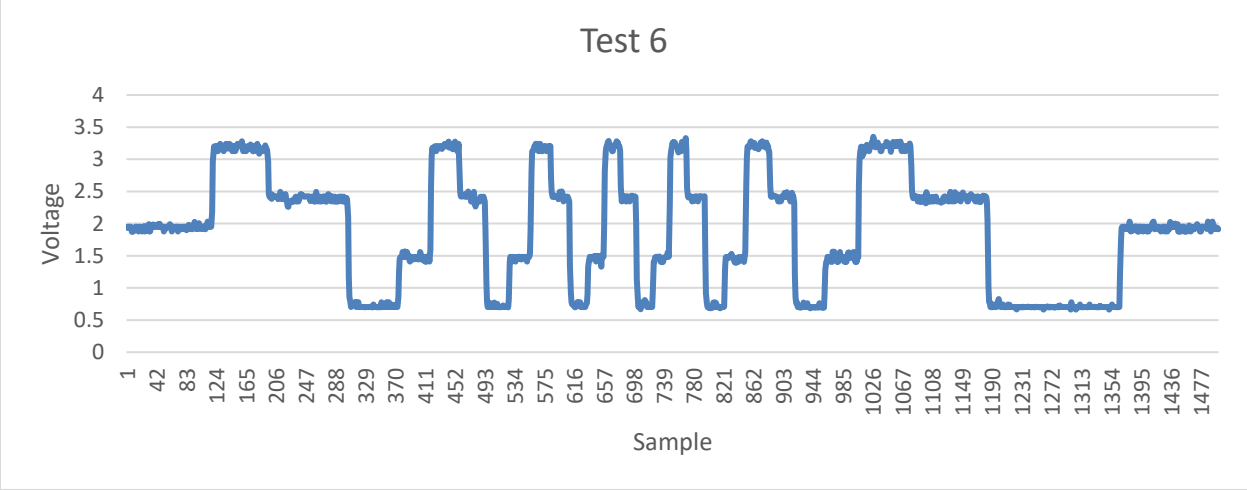


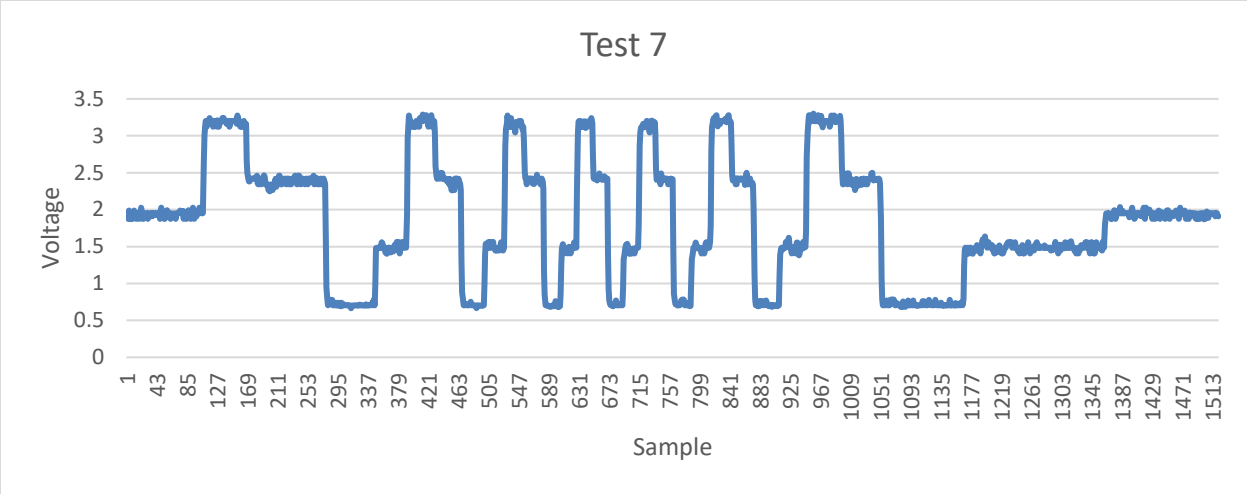


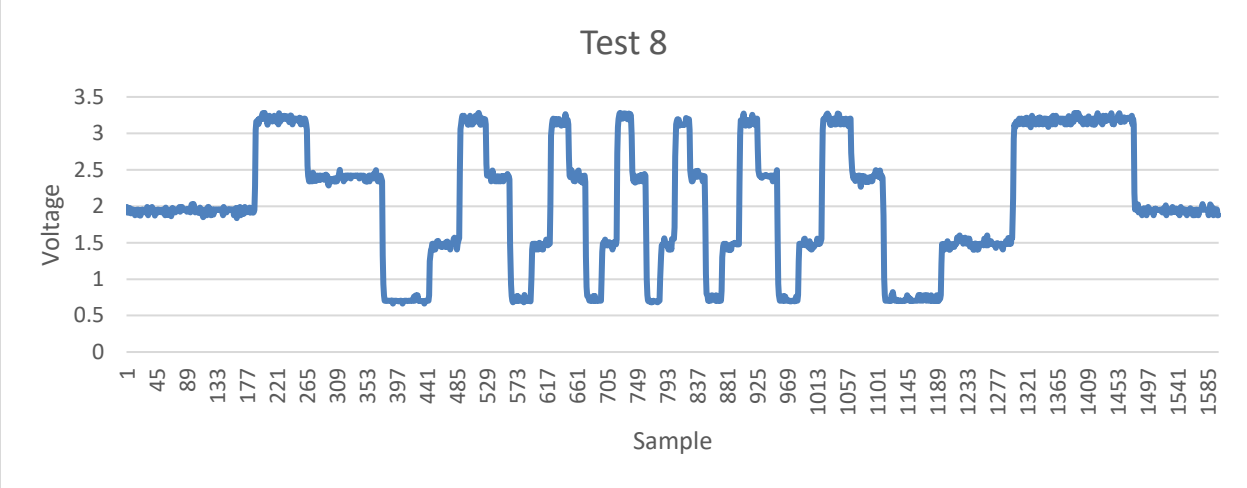


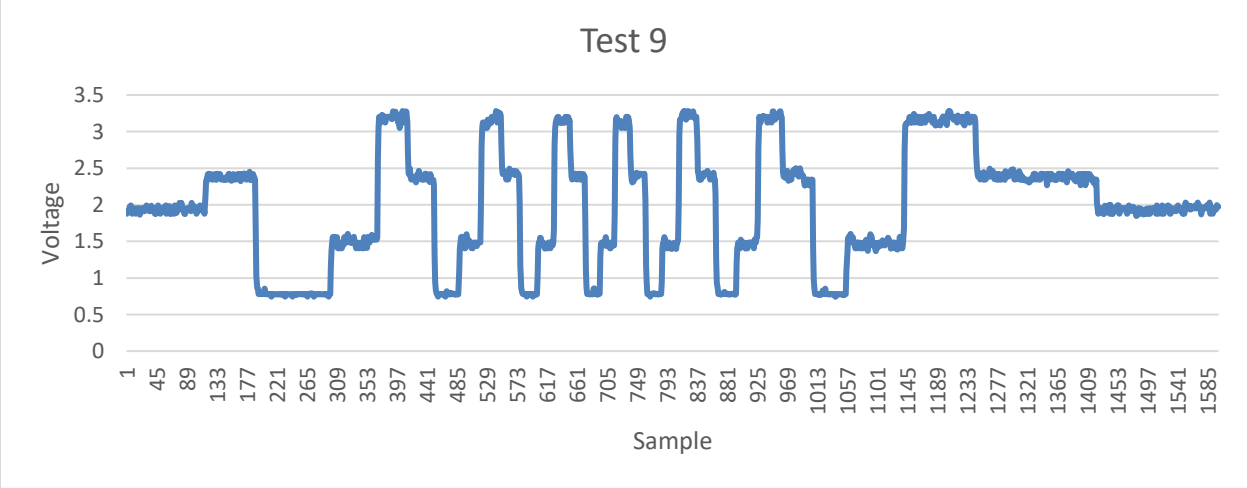

Test 4
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
45
89
133
177
221
265
309
353
397
441
485
529
573
617
661
705
749
793
837
881
925
969
1013
1057
1101
1145
1189
1233
1277
1321
1365
1409
1453
1497
1541
1585
Sample

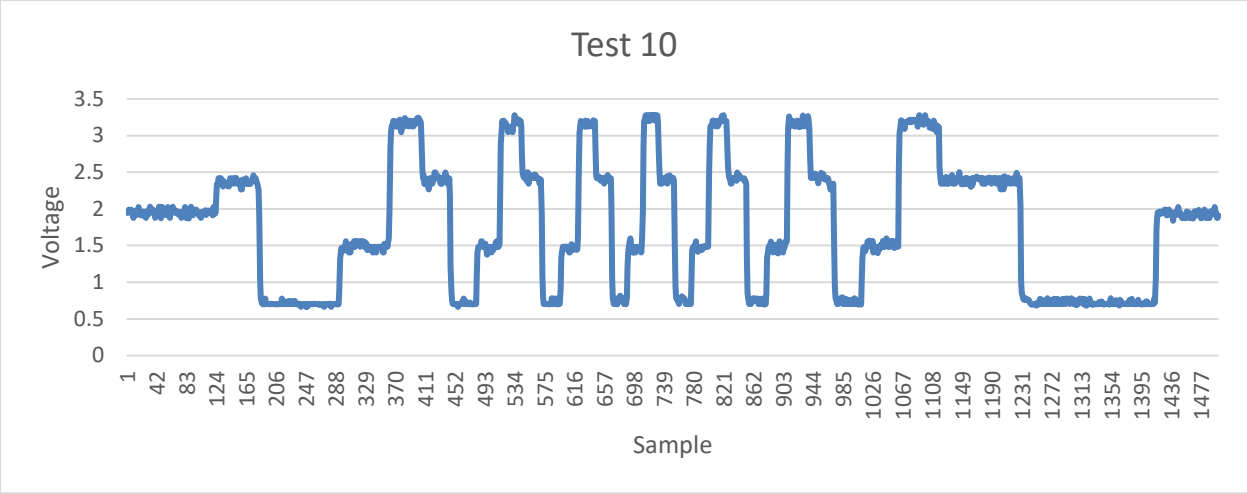

Test 5
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
45
89
133
177
221
265
309
353
397
441
485
529
573
617
661
705
749
793
837
881
925
969
1013
1057
1101
1145
1189
1233
1277
1321
1365
1409
1453
1497
1541
1585
Sample

