THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF MECHANICAL ENGINEERING

ANALOG TWINS: DEVELOPMENT OF WIRELESSLY COMMUNICATING ROBOTIC TWINS WITH MULTIPLE DEGREES OF FREEDOM

NATHANIEL D. GRAHAM
SPRING 2019

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Biomedical Engineering and Mechanical Engineering
with honors in Mechanical Engineering

Reviewed and approved* by the following:

Aman Haque
Professor of Mechanical Engineering
Thesis Supervisor

Stephanie Stockar
Assistant Professor of Mechanical Engineering
Honors Adviser

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

Digital twinning is a concept which takes a physical object and collects phenomenon from the object, importing that information into a simulation, optimizing behavior in simulation, and returning it for better performance. The question which follows is what if one changes the purpose of digital twinning into purely movement copying and creates two independent objects that move together wirelessly?

The objective of this study is to demonstrate the digital twin concept using two robotic objects that could mirror each other via a wireless connection and establish a protocol for such. Arduino Unos and nrf24l01+ transceiver modules are used in this study to create the connection and Arduino code, a derivative of C/C++, is developed to collect movement readings of a 2dof system. Two robots from VEX, specifically their Clawbot, with four degrees of freedom each are used. Original', the robot given a movement command first, acts as a parent and potentiometers are used to collect data which are then wirelessly sent to Copy, which is the child. 'Copy' is programmed to receive and move to reproduce the movements of 'Original' in real time based on the measurements collected and transmitted wirelessly. This study tests the reproduction of movement across two degrees of freedom, one in the arm movement of the robots and one in the claw movement of the robots.

This study presents a preliminary proof of concept where Copy is capable of mirroring the movements of Original on multiple degrees of freedom. The movements are delayed by seconds but accurate in position. They can be improved by further investigating movement delays and mapping functions to compensate time delays and relative movement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thank you to my roommates Noah Yoskowitz, Andres Ruiz, Kyle Steen, Collin Hensley, Laith Yousif, and Matthew Auerbach who deserve acknowledgement for allowing me to set up my research and robots in our living room while working on them. For every time I left them out on our dining room table and left the apartment a mess because of a full night of working, or whatever it was I was doing, for being interested enough to ask questions, and unknowingly push me to finish my thesis.

My thanks reach out to my club, Club Gymnastics, and the club sports team here at Penn State. They gave me a place to go and physically let myself destress or exhaust myself to a point that I was content. To all the nights out of state at competitions, every Nationals, every movie night, brunch, and time we spent together. I wouldn't have loved college as much without it.

Thank you to my parents, Ann and Scott Graham, and my girlfriend, Taryn Ryan, for being a source of sanity during the last year and a half when things would work one day and then the exact same thing wouldn't the next. The sessions of griping, confusion, and exhaustion are much appreciated.

I thank VEX robotics and Arduino for being the backbone of this experiment and supplying their equipment for use, as well as having endless forums with which strings of communication were searched for answers during troubleshooting times. To all the people who posted example codes or helped solve questions others asked, I thank you.

I would like to thank my Thesis Advisor and Research Principal Investigator Aman Haque for his support and flexibility. He was instrumental in my cooperation with people with more experience and knowledge than myself, helped established a project idea, timeline, and

helped to complete the project. He was essential to the editing of this paper and my research experience here at Penn State.

I would also like to extend my thank you to my Honors Advisor and Professor Stephanie Stockar who helped facilitate the writing and editing of this thesis as well as being a person I could approach with any problem.

And finally, a big thank you to Penn State and Schreyer Honors College for pushing me to do research, write a thesis, and get more involved. They've shown me paths I want to take, I don't wish to take, and opened doors and facilitated relationships I couldn't say thank you enough for. I'm honored to be a part of their legacy.

Chapter 1

A Literature Review of Wireless Communication, the Digital Twin Simulation Technique, and Arduino Microcomputers

Tse and Pramond define two fundamental aspects of wireless communication that don't prominently exist in literature.[1] The first is a phenomenon known as fading. Fading is defined as time varying channel strength in response to different effects. The variations can come from a small-scale effect, such as when considering multiple paths of communication for data, or they can come from a large-scale effect, such as distance. This time varying channel strength can cause time delays and communication error between two wirelessly communicating objects. The second phenomenon is very intuitive. As seen in figure 1(a) when we look at a wireline communication the transmitter and receiver devices are connected and can be thought of as the two end points of an isolated linkage. When considering wireless, there is no direct linkage and the prospect of obstacles must be considered. Figure 1(b) symbolizes this interaction between transmitter and receiver. The solid line between them is analogous to the obstacles that must be dealt with when developing a wireless connection. This is formally known as interference. Interference comes from the interaction of many waves and just as in fading, result in time delays and communication errors between two wirelessly communicating objects.[1]

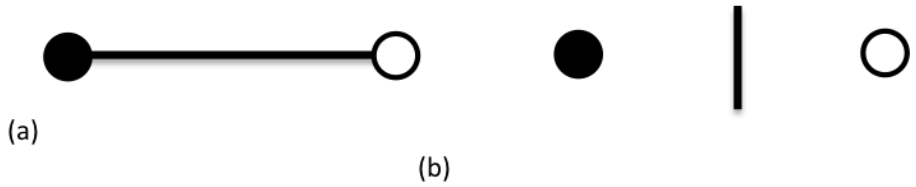

Figure 1. A brief description of the types of communications between two bodies - (a) Wireline communication has a direct connection (b) Wireless communication can move around obstacles because it is not a direct connection