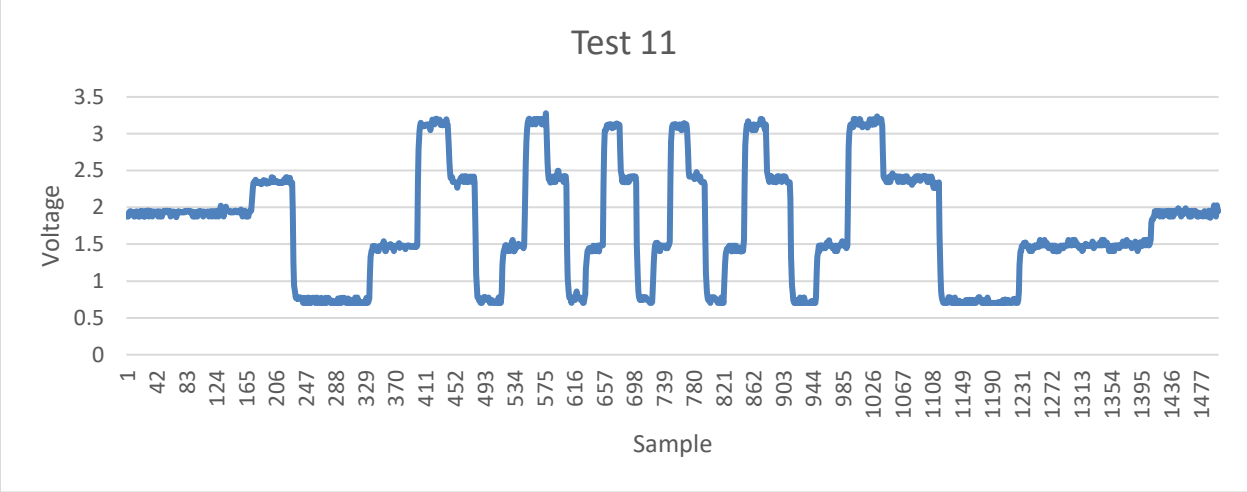

Test 6
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

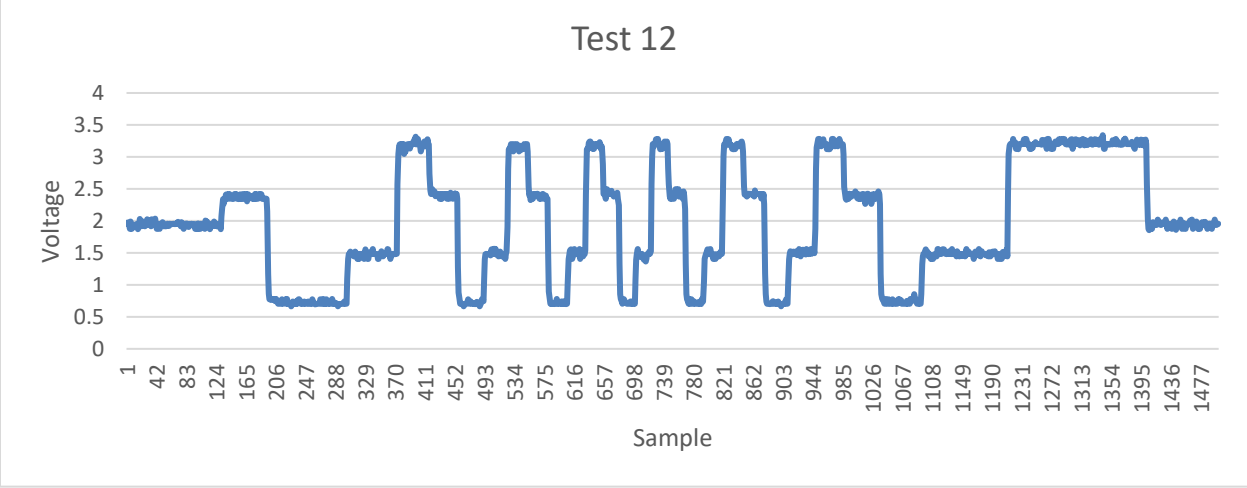

Test 7
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
43
85
127
169
211
253
295
337
379
421
463
505
547
589
631
673
715
757
799
841
883
925
967
1009
1051
1093
1135
1177
1219
1261
1303
1345
1387
1429
1471
1513
Sample

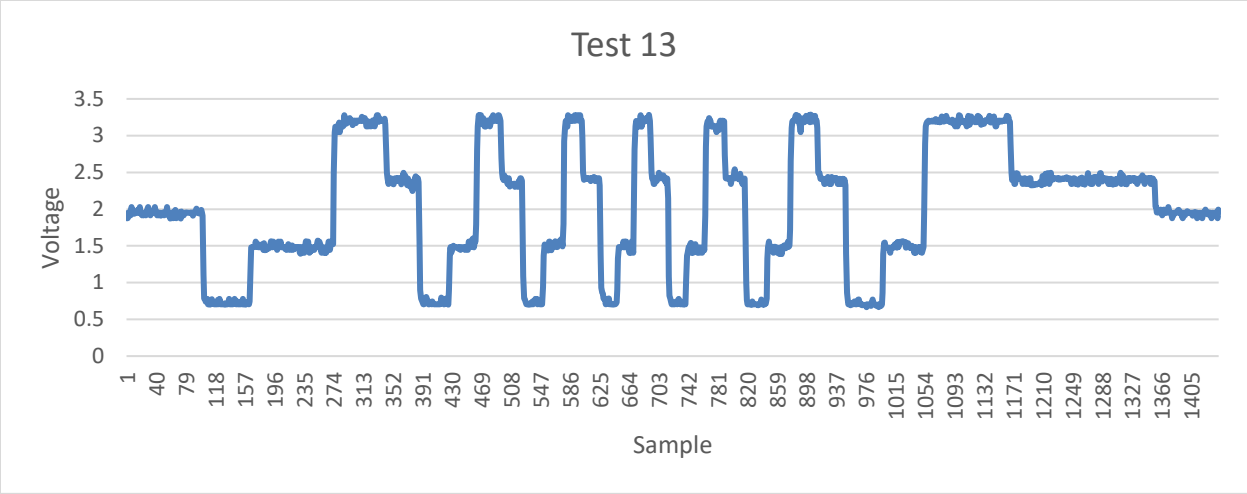

Test 8
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
45
89
133
177
221
265
309
353
397
441
485
529
573
617
661
705
749
793
837
881
925
969
1013
1057
1101
1145
1189
1233
1277
1321
1365
1409
1453
1497
1541
1585
Sample

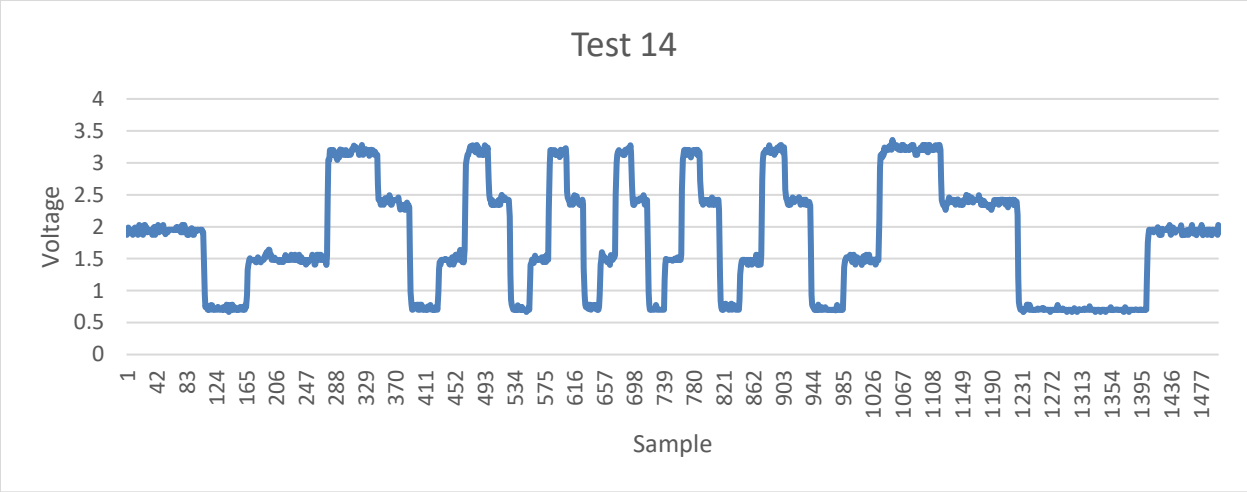

Test 9
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
45
89
133
177
221
265
309
353
397
441
485
529
573
617
661
705
749
793
837
881
925
969
1013
1057
1101
1145
1189
1233
1277
1321
1365
1409
1453
1497
1541
1585
Sample

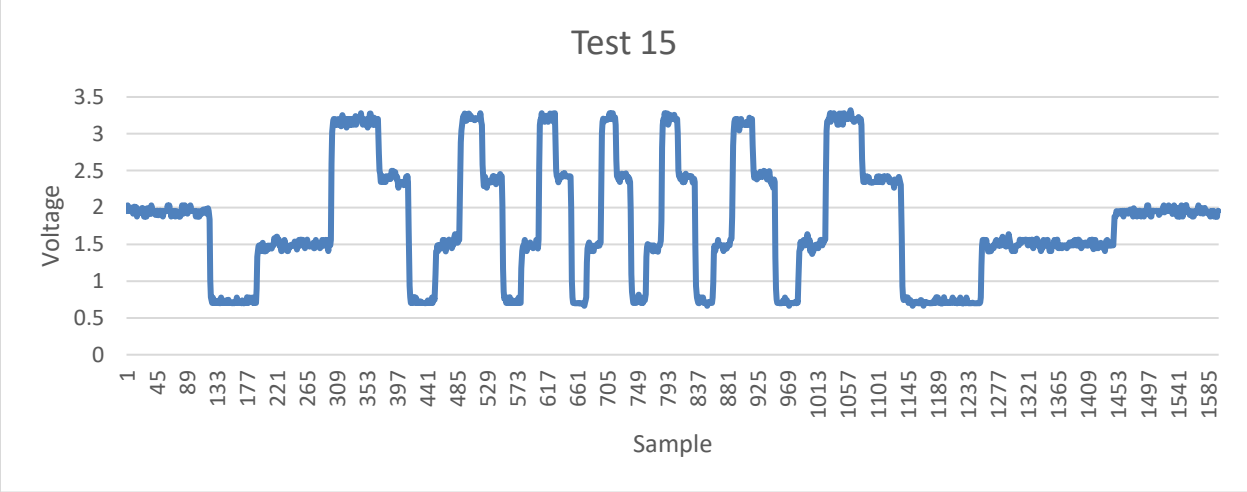

Test 10
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

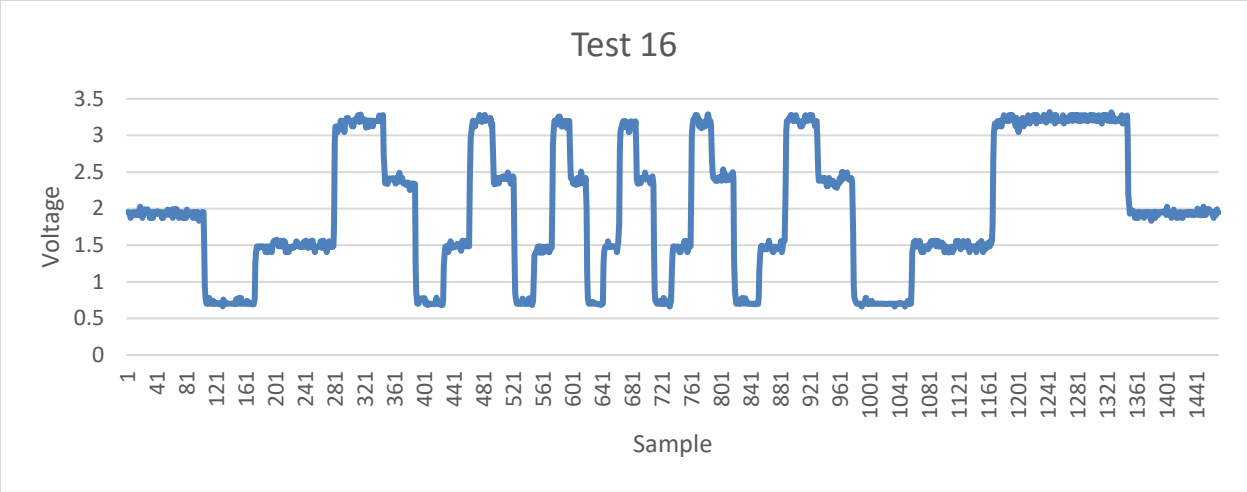

Test 11
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

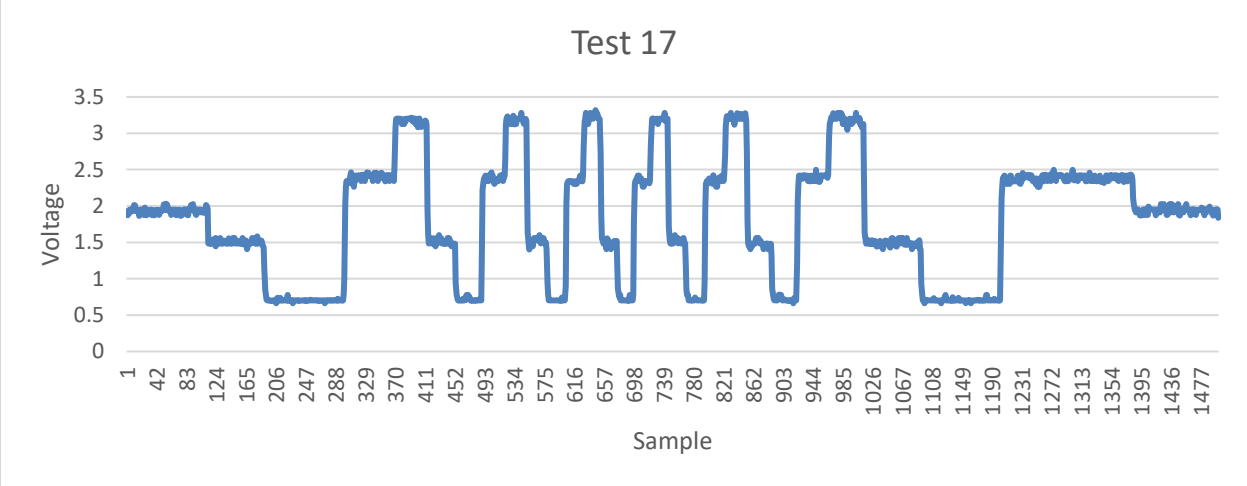

Test 12
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

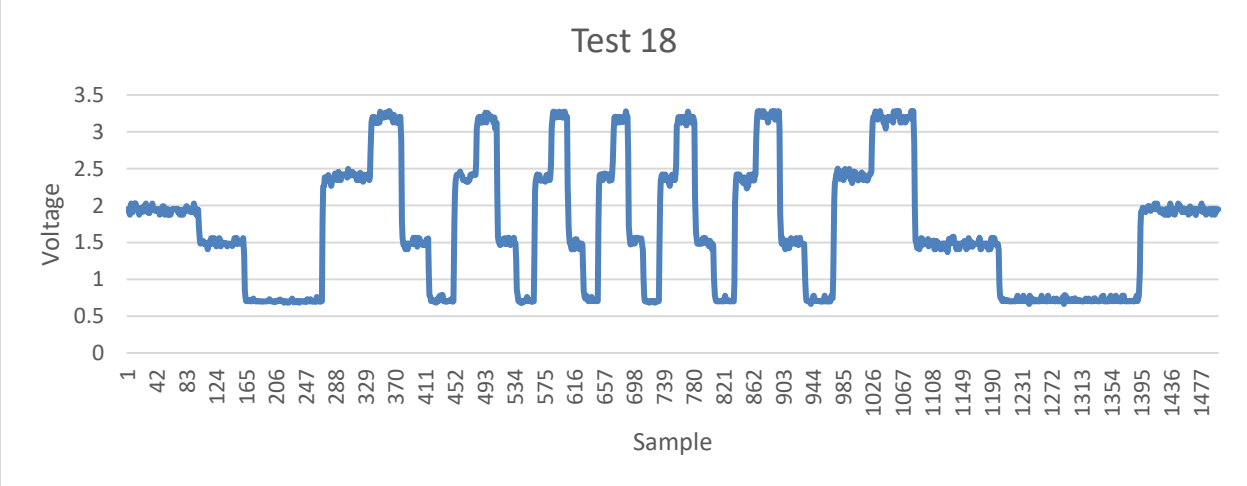

Test 13
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
40
79
118
157
196
235
274
313
352
391
430
469
508
547
586
625
664
703
742
781
820
859
898
937
976
1015
1054
1093
1132
1171
1210
1249
1288
1327
1366
1405
Sample

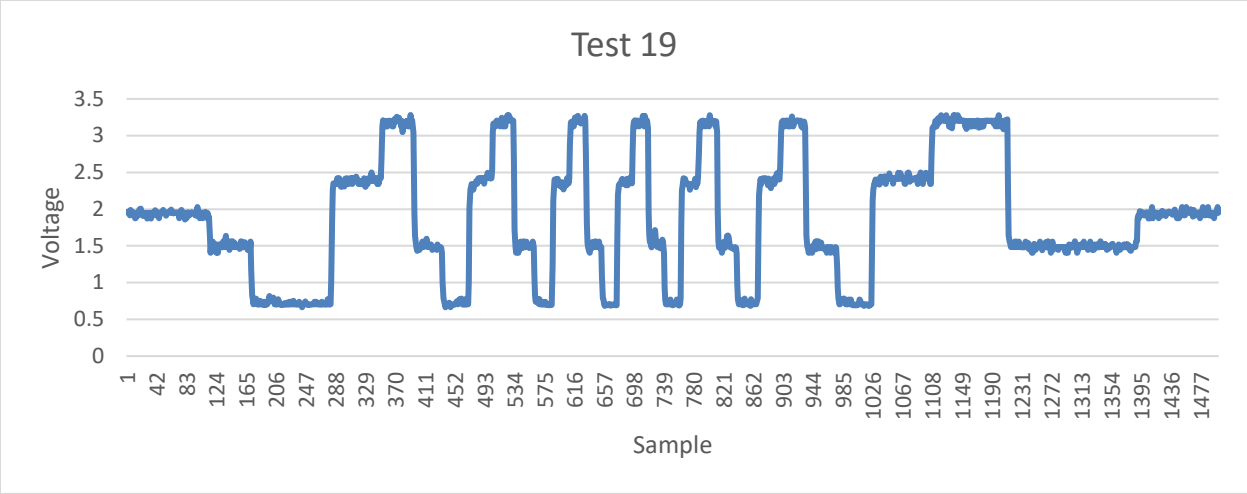

Test 14
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

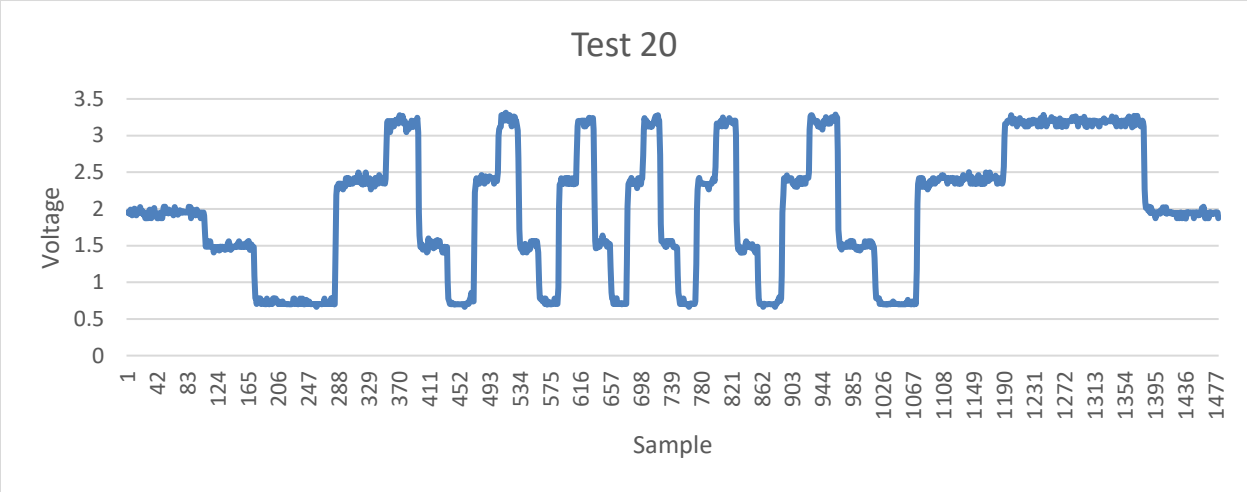

Test 15
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
45
89
133
177
221
265
309
353
397
441
485
529
573
617
661
705
749
793
837
881
925
969
1013
1057
1101
1145
1189
1233
1277
1321
1365
1409
1453
1497
1541
1585
Sample

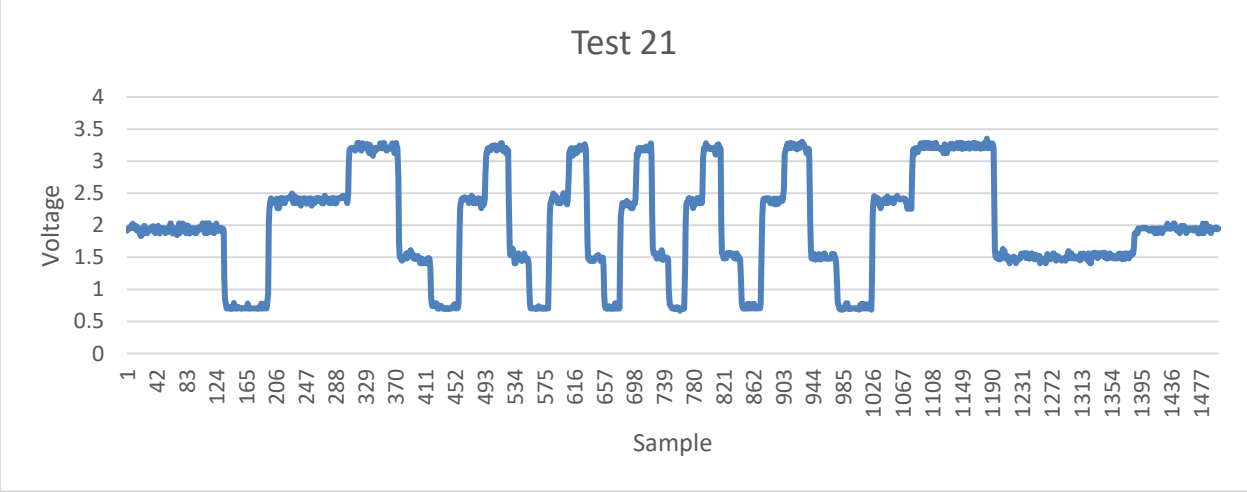

Test 16
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
41
81
121
161
201
241
281
321
361
401
441
481
521
561
601
641
681
721
761
801
841
881
921
961
1001
1041
1081
1121
1161
1201
1241
1281
1321
1361
1401
1441
Sample