Advancements in the field have led to advanced applications. One is known as telerobotic surgery. Ballantyne gives a review of early clinical results using telerobotics in laparoscopic surgery.[2]

There are inherent limitations to performing laparoscopic surgery in a traditional manner. Laparoscopic surgery is a minimally invasive technique where a small incisions are made to allow access to the portion of the body undergoing surgery. A camera is inserted in to the individual so the surgeon may see the operation and other small tools are inserted at the site of different small incisions, restricting the freedom of the surgeon's movement. In a traditional setting problems such as an unstable camera, limited motion, and poor ergonomics for the surgeon arise due to human error. One possible solution to these problems is the development of a surgical robot platform which surgeons control remotely[2]. Original development came from the idea of battlefield applications where wounded soldiers would be placed in a telepresence surgery vehicle and using 3D imaging systems and telecasting, the scene would be projected to the doctor located in a remote and safe location. The doctor, is then be able to supply immediate attention to the wounded. This technology is still developing and currently the consoles used to prove the concept rely on direct connections[2]. For example, a console named Zeus was able to perform surgery 3800 miles away

from the surgeon controlling it, however this was done via a transatlantic fiber optic cable. According to Ballantyne, the use of satellites delayed the movements of the robot too much for use in 2002.[2] In 2005 it was concluded by Anvari et al. that the telerobotic surgical system was officially in routine use, and it was shown that between two hospitals, a wireless telecommunication link using the local internet and a high priority quality of service (QOS) could be established to connect the surgeon to the robotic system.[3]

Research surrounding wireless communication involves the concept of digital twins. Yang et al.[4] defines digital twin technology as "a method that enables autonomous objects to link the current state of their processes and behavior in interaction with the environment of the real world." This means it is the brain behind Cyber-Physical Systems (CPS) which involves the interaction of some system with another or with humans. The idea of digital twins includes three components. The first is a physical object in real space. The second is a virtual object in virtual space. The third is the connection between the first two components.[4] If applied to a manufacturing setting, a whole new frontier is created.

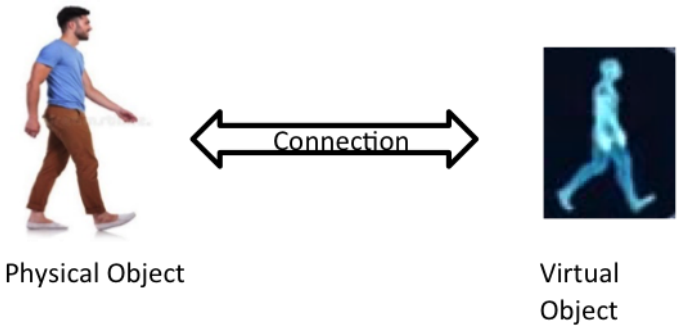

Figure 2. An illustrative example of what Digital Twins are.

The components of digital twins make them extremely valuable in the optimization of behavior. The applications Digital Twins can be used for with regards to connectivity, modularity,

and autonomy are visualized in figure 3a. In mechanical processes, application of the digital twin idea has the potential to create more optimized behavior. Take a machine that is connected to a digital twin. If the actions of the physical device were sent to a simulation which had limitations feedback could be given. If the simulation broke its defined limitations the physical object could be directed to adjust accordingly and thus effectively save the mechanism from malfunctioning.[5] This feedback cycle is visualized in figure 3b. The idea is very similar to that of a negative feedback loop in circuits or biology. If the system deviates to far from an equilibrium or normal position the feedback is initiated to return the system back to normal.

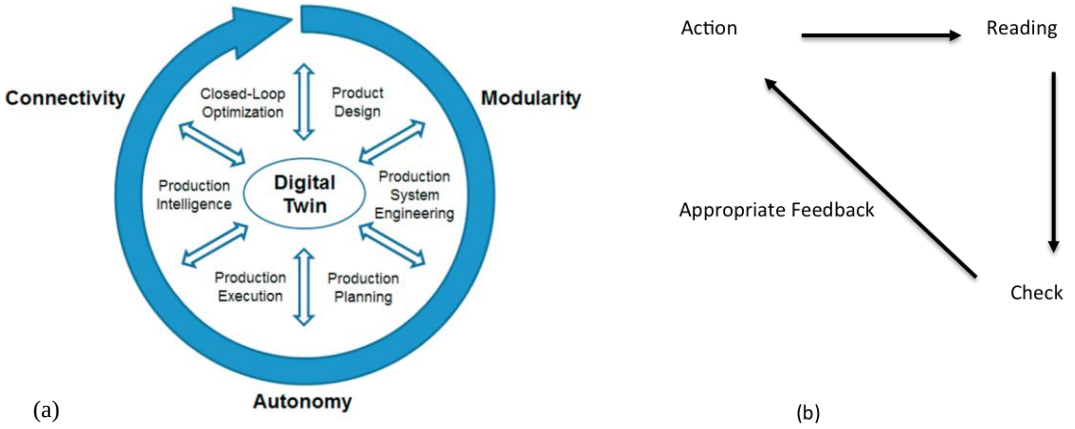

Figure 3. An explanation of feedback cycles and Digital Twins' relationship with connectivity, modularity, and connectivity - (a) Relationships of a Digital Twin[5] and (b) the Feedback cycle

Boschert and Roland[6] defines a digital twin as an integrated Multiphysics, multiscale simulation that uses sensor updates, to mirror it's the actions of a corresponding twin. In flight applications, Digital Twins are used to represent current aircraft state, while determining the most

likely place of structural damage, and thus the most optimal measures of maintenance. Boschert and Roland give the Digital Twin three core characteristics; 1) a link between relevant digital artefacts, 2) the ability to evolve with the real system and integrate current available knowledge, and 3) the capability to derive solutions which are relevant to the real system. Additionally, one of the more beneficial aspects of a Digital Twin is the retention of usable old data. This allows for computer simulations, or virtual imitations, of the physical object to be run, without excessive physical tests, which can be used to predict not only what the test was originally for, but also variables which were not considered and may have an impact on the performance of the physical object in testing. For example, after enough data regarding loading on an engineering part is collected from physical tests, the twin will then be able to predict, not only loading information, but also component lifetime as well. Ultimately the purpose of the Digital Twin, from a mechatronic stand point, was for the retention and availability of data, such that accurate evaluations may be made in the future.[6]

One application of the Digital Twin is the recreation of human motion. A European research project known as REALMAN utilized external marker trajectories to recreate natural looking motion. It involved two steps: 1) defining a digital twin and 2) using kinematic analytical models to estimate joint angles. The digital twin is a digital human model. It was based on a real subject using two superimposing images from different views and created using the software PCMAN. The accuracy of the digital twin was based off of anthropometric dimensional comparisons. The study found that accuracy was related to the original position of the body used in analysis, whether it was sitting, laying, or standing. In the seated position error of up to 44 mm would occur. The mean and standard deviations of differences for the generic arm movement

measurements made by Wang et al.[7] are below in Table 1. A value of zero millimeters would correspond to imitation without any error.

Table 1. Wang et al.'s[7] test of Digital Twin accuracy with the measured differences in generic arm movement and position of a robotic arm known as REALMAN using the software PCMAN.

Subject		04_SD (N=83)		06_FC (N=91)	
Body part	Marker	MEAN (mm)	STD (mm)	MEAN (mm)	STD (mm)
Hand	smc_r	11.8	13.7	12.1	13.9
	vmc_r	13.4	14.2	13.5	13.4
	ls_r	15.6	12.1	15.9	10.9
	ms_r	11.2	12.3	12.2	12.2
Forearm	forearm1	10.9	9.1	11.4	8.1
	forearm2	10	10.8	10.8	10.5
	forearm3	11.8	10	10.7	8.4
	lhe_r	13.2	10.8	10.6	7.7
	mhe_r	13.5	13.8	7.5	7.3
Arm	arm1	10.4	9.1	7.6	6.2
	arm2	10.8	9.3	9.6	7.2
	arm3	13.2	9.8	11.4	7.3
Torso	sn	6.3	5.3	5.1	3.4
	xp	4.6	3.9	5.0	3.4
	t4	5.1	4	4.5	4.2
	c7	4.3	3.7	4.0	2.9
Head	Head_r	4	3.6	3.7	3.0
	Head_l	6	5.4	4.3	3.5
	Head_f	4.5	4.2	4.0	3.4
Pelvis	psis_r	3.2	2.3	3.0	2.2
	psis_l	2.7	2.4	2.1	1.6
	psis_m	3	2.2	3.7	2.5
	crest_r	3.9	2.7	2.6	1.7
	crest_l	7.6	4.9	2.7	2.0

It was concluded that 90% of captured motions were successfully reconstructed with an average error of 20 mm. This digital twin was created with direct connections. Even with a direct connection the accuracy of REALMAN was not very high, thus begging the question of how accurate digital twins may be over wireless.[7]

In order to study wireless communication, an Arduino can be used as the microcontroller in the system. Several studies have been using Arduinos due to their relatively low cost and the fact that the code is open source. A study done by Ali et al.[8] uses Arduino for the interaction between the environment and the sensors within a home. Physical phenomenon outside the home induced a response from sensors placed outside the home which an Arduino was designed to connect to and then, heating and cooling systems would react appropriately within the house as the temperatures rose and fell.

Arduino can be paired with a wireless receiver/transmitter nRF24L01+ to create a wireless connection between a transducer and an actuator. A study done by Rahim et al found that through the use of a single Arduino, nRF24L01+, and sensors at different locations outside a home, 14 different sensors could be communicated with to provide feedback of a house's environment.[8] The nRF24L01 utilizes a frequency of 2.4 GHz and has a max range of 1 km in open space. In closed space its maximum communication distance falls to a tenth of that due to physical obstacles between the nRF24L01 modules, while data rates average 2 Mbps. Each nRF24L01+ is connected to a 3.3 V power supply and each node is powered by a single battery given the Arduino's capability to divide voltage among components.[8]

Figure 4. Images of Arduino and nRF24L01+ Transceiver modules

Arduino's are also common in the development of robotics. Ashraf et al.[9] studied the development of a robotic arm with four degrees of freedom using an Arduino. Robotic control consists of three levels of operation: a microcontroller, a driver, and a computer-based user interface. As previously mentioned, the Arduino programming language is very similar to C but it includes many libraries that help in the control of the I/O ports, timers, and serial communication. The programming process consists of four main steps shown in figure 5 below.