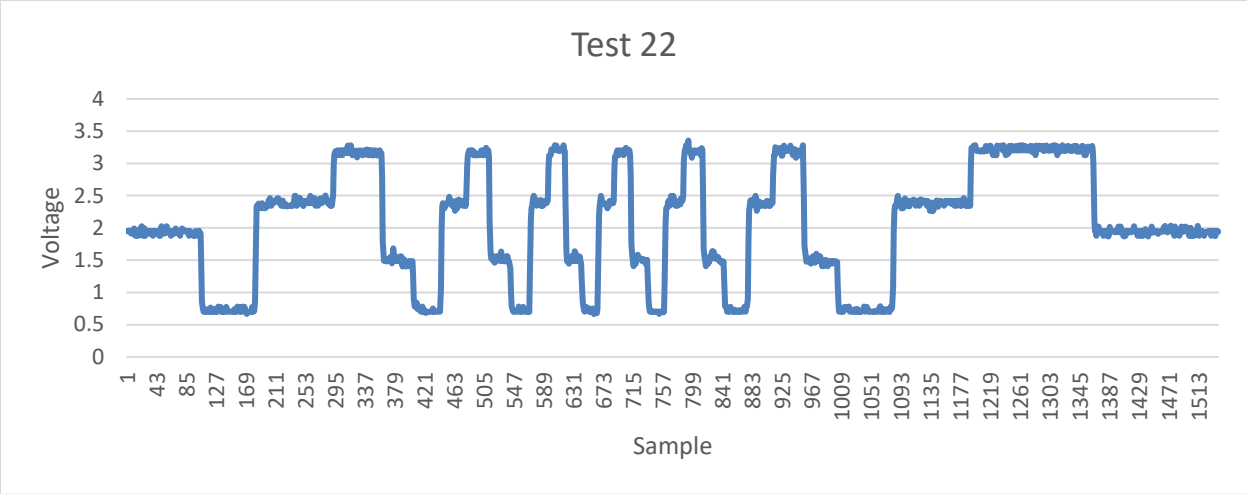

Test 17
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

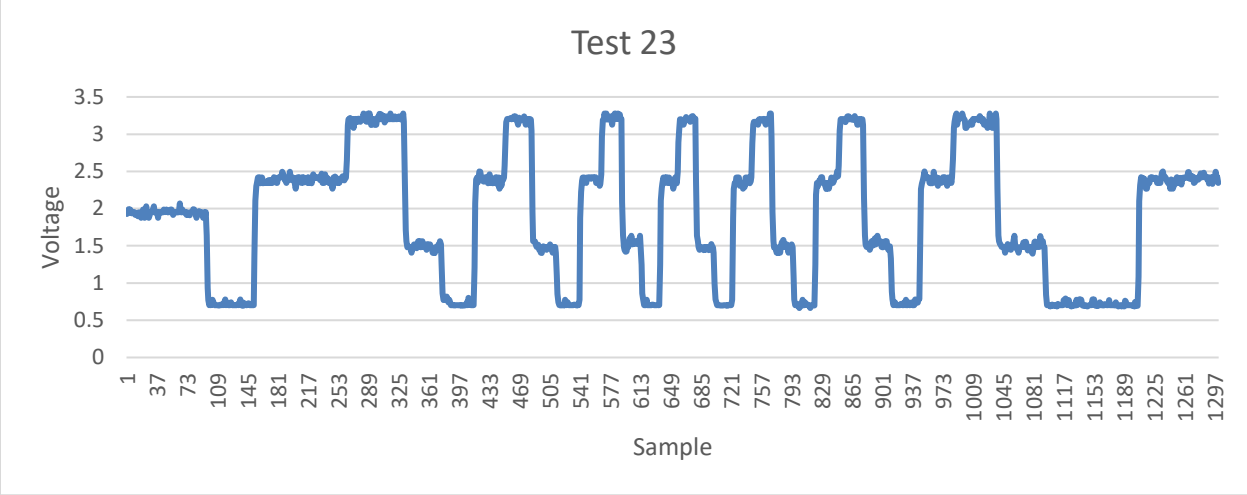

Test 18
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

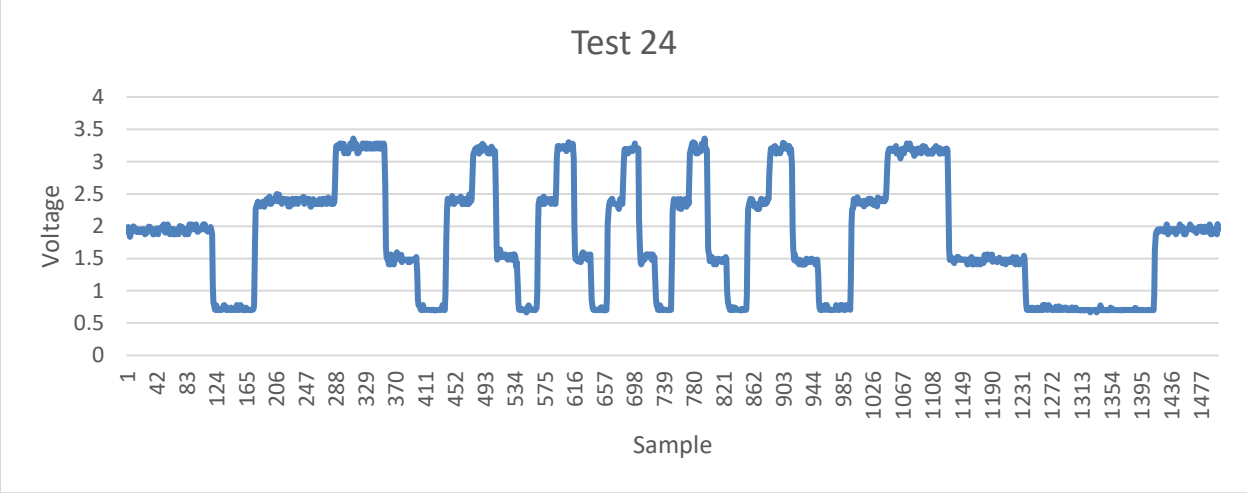

Test 19
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

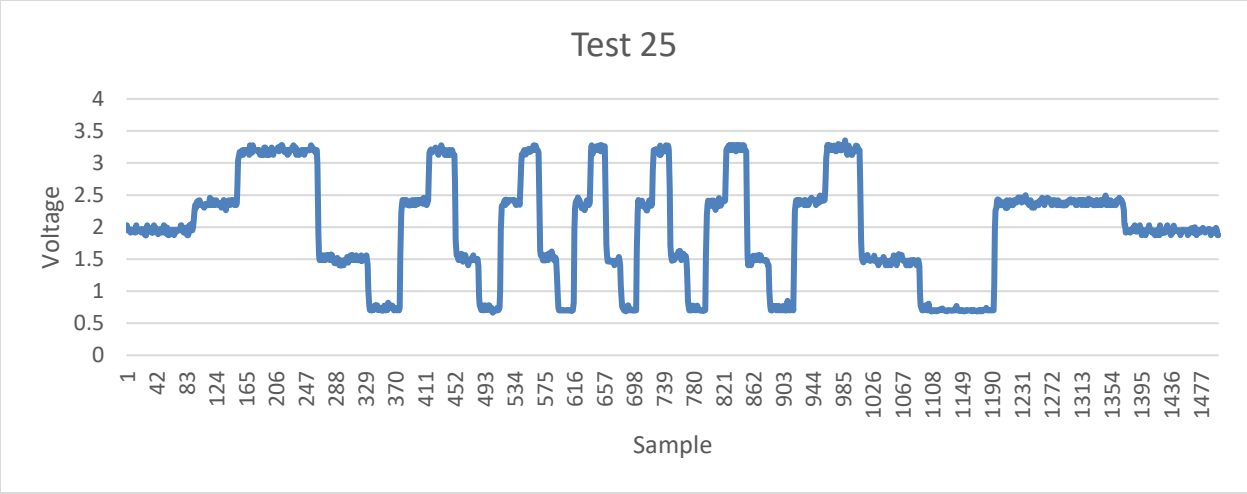

Test 20
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

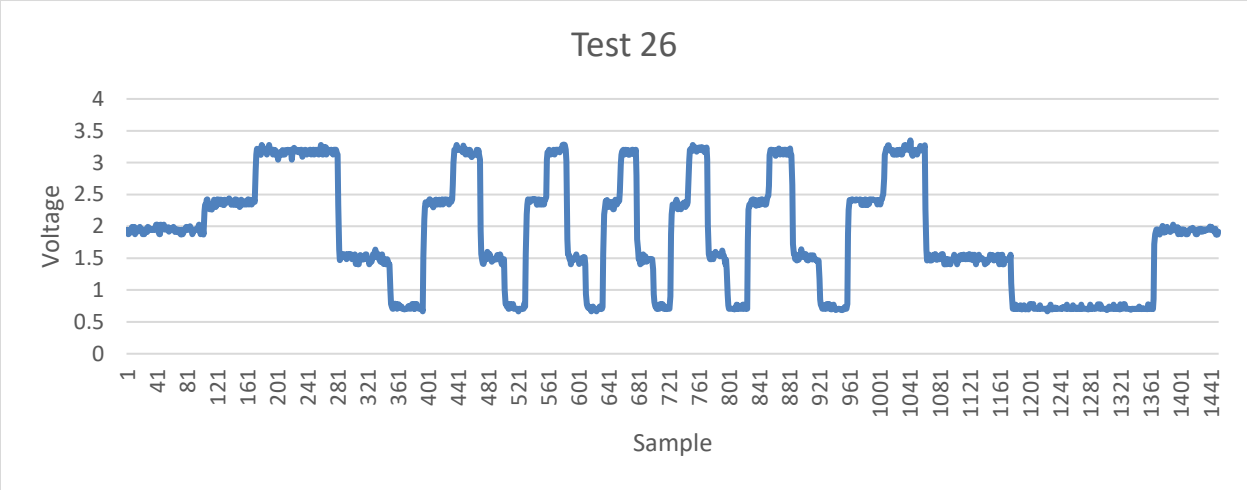

Test 21
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

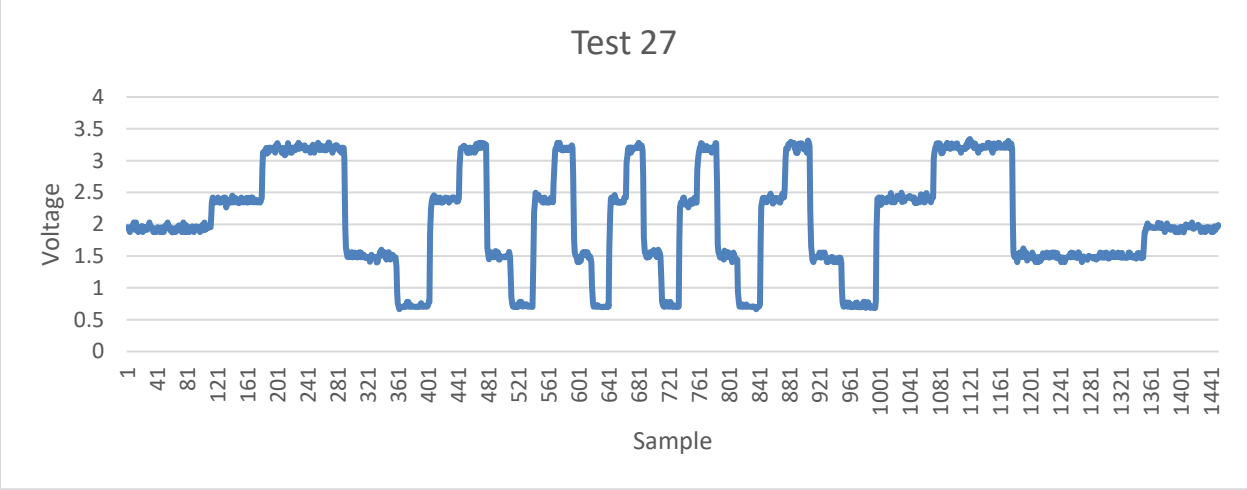

Test 22
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
43
85
127
169
211
253
295
337
379
421
463
505
547
589
631
673
715
757
799
841
883
925
967
1009
1051
1093
1135
1177
1219
1261
1303
1345
1387
1429
1471
1513
Sample

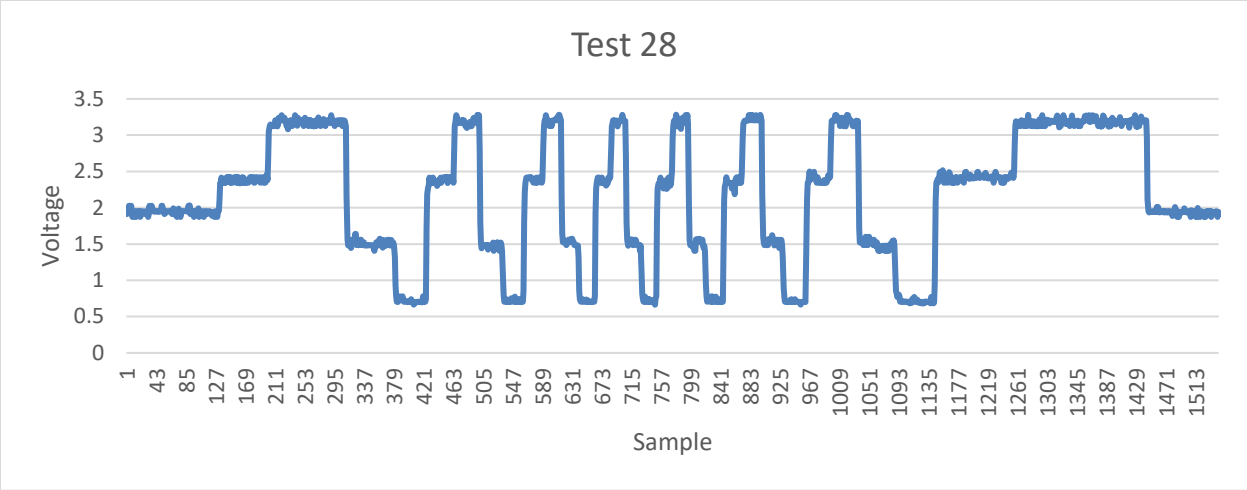

Test 23
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
37
73
109
145
181
217
253
289
325
361
397
433
469
505
541
577
613
649
685
721
757
793
829
865
901
937
973
1009
1045
1081
1117
1153
1189
1225
1261
1297
Sample

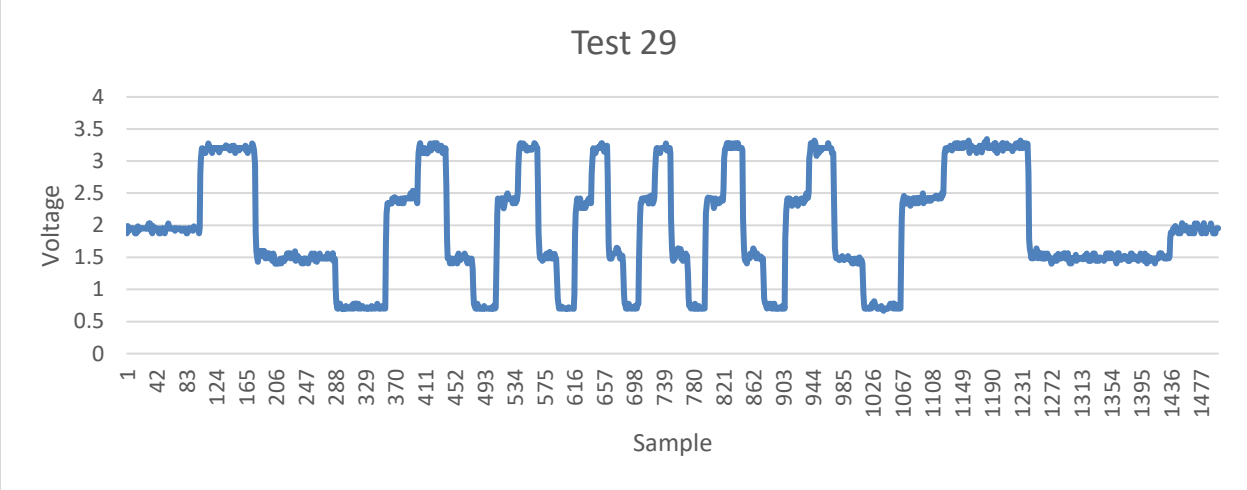

Test 24
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

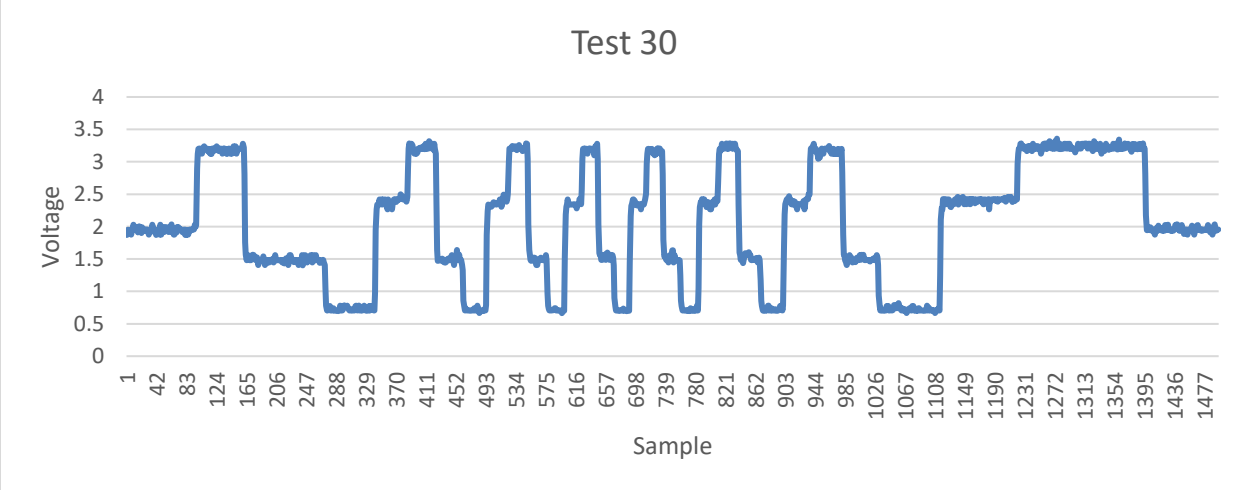

Test 25
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1 42 83 124 165 206 247 288 329 370 411 452 493 534 575 616 657 698 739 780 821 862 903 944 985 1026 1067 1108 1149 1190 1231 1272 1313 1354 1395 1436 1477
Sample

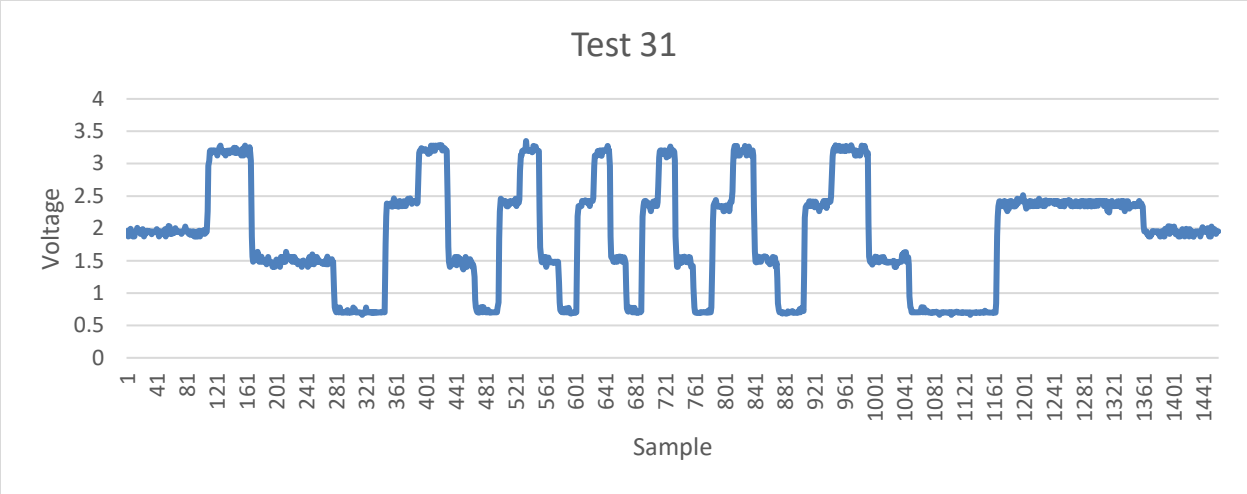

Test 26
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1 41 81 121 161 201 241 281 321 361 401 441 481 521 561 601 641 681 721 761 801 841 881 921 961 1001 1041 1081 1121 1161 1201 1241 1281 1321 1361 1401 1441
Sample

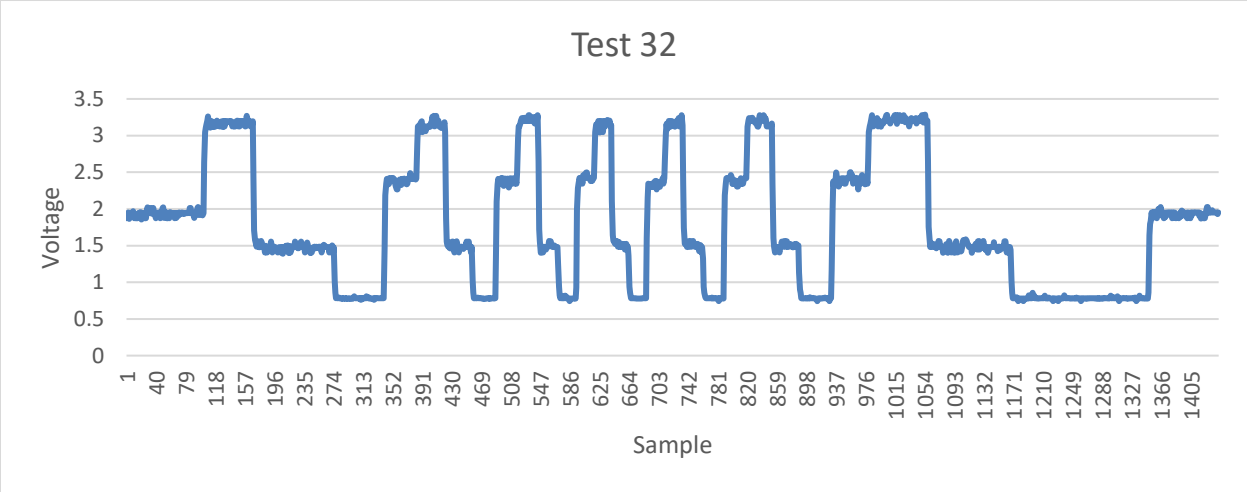

Test 27
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1 41 81 121 161 201 241 281 321 361 401 441 481 521 561 601 641 681 721 761 801 841 881 921 961 1001 1041 1081 1121 1161 1201 1241 1281 1321 1361 1401 1441
Sample