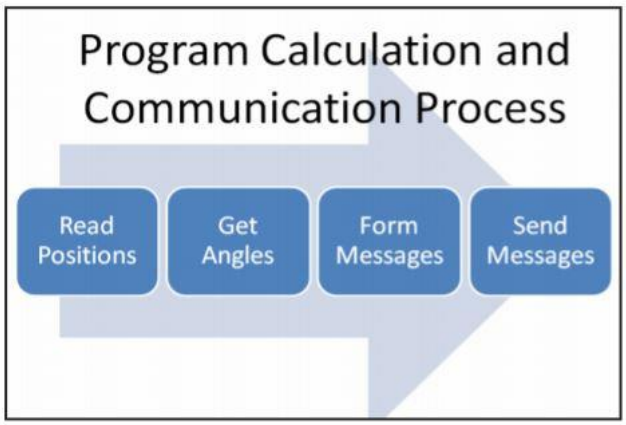

Figure 5. How to Program a robotic arm in simple terms.

Inverse kinematics are common in control, but the option of manual control with the use of potentiometers does exist. Manual use with potentiometers avoids the possibility that the inverse kinematics incorrectly calculates or cannot calculate a valid angle. The potentiometers are connected to the microcontroller and interpret the values of the given movement, then the send of command to the driver.[9]

What literature lacks is the use of a wireless connection in the development of connecting two robots. In this thesis, the idea of "analog twins" is proposed where a real object mocks the actions of another real object as compared to the traditional sense of a digital twin, which is a digital simulation that mocks the actions of a real object. The goal of this thesis is to develop a wireless connection between two VEX Clawbots via an Arduino and an nRF24L01+ transmitter/receiver with the objective of the two robots to perform identical tasks simultaneously. Thus, a more accurate digital twin is brought to life in another real object without having to go through the hassle of a feedback loop to correct the original object. The result of this research will support the improvement of telerobotic surgery application, robotic assembly production lines, and other robotic applications.

Chapter 2

Experimental Method

Building the Bodies of the Robots' 'Original' and 'Copy'

The robots considered in this work are developed by VEX Robotics. They consists of an arm which moves radially in one plane, a claw which moves radially in another separate plane, and four wheels allowing for linear and radial movement in a third plane.[10]. The movement of the robots are restricted to only two degrees of freedom: the movement of the claw apparatus and the movement of the arm. Figure 6 shows the completed forms of the robots in use.

There are two robots and two distinctions between them. The first robot is called 'Original'. Original has a power source directly attached to it and also has a controller which can be programmed to make it move in a particular way. The second robot is called 'Copy'. Copy uses a 9V battery to power its motors and uses an Arduino to be moved. Further description of this will be described in the section: Modifying the Robots 'Original' and 'Copy' for Proper Use in this Study.

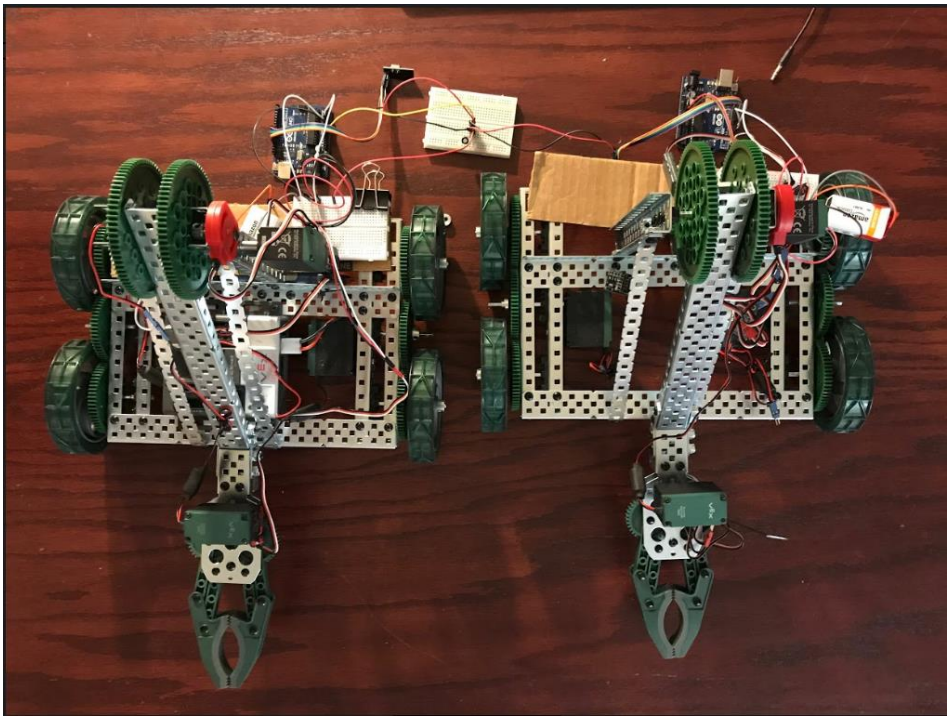

Figure 6. The completed form of the robots in use. A piece of cardboard was used in the back to hold all of the circuitry together while running.

Modifying the Robots 'Original' and 'Copy' for Proper Use in this Study

The robots are predesigned to be programmed and equipped with sensors to perform actions as a single unit. This project involves the use of 'Original' to direct the movements of Copy and thus, the robotic design is adjusted accordingly.

Specific Modifications Made to the Body of 'Original'

Original came with a controller which was programmable and a battery to power that controller. This is programmed as described in the section: Software for 'Original': Programming

the Movement for the Robot ‘Original’ Using ROBOTC Programming Software, and creates the simple motion desired to be duplicated. The battery is directly equipped to the controller and a secondary 9V battery is used to power the Arduino and circuitry attached. Original is equipped with potentiometers which read the degree of movement of both the arm and the claw. One VEX potentiometer is added near the arm motor simply by turning the metal support 180˚ along the z-axis. The second potentiometer is added to the claw motor by removing the casing and attaching it as indicated in Figure 7. The potentiometers have three wires which attach to the breadboard in use. The value wires (white) are then run into the A0 and A1 ports of the Arduino respectively and provide readings of movement. These are the analog ports (as opposed to the digital ports) of the Arduino. These readings are converted to applicable digital data (explained below in the code explanation) and sent to an nRF24L01+ module. The nRF24L01+ module serves as a transceiver allowing for the transmission and reception of information between modules. Original’s nRF24L01+ serves as a transmitter of the data obtained from the potentiometers. The code will be explained in the section: Programming the Arduinos to Transmit and Receive Data Between the Robots ‘Original’ and ‘Copy’ using NRF24L01+ Modules.

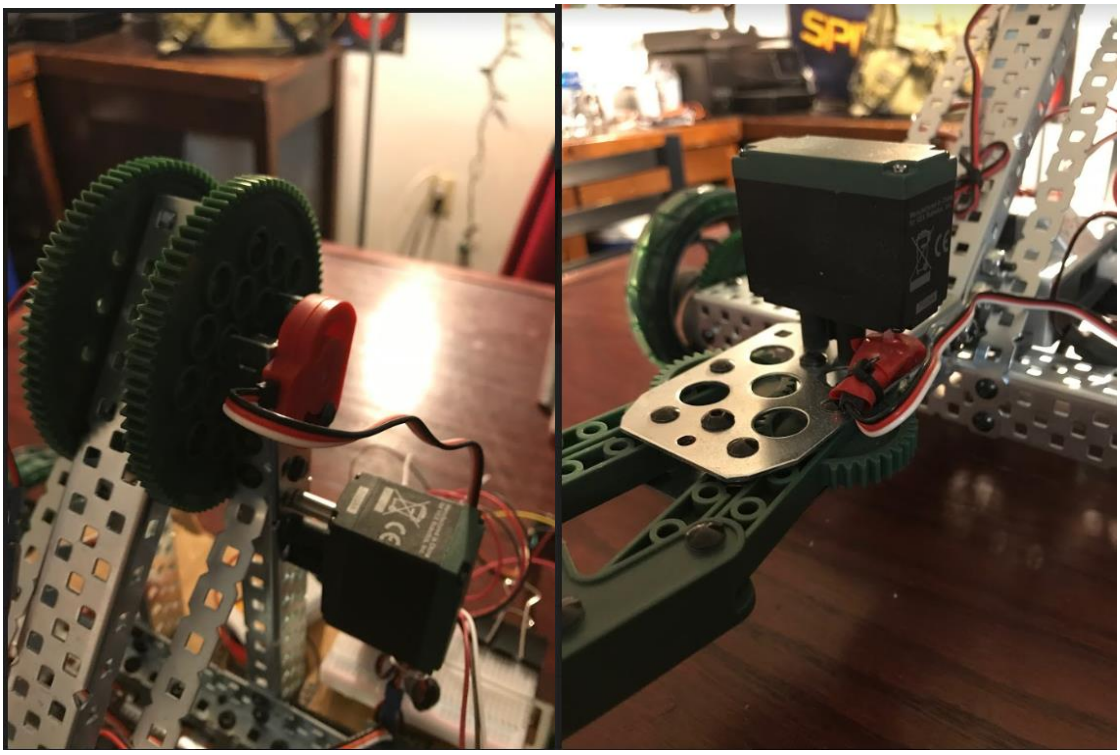

Figure 7. Left, The attached potentiometer for the arm of the robot (degree of freedom one). Right, the stripped potentiometer for the claw (degree of freedom two).

Specific Modifications Made to the Body of 'Copy'

'Copy' is equipped with potentiometers in the same locations to have the option for movement confirmation. The readings from 'Original''s potentiometers are calibrated to the VEX motors as described in the section: Software for Wireless Communication – Programming the Arduinos to Transmit and Receive Data Between the Robots 'Original' and 'Copy' using NRF24L01+ Modules. This causes movement in 'Copy' which is the same as the movement in 'Original.' 'Copy' doesn't have the same supplied power source or controller and is operated completely from the Arduino attached to the VEX motors. A 9V battery operates the system as a whole. The nRF24L01+ attached to 'Copy' serves as a receiver and sends information to the

Arduino instead of from it. After being converted to the correct values the data is projected to the motors attached to the fifth and sixth ports in the Arduino. These control the arm and claw respectively. This then activates the motors to run to a certain degree.

Programming the Movement for the Robot 'Original' Using ROBOTC Programming Software

'Original''s movements are coded in the VEX Robotics 4.X interface: ROBOTC. The code is also found in Appendix A. The motor connected to the arm of Original is plugged into port 7 and labeled leftMotor. The motor connected to the claw of Original is plugged into port 6 and labeled rightMotor. As the code is explained each line of general importance is shown with an explanation following.

```
#pragma config(Motor,  port6,  rightMotor,  tmotorNormal, openLoop)
```

The *#pragma config()* function is automatically generated the 'ROBOTC' configuration wizard and defines these connections as motors, labels them, and opens them to command.

```
task main()
```

The main code is then opened using the command *task main().*

```
wait1Msec(2000);
```

The command *wait1Msec()* tells the motors to act, or hold if nothing comes beforehand, for a certain number of milliseconds before the next line of code is enacted. In this instance Original is told to wait two seconds before performing any action.

```
motor[rightMotor] = -50;
motor[leftMotor]  = 50;
wait1Msec(1000);
motor[rightMotor] = 50;
motor[leftMotor]  = 0;
wait1Msec(1000);
motor[leftMotor] = -10;
wait1Msec(700);
```

The command *motor[] = x* defines the motor which will be moved within the brackets at a certain power level, x. Full power is defined at 127 and full reverse power is defined at -127. A power level of 50 is used to move the arm up and open the claw for one second. The claw is then shut using a power level of 50 for one second and the robot arm is dropped using a reverse power level of -10 for 0.7 seconds. The robot arm is dropped with less power and time due to the presence of gravity and the desire for controlled speed.