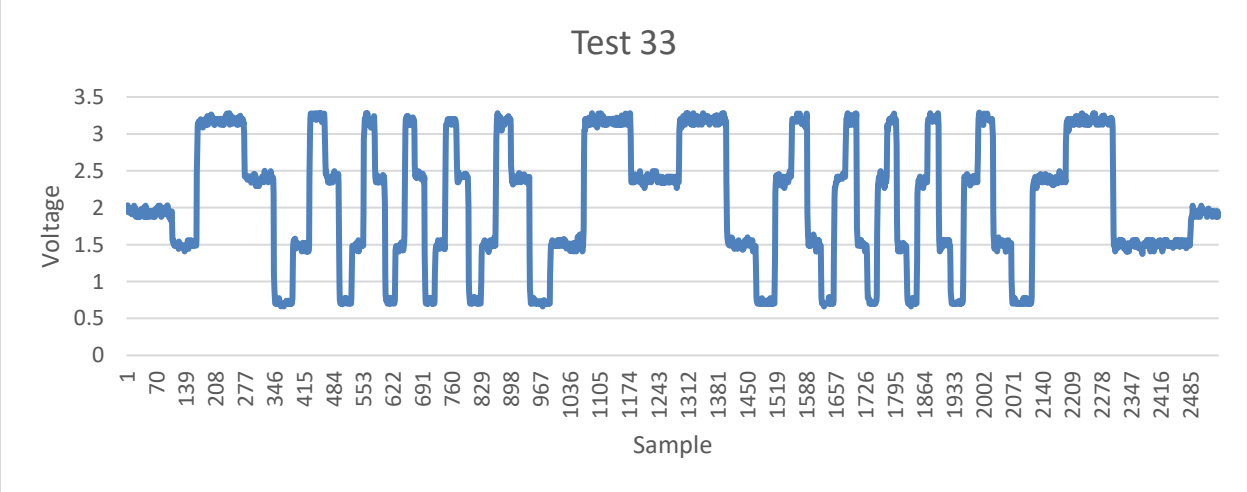

Test 28
Voltage
3.5
3
2.5
2
1.5
1
0.5
0
1
43
85
127
169
211
253
295
337
379
421
463
505
547
589
631
673
715
757
799
841
883
925
967
1009
1051
1093
1135
1177
1219
1261
1303
1345
1387
1429
1471
1513
Sample

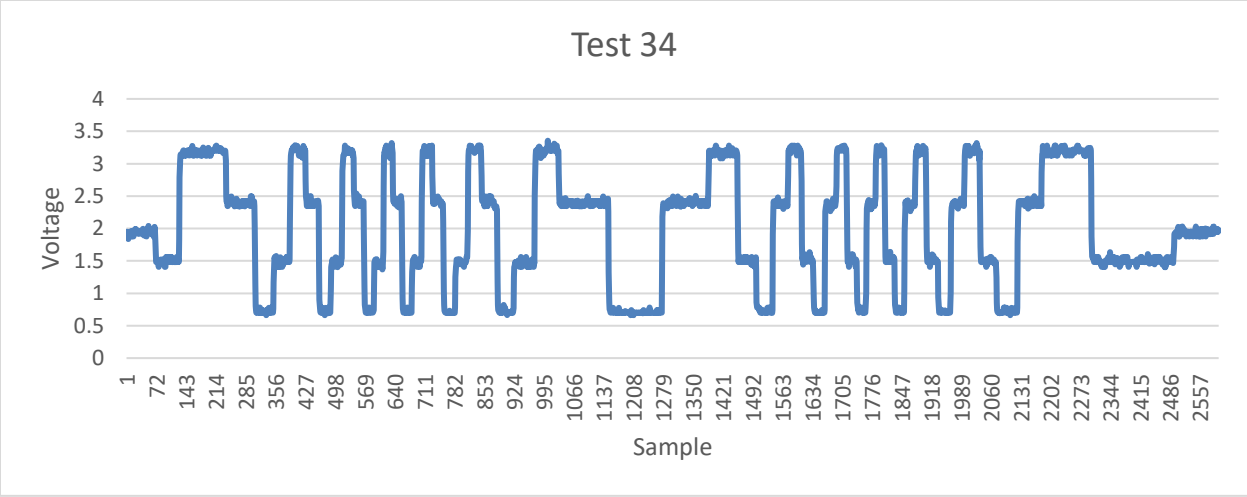

Test 29
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

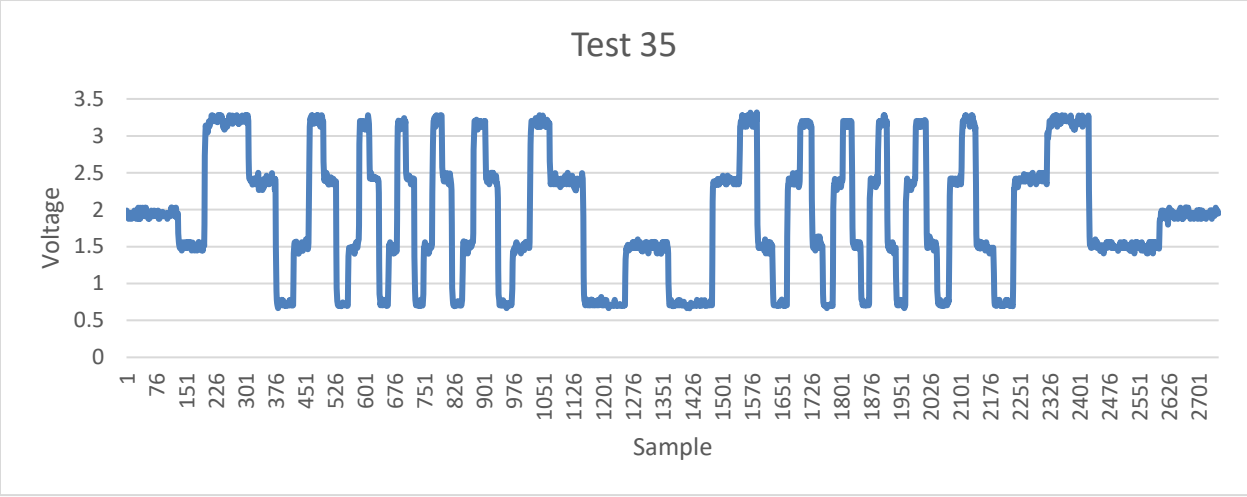

Test 30
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
903
944
985
1026
1067
1108
1149
1190
1231
1272
1313
1354
1395
1436
1477
Sample