Creating a Circuit for Each Robot Using an Arduino and NRF24L01+ Module for Wireless Communication

Wireless hardware is created from scratch using a breadboard to connect the appropriate sensors, motors, Arduino, nRF24L01+ wireless transceiver module, and wires.

In both ‘Original’ and ‘Copy’ the nRF24L01+ module only ran off of a power supply of 3.3V before risking frying the electronics. The Arduino controls the power supply by allowing for restricted use of 3.3V or 5V. To secure proper power supply however these power supplies are only connected to parts, such as the nRF24L01+ module. The motors are capable of running on the 9V battery supply.

Figure 8 shows the schematics for the wiring of ‘Original.’ The potentiometers connect to the analog ports A0 and A1 on the Arduino and send readings from ‘Original’’s movement to the Arduino. After data processing the Arduino then sends this information to Copy via the nRF24L01+ module which serves as a transmitter in this case. The nRF24L01+ module attaches to the Arduino at ports 9, 10, 11, 12, 13, and the 3.3V and GND pins. The nrf24l01+ module’s VCC pin, or supply voltage, connects to the 3.3V pin. The GND, or ground pin, connects to the GND pin on the Arduino. The nrf24l01+ module also has five other important pins: CE, CSN, MOSI, MISO, and SCK. They are connected to pins 8 and 10-13 respectively. The CE pin in an input pin with respect to the nrf24l01+ module and is used for data transmission and reception control when the module is in TX or RX modes (transmission or reception modes). The CSN is the “chip select not” pin. It is active low meaning it should remain high unless one wants to drive using SPI command (aka getting data from this pin). When this pin goes low the module begins listening on its SPI port for data. The MOSI pin stands for “master out, slave in” where the microcontroller is the master and the slave is the module. This pin is where data is sent from master to slave. The MISO pin is the reverse of the MOSI pin and is where data is sent from slave to master. The SCK pin is the serial clock of the SPI interface. Clock sampling occurs in the middle of data bits and should stay low. The nrf24l01+ module has another pin labeled IRQ, known as the

interrupt pin, which is active-low. The transmitter works on a 2.4 GHz frequency and can communicate with each other up to 1 km away.

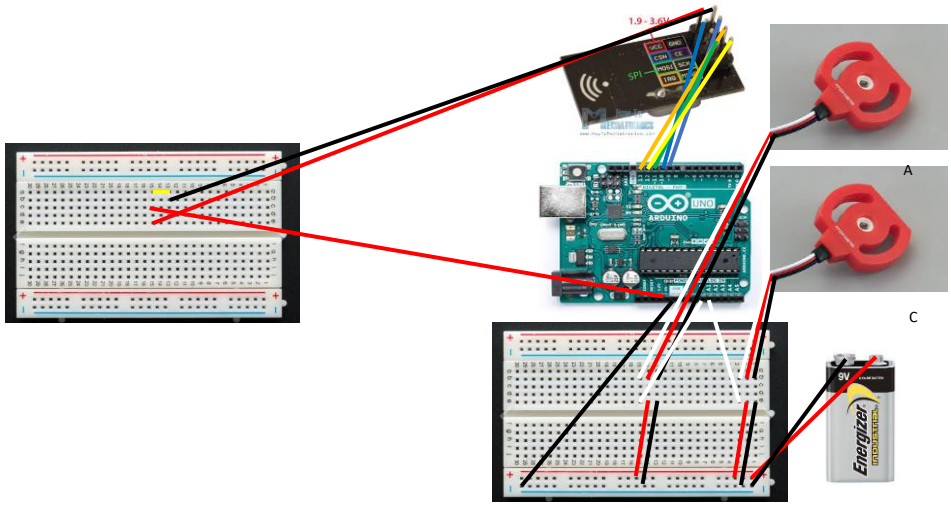

Figure 8. Circuit diagram for 'Original,' reading the potentiometers and transmission.

Figures 9 shows the schematics for the wiring of 'Copy.' The nRF24L01+ module is connected to the same ports/pins but now as the input serving as a receiver. The information is carried through the Arduino where it is sent to the VEX servo motors to move the arm and claw. Each motor is connected to the Arduino at ports five and six respectively. Figure 10 shows the complete circuit diagram.

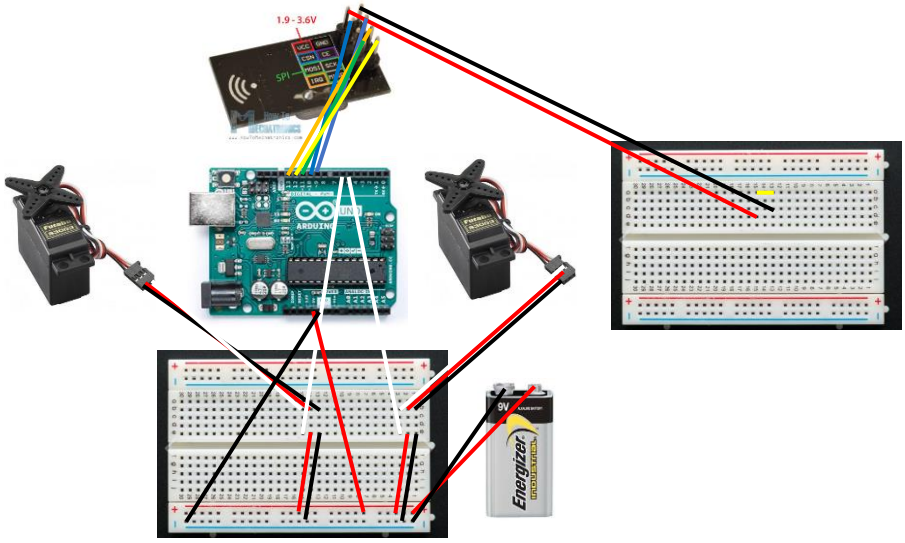

Figure 9. Circuit diagram for ‘Copy,’ data reception and implementation.

\

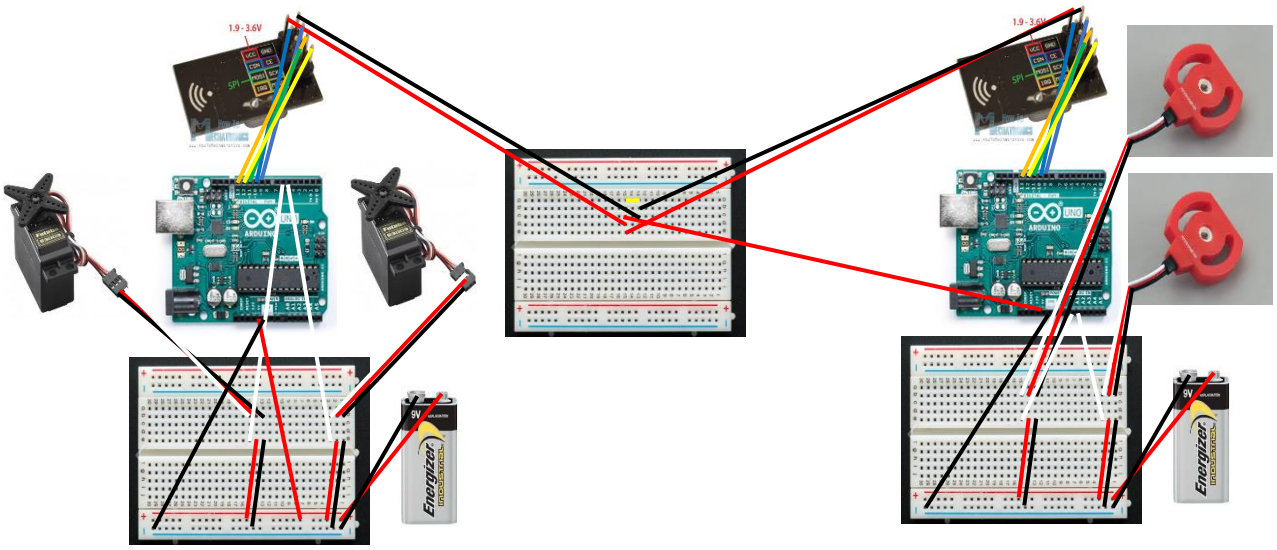

Figure 10. Complete circuit diagram for use of both robots, ‘Original’ and ‘Copy’ during use in the study.[11–13]

Programming the Arduinos to Transmit and Receive Data Between the Robots 'Original' and 'Copy' using NRF24L01+ Modules

There are two sets of Arduino code used to create the wireless network between 'Original' and 'Copy.' The libraries NRFLite by Dave Parson, RF24, RF24Ethernet, and RF24Mesh by TMRh20, and RF24G by Caio Motta were all downloaded within the Arduino 1.8.8 interface and are used by the transceivers. The libraries nRF24L01 and RF24 by TMRh20 are the main libraries used as they are the most widely used and accepted among users of the nRF24L01+. The library SPI is used to account for the communication between the Arduino and a Serial Peripheral Interface (SPI). The following code is explained similarly to the code for the movement of 'Original.' Each line of importance is introduced and an explanation for it follows. The complete code is found in Appendix A.

Programming Communication for 'Original' and Making 'Original' the Parent – Code for the Transmitter

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
```

The *#include*<> command introduces the libraries SPI, nRF24L01, and RF24 to the Arduino so commands which involve their syntax is understandable by the microcontroller.

```
#define CE_PIN   9
#define CSN_PIN 10
```

The *#define* pairs an integer to a string of characters. In this case the string meant to represent the CE and CSN pins of the nrf24l01+ module are paired to the integers 9 and 10 which will later be defined as the pin numbers on the Arduino Uno which they are connected to. The *RF24 radio()* command assigns the CE and CSN pins respectively in the RF24 library which was previously defined. As the input and listening pins, these two must be defined prior to use, however the other pins go undefined because they are purely connections which data is not taken from or commands are given to.