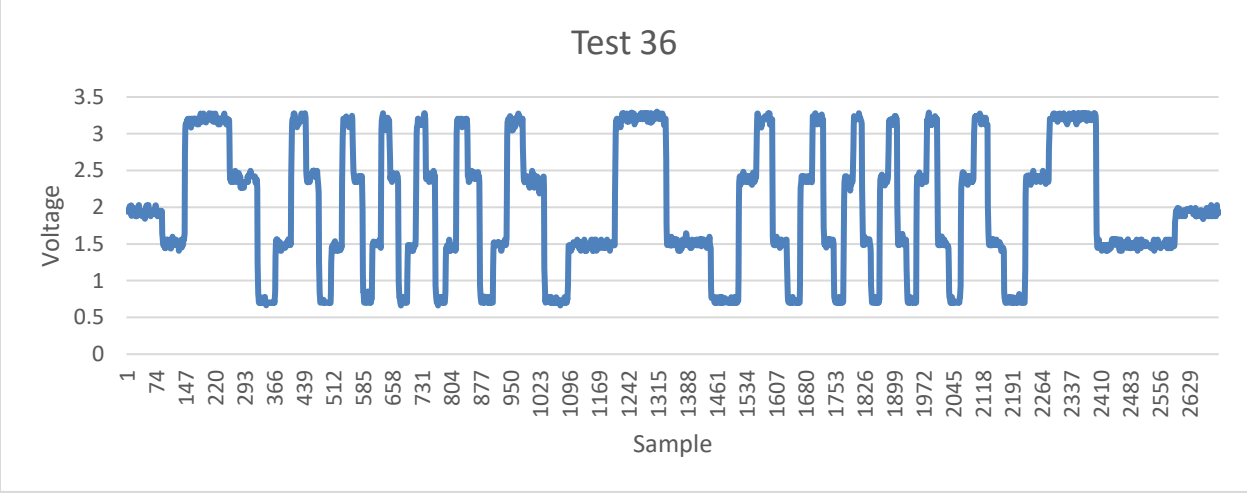

Test 31
Voltage
Sample

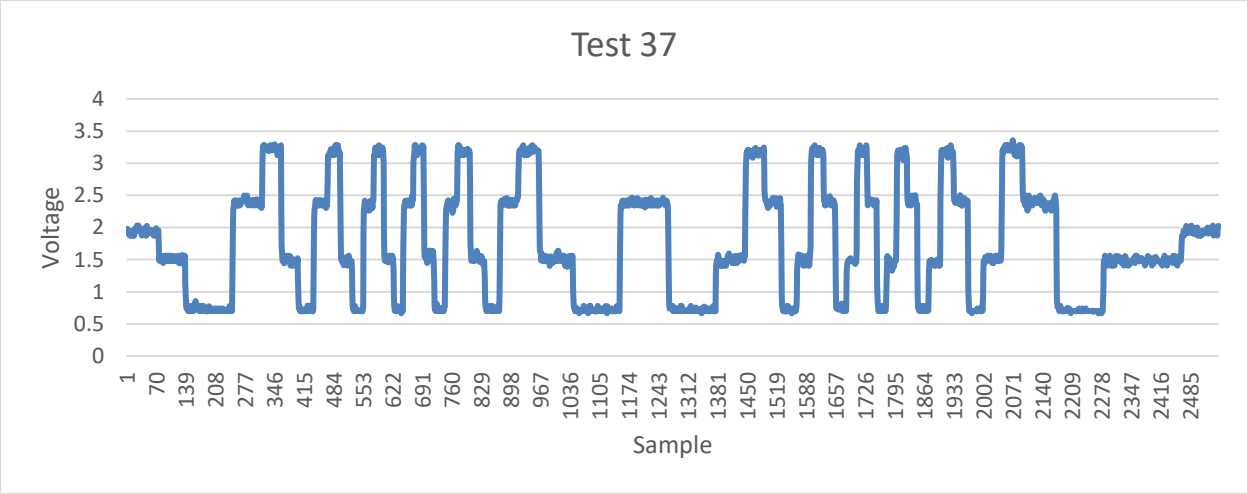

Test 32
Voltage
Sample

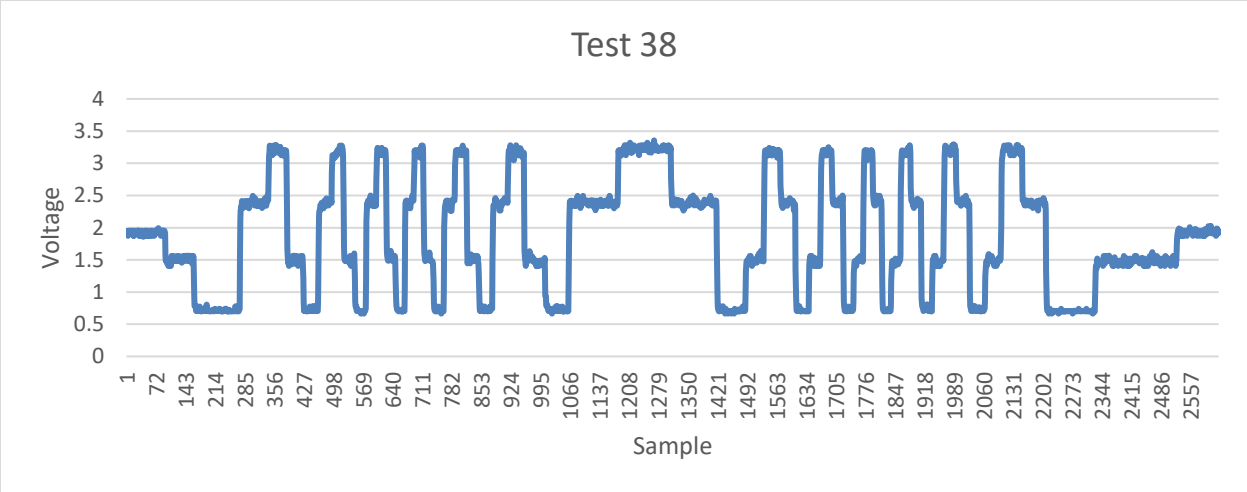

Test 33
Voltage
Sample

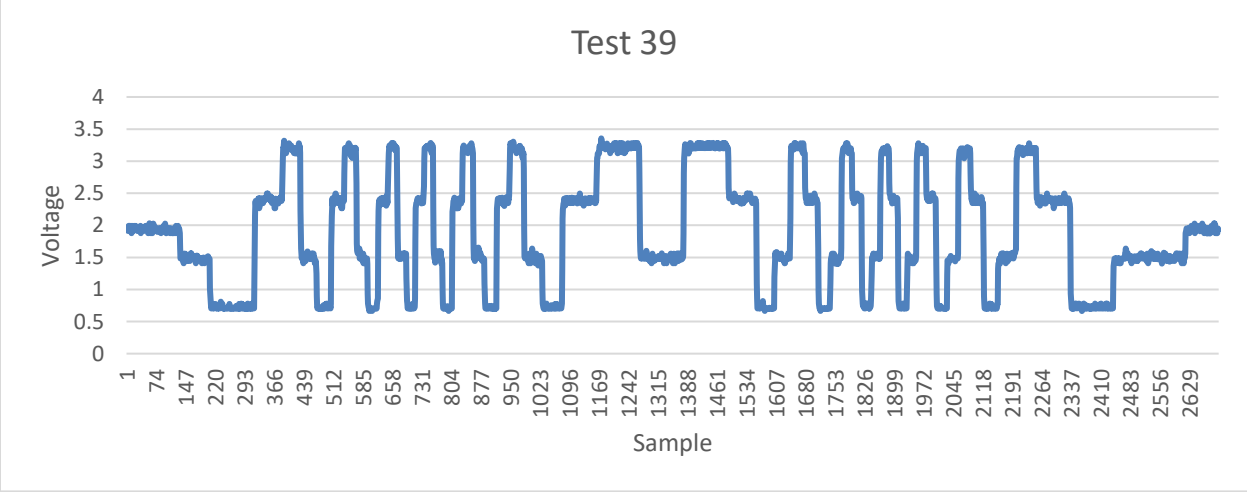

Test 34
Voltage
Sample

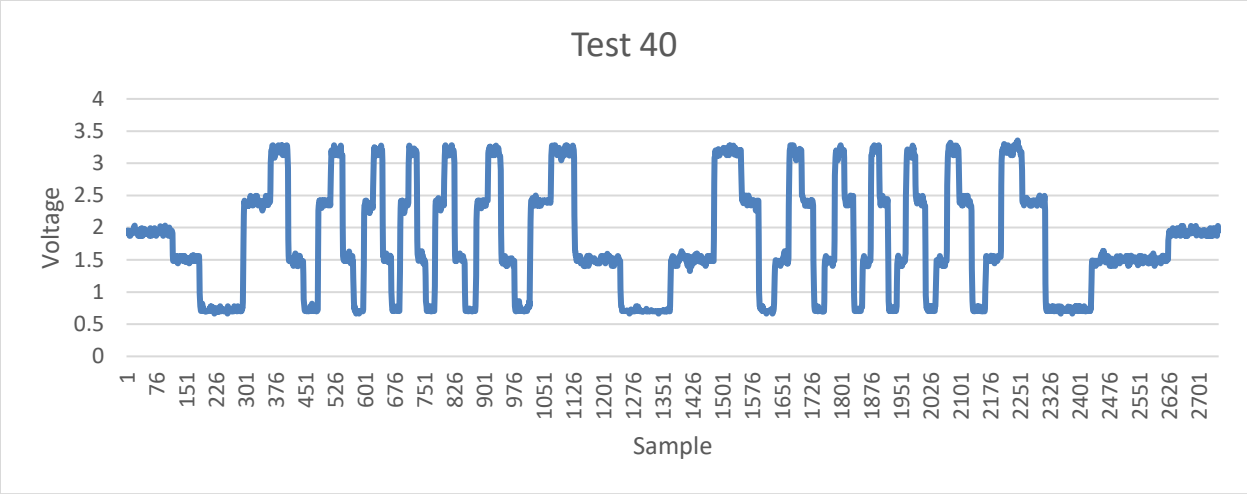

Test 35
Voltage
Sample


Test 36
Voltage
Sample


Test 37
Voltage
Sample
0
0.5
1
1.5
2
2.5
3
3.5
4
1
70
139
208
277
346
415
484
553
622
691
760
829
898
967
1036
1105
1174
1243
1312
1381
1450
1519
1588
1657
1726
1795
1864
1933
2002
2071
2140
2209
2278
2347
2416
2485


Test 38
Voltage
Sample
0
0.5
1
1.5
2
2.5
3
3.5
4
1
72
143
214
285
356
427
498
569
640
711
782
853
924
995
1066
1137
1208
1279
1350
1421
1492
1563
1634
1705
1776
1847
1918
1989
2060
2131
2202
2273
2344
2415
2486
2557


Test 39
Voltage
Sample
0
0.5
1
1.5
2
2.5
3
3.5
4
1
74
147
220
293
366
439
512
585
658
731
804
877
950
1023
1096
1169
1242
1315
1388
1461
1534
1607
1680
1753
1826
1899
1972
2045
2118
2191
2264
2337
2410
2483
2556
2629


Test 40
Voltage
4
3.5
3
2.5
2
1.5
1
0.5
0
1
76
151
226
301
376
451
526
601
676
751
826
901
976
1051
1126
1201
1276
1351
1426
1501
1576
1651
1726
1801
1876
1951
2026
2101
2176
2251
2326
2401
2476
2551
2626
2701
Sample

APPENDIX B: ADC RECORDER CODE

Instructions for recorder code:

The base source code was taken from the maker of the AD converter, Waveshare, found at the link in citation number 32. The file path for the source code is High Precision AD DA Board Code > RaspberryPi > AD-DA > bcm2835. The code is compiled by issuing a "make" command in the terminal and executed by issuing "sudo ./main" in the terminal. Output data can be saved by issuing "sudo ./main > filename.txt" This will output the data to a text file in the bcm2835 folder.

Inside the bcm2835 folder is a folder titled obj. Inside obj is the main file (main.c) which can be replaced with the code under "Main file for recorder code execution". Also in the obj file is a file (ADS1256.c) which consists of several functions. The function displayed below "function for continuous data recording" must be added to these functions. This function is called in the altered main file.

Main file for recorder code execution

```
#include <stdlib.h>     //exit()
#include <signal.h>     //signal()
#include <time.h>
#include <string.h>
#include "stdio.h"

#include "ADS1256.h"
#include "DAC8532.h"

void  Handler(int signo)
{
    //System Exit
    printf("\r\nEND\r\n");
    DEV_ModuleExit();
    exit(0);
}

int main(void)
{
    UDOUBLE ADC[8],i;
    int steps;
    DEV_ModuleInit();
    // Exception handling:ctrl + c
    signal(SIGINT, Handler);

    if(ADS1256_init() == 1){
        printf("\r\nEND\r\n");
        DEV_ModuleExit();
        exit(0);
```

```
    }
    while(1){
        int steps = ADS1256_GetChannalValueCont(7);  //Call continuous read
function
        break;
    }
    return 0;
}
```

Function for continuous data recording

```
/************************************************************************
**
function:  Read ADC data continuously
parameter:
Info:
************************************************************************
*/
static UDOUBLE ADS1256_Read_ADC_Data_Cont(void)
{
    int i;                                  //Initialize Counter
    UDOUBLE read = 0;                       //Initialize Read
    UBYTE buf[3] = {0,0,0};                 //Initailize Buffer
    float timestamp;                        //Initialize Timestamp Variable
    ADS1256_WaitDRDY();                     //Wait for DataReady Pin to go low
    DEV_Delay_ms(1);                        //Delay
    DEV_Digital_Write(DEV_CS_PIN, 0);     //Set Pin Low
    DEV_SPI_WriteByte(CMD_RDATAC);        //Issue Continuous Data Read
Command
    DEV_Delay_ms(1);                        //Delay
    for (i=1; i<=10000; i++)      //For loop where i value is samples taken
    {
    read = 0;
    UBYTE buf[3] = {0,0,0};
    DEV_Delay_micro(167);                   //Delay 167 microseconds
    ADS1256_WaitDRDY();                     //Wait for Dataready
    buf[0] = DEV_SPI_ReadByte_Cont(); //Read first eight digits of conversion
    buf[1] = DEV_SPI_ReadByte_Cont(); //Read second eight digits of
                                      conversion
    buf[2] = DEV_SPI_ReadByte_Cont(); //Read third eight digits of conversion
    timestamp = clock();                  //Note clock time
    read = ((UDOUBLE)buf[0] << 16) & 0x00FF0000; //Shift first eight digits
    read |= ((UDOUBLE)buf[1] << 8);  ///* Pay attention to It is wrong   read
                                     //|= (buf[1] << 8) *///shift second
                                    //eight digits
    read |= buf[2];                   //Set third eight digits
    printf("%f %f\r\n", read*5.0/0x7fffff, timestamp/CLOCKS_PER_SEC);
                                    //Record reading and timestamp

    }
    DEV_Digital_Write(DEV_CS_PIN, 1);     //Set CS Pin low
}
```