```
const byte slaveAddress[5] = {'R','x','A','A','A'};
const int analogInPinArm = A0;
const int analogInPinClaw = A1;
RF24 radio(CE_PIN, CSN_PIN); // Create a Radio
int sensorToSend[2];
```

The *const byte* command assigns a radio address, called slaveAddress, for the wireless communication. It is defined as a five-string address within the square brackets and as long as this number and series of strings within the curly brackets is the same in both the transmitter and receiver code the two wireless modules will be capable of trading information. The *const int* command defines the constant integers and their analog ports which we take readings from the potentiometers with. The *RF24 radio()* command defines the radio CE and CSN pin numbers which defines the SPI communication of the nrf24l01+ chips. The *int* command creates an array called “sensorToSend” of size two, defined by the number within the square brackets, which index starts at zero. Thus, when calling the first value in the array one must call it using “sensorToSend[0].”

```
void setup() {
   Serial.begin(9600);
   Serial.println("SimpleTx Starting");
   radio.begin();
   radio.setDataRate( RF24_250KBPS );
   radio.setRetries(3,5); // delay, count
   radio.openWritingPipe(slaveAddress);
}
```

To wrap up the setup of the transmitter code the *void setup()* command is introduced. This code is run once, begins the serial printing in the serial monitor, and turns on the radio. The *Serial.begin()* command opens up a serial monitor which transmits data at a specific rate of bits per second (baud). *Serial.println()* then prints the character string which is in parantheses. This lets us know that the code is not broken at this point. The *radio.begin()* command starts up the radio. *radio.setDataRate()* defines the rate of data transmission between the nrf24l01+ modules. *radio.setRetries()* has two inputs separated by a single comma and defines the delay and number of times the system will try to connect upon failure. Assuming the program progresses, the *radio.openWritingPipe()* command establishes which address the transmitter will be sending data on.

```
void loop() {
   sensorToSend[0] = analogRead(analogInPinArm);
   sensorToSend[1] = analogRead(analogInPinClaw);
   sensorToSend[0] = map(sensorToSend[0], 0, 1023, 0 , 180);
   sensorToSend[1] = map(sensorToSend[1], 0, 1023, 0 , 180);
   send();
}
```

The *void loop()* command opens up the section of the code which runs continuously until told to seize. This is where communication is run and values of movement are considered and converted for use in 'Copy.' Each *sensorToSend[]* command is designed to give the value which is transmitted to 'Copy.' First the analog pins are read via the *analogRead()* function. These values come from the VEX potentiometer which gives values between 0 and 1023 so those values are redefined with a mapping function. The mapping function is key here because it changes depending on the motors in use. This will be explained more in the discussion section of this thesis. For this set up values are mapped between 0 and 180 as these values should correspond to the angle with which the motors are run to. The *send()* command then initiates another void function called "send."

```
void send() {
   bool tx;
   tx = radio.write( &sensorToSend, sizeof(sensorToSend) );
   Serial.print("Data Sent: Arm- ");
   Serial.print(sensorToSend[0]);
   Serial.print(" Claw- ");
   Serial.print(sensorToSend[1]);
   if (tx) {
      Serial.println("  received");
   }
   else {
      Serial.println("  Tx failed");
   }
}
```

Here the data transmission of 'Original' is established. A bool function is used to begin as they are in the simplest form a true//false statement. By defining the bool as tx and then defining tx as the *radio.write()* command the command is tested for operation. The bool returns true if it does and false if not. The *if()* statement at the end is set up to print whether the data is received or not depending on these results. The *radio.write()* function takes the values previously defined by the array sensorToSend and sends a data array the size of, and with the values of, sensorToSend. Then the sent values are confirmed with the series of *Serial.print()* commands.

Creating ‘Original’’s Twin - Programming ‘Copy’ as the Receiver,

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>
```

Again the *#include*<> command introduces the libraries in use. All are the same, but now the servo library is added because command of the servo motors is required.

```
#define CE_PIN   9
#define CSN_PIN 10
```

Again, these character strings are paired to the digits 9 and 10.

```
const byte thisSlaveAddress[5] = {'R','x','A','A','A'};
Servo armServo;
Servo clawServo;
```

The *const byte* command defines the communication address. It can be named something different here but must be the same size and made of the same character strings. The *servo* command defines the servo motors names for future use.

```
RF24 radio(CE_PIN, CSN_PIN);
int sensorReceived[2]; // this must match dataToSend in the TX
bool newData = false;
```

The radio and SPI is established with CE and CSN pins once again here. The integer being received is given a size, within the square brackets, with the *int* command. This established an empty array which is filled later, but it is important that the size of this array is the same size as the one defined

in the transmitter code. A bool variable named newData is also established as false here. This will be important later.

```
 void setup() {
   Serial.begin(9600);
   armServo.attach(5);
   clawServo.attach(6);
   Serial.println("SimpleRx Starting");
   radio.begin();
   radio.setDataRate( RF24_250KBPS );
   radio.openReadingPipe(1, thisSlaveAddress);
   radio.startListening();
}
```

Most of this code is the same as in the setup of the transmitter. The serial baud is established, the start-up of the module is printed as a check, the radio is initialized, and the data rate for the nrf24l01+ module is established and should be the same as in the transmitter code. The *armServo.attach()* and *clawServo.attach()* commands define the pins which the servo motors are connected to. In contrast to the transmitter code where a writing pipe is opened, here a reading pipe is opened. The value of 1 defines the pipe number and then the address of communication is paired to that pipe. There are up to 6 pipes available with nrf24l01+ modules thus one could have six different modules communicating with a single base if necessary. In this experiment only one pipe is used and an array containing all data is sent between them, but two pipes each with a single value could have been used. The radio is then commanded to begin listening for data using the *radio.startListening()* command.

```
void loop() {
   getData();
   implementData();
}
```

The continuous loop established here runs the code through two other void functions called “getData” and “implementData