APPENDIX C: MATLAB INTERPRETATION CODE

```
function g_code_writer(data)
%The main function for writing out g-code consists of two main loops with
%several sub functions.  Each loop starts at the beginning of the
%voltage data and reads line by line. At each voltage the state of the
%motor is checked.  This motor state corresponds to a voltage state defined
%by values 1-5 corresponding to the five states of the motor (HH, LH, LL,
%HL, off). The first loop uses the voltage states to determine direction.
%The second loop uses the voltage states to count the steps the motor has
%taken.  At the end of each section of data corresponding to a single line
%of g-code, the line of g-code is printed.

global v v_data v_diff i i_old v_ave
global direction direction_known direction_trigger
global steps state state_known step_state step_state_known state_old state_old_dir
global v_low v_high v_mid_low v_mid_high
global y_loc steps_line mmperstep
%initiate variables%
direction = 0;                      %initial direction set to 0 trigger direction finder
direction_trigger = 0;              %direction trigger starts direction finder
v_low = 1.15;                       %Low Voltage threshold for state identification
v_high = 2.9;                       %High voltage threshold for state identification
v_mid_low = 1.85;                   %Mid-low voltage for state identification
v_mid_high = 2.3;                   %Mid-high voltage for state identification
state_known = 0;                    %variable for if the state of the motor is identified
steps = 0;                          %initiate steps at zero
steps_line = 0;                     %initiate steps per line of g-code at zero
y_loc = 0;                          %initiate y location at zero
step_state_known = 0;               %variable for if the state of the motor for step counting is known
state = 0;                          %initiate state of the motor at zero
direction_known = 0;                %variable for if motor direction is known
mmperstep = 0.004;
%import the text data and identify length%
v_data = importdata(data,' ');          %import voltage data from a text file
samples = length(v_data);               %find the number of data samples taken
%loop to determine direction%
%The direction determination loop runs through each voltage in the data and
%checks the state of the voltage (1-5) corresponding to the five states of
%the motor (HH, LH, LL, HL, off).  It does not check the state in a
```

```
%transition between voltages.  This is determined by checking the standard
%deviation for twenty points surroudning the current voltage.  Once the
%state has been identified, the direction identifier can be run.  The
%direction identifier needs two consecutive states to determine direction.
%The state_old_dir is the variable for storing the previous state.
for i = 11:samples-10                    %starts at 11 to allow for stddev calculation
v = v_data(i,1);                         %identify voltage at desired instance
v_diff = std(v_data(i-10:i+10,1));       %calculate stddev for 20 samples surrounding
v_ave = mean(v_data(i-10:i+10,1));       %calculate average for 20 samples surrounding
if (v_mid_low < v) && (v < v_mid_high)  %if the voltage is in the "stopped" threshold the code returns to
the top of the for loop
    continue
end
if (v_diff < 0.1)                        %if the stddev is below the threshold, it performs functions
    state_old_dir = state;               %the state identifier for the direction detection is updated
    state_identifier()                   %the state identifier function is run
else                                     %if the stddev is too high, the code does not check the state
    continue
end
if (direction == 0) && (state_old_dir <= 4) && (state_old_dir >= 1)         %If the direction is unknown
                                                                              and the state has been
                                                                              identified
    direction_identifier()         %the direction identifer is run
elseif (direction_known == 0)      %if the direction is unknown and the state has not been identified, the
                                     code returns to the top of the for loop
    continue
else
    break                                    %The for loop ends when the direction has been identified
end
end
state = 0;           %The state is reset at zero for the counting loop
%step counting loop%
%The step counting loop operates similarly to the direction determination
%loop.  Two state variables are used.  State the state as defined before.
%Step_state is a variable for motor state used for counting steps.
%The state is used to set the initial value of step_state, so initially
%they match, but as the motor speeds up, the state calculator
%would not be able to count the states properly, so they may differ at full
%speed.  After each step is counted t
for i = 11:samples-10                    %starts at 11 to allow for stddev calculation
v = v_data(i,1);                         %identify voltage at desired instance
```

```
v_diff = std(v_data(i-10:i+10,1));      %calculate stddev for 20 samples surrounding
v_ave = mean(v_data(i-10:i+10,1));      %calculate average for 20 samples surrounding
if (v_diff < 0.1)                       %if the stddev is below the threshold, the state is
                                        %not transitioning and can be read reliably
    if (i-i_old) > 30                   %if the state has not been read in 30 data points
        state = 0;                      %the motor has entered a rapid step pattern and the old
    end                                 %state information is lo longer accurate

    state_old = state;                  %The previous state is recorded
    state_identifier()                  %state identifier function is run
    i_old = i;
end

if (step_state_known == 0) && (state ~=10)       %if the step state is known and the state is not state 10
    step_state = state;                                  %step state is set to the known state
    step_state_known = 1;                                %step state is known
end

if (direction == 1) && (step_state_known == 1)       %if the direction detected is forward the forward step
                                                      counter is run
    step_counter_forward()
end
if (direction == 2) && (step_state_known == 1)       %if the direction detected is reverse the
                                                     reverse step counter is run
    step_counter_reverse()
%    step_counter_reverse_skip()
end
check_state()                                          %the state checker function is run%
end
end

%The state identifier function checks the average voltage of 20 points
%surrounding the current data point.  The average voltage will fall within
%one of the five states.  The five states cover the entire range of the
%voltage output.  Once the state has been identified, the code records the
%state as having been identified
function state_identifier()
global v_mid_high v_high v_mid_low v_low state state_known v_ave
if (v_mid_high < v_ave) && (v_ave < v_high)       %voltage range for upper middle threshold
    state = 1;                                   %state set to 1
```

```
    state_known = 1;                                %the state is known so state_known is set to 1
end
if (v_ave < v_low)                                  %voltage range for lower threshold
    state = 2;
    state_known = 1;
end
if (v_low < v_ave) && (v_ave < v_mid_low)           %voltage range for lower middle threshold
    state = 3;
    state_known = 1;
end
if (v_high < v_ave)                                 %voltage range for lower threshold
    state = 4;
    state_known = 1;
end
if (v_mid_low < v_ave) && (v_ave < v_mid_high)      %voltage range for middle threshold of no motion
    state = 10;                                     %state value of 10 is used to aid direction determination
end
end

%The direction identifier function checks the difference between the old
%state and the new state to determine the sequence of states.  Once the
%sequence of two consecutive states is known, direction can be determined.
%If the motor is moving forward, the states will progress 1-2-3-4 so the
%difference between the old and new will either be 1 or -3 (1-4).  The
%opposite occurs for reverse motor motion.  Once the direction has been
%identified, the function signals that the direction is known and the
%direction identifier is no longer necessary.
function direction_identifier()
global  state direction direction_known state_old_dir

if ((state - state_old_dir) == 1) || ((state - state_old_dir) == -3)
    direction = 1; %forward
    direction_known = 1;
end
if ((state - state_old_dir) == -1) || ((state - state_old_dir) == 3)
    direction = 2; %reverse
    direction_known = 1;
end
end
```

```
%The forward step counter function follows a simple progression.
%Step_state is a variable used similar to the state for counting steps,
%except it is defined by fewer parameters and changes each time a new step
%is recorded.  When the state is identified the step state is set to match
%the state.  As soon as the voltage passes the threshold to move to the
%next state a step is counted and the step state is redefined.  Steps are
%the total number of steps in the entire sequence of g-code commands.
%Steps_line is the number of steps in one g-code command.  These steps are
%necessary in determining the distance traveled for each step.  Voltage
%thresholds are set so that the maximum and minimum voltage recorded at
%high speed will always cross the threshold and trigger the new state.  The
%step state sequence follows 1-2-3-4-1...
function step_counter_forward()
global v steps step_state steps_line
v_step_low = 1.3;                           %Lower voltage threshold
v_step_high = 2.7;                          %Upper voltage threshold
if (step_state == 4) && (v < v_step_high)   %If the voltage drops below the upper threshold, state 4 has
been left
    steps = steps + 1;                      %steps are incremented
    steps_line = steps_line + 1;            %steps_line are incremented
    step_state = 1;                         %The step state is changed
end
if (step_state == 1) && (v < v_step_low)
    steps = steps + 1;
    steps_line = steps_line + 1;
    step_state = 2;
end
if (step_state == 2) && (v > v_step_low)
    steps = steps + 1;
    steps_line = steps_line + 1;
    step_state = 3;
end
if (step_state == 3) && (v > v_step_high)
    steps = steps + 1;
    steps_line = steps_line + 1;
    step_state = 4;
end
end

%The step counter reverse function works identically to the forward
%version, but the sequence of step states is reversed.
```

```
function step_counter_reverse()
global v steps step_state steps_line
v_step_low = 1.3;
v_step_high = 2.7;
if (step_state == 4) && (v < v_step_high)
    steps = steps + 1;
    steps_line = steps_line + 1;
    step_state = 3;
end
if (step_state == 3) && (v < v_step_low)
    steps = steps + 1;
    steps_line = steps_line + 1;
    step_state = 2;
end
if (step_state == 2) && (v > v_step_low)
    steps = steps + 1;
    steps_line = steps_line + 1;
    step_state = 1;
end
if (step_state == 1) && (v > v_step_high)
    steps = steps + 1;
    steps_line = steps_line + 1;
    step_state = 4;
end
end

%The check state function is called after each data point is read and
%serves to identify transitions between lines of code and the end of the
%code.  If the direction is known the sequence of steps is known.  If the
%sequence recorded does not match the sequence expected, the end of a line
%of code has been reached and the motor has reversed direction or completed
%the sequence of instructions.  If this case is reached, the steps are
%tranlated into a distance traveled and the line of g-code which commanded
%those steps is printed out.  The steps_line is reset for the next line and
%the program continues to the next command or the end of the dataset.
function check_state()
global state state_old direction y_loc steps_line step_state_known mmperstep
if (direction == 1) && (((state - state_old) == -1) || ((state - state_old) == 3)) && (state_old ~=0) %The
state is checked to verify the end of a line
    if (state_old == 2) || (state_old == 4)  %If the old state was one of the peaks or troughs of the
voltage level and extra step is incorrectly counted
```

```
        steps_line = steps_line - 1;          %The extra step is removed
    end
    y_dist = steps_line*mmperstep;                %The distance traveled is calculated based on the number
                                                      of steps and the distance per step.
    y_loc = y_loc + y_dist;                   %The new location is determined by adding the new distance to
                                                  the old location.
    fprintf('G1 Y%f\n',y_loc)                 %The G code instruction is printed
    %fprintf('Steps in Line %i\n',steps_line)
    step_state_known = 0;                     %The step state is reset so it can be properly determined by
                                                  the state determination
    steps_line = 1;                           %The steps per line is set to 1, because for the transition to
                                                  occur, the first step in the new sequence must occur
    direction = 2;                            %The direction is changed, but the direction identifier is not
                                                  needed
end
%The following if statement is the same as above, except the new y location
%is determined by subtracting the y distance because the travel is in the
%negative direction.  The direction is changed to forward as well.
if (direction == 2) && (((state - state_old) == 1) || ((state - state_old) == -3)) && (state_old ~=0)
    if (state_old == 2) || (state_old == 4)
        steps_line = steps_line - 1;
    end
    y_dist = steps_line*mmperstep;
    y_loc = y_loc - y_dist;
    fprintf('G1 Y%f\n',y_loc)
    %fprintf('Steps in Line %i\n',steps_line)
    step_state_known = 0;
    steps_line = 1;
    direction = 1; %forward
end
%The followin if statement occurs at the end of the step sequence, when the
%voltage state returns to no motion (state 10).  The same sequence of
%calculations and lines printed is followed as before.
if (state == 10) && (state_old <= 4) && (state_old >= 1)
    if (state_old == 2) || (state_old == 4)
        steps_line = steps_line - 1;
    end
    y_dist = steps_line*mmperstep;
    if (direction == 1)
        y_loc = y_loc + y_dist;
    end
```

```
    if (direction == 2)
        y_loc = y_loc - y_dist;
    end
    y_loc = round(y_loc,3);
    %fprintf('steps %d\n',steps)
    fprintf('G1 Y%f\n',y_loc)
    %fprintf('Steps in Line %i\n',steps_line)
    steps_line = 0;
    step_state_known = 0;
    direction = 0;

end

end
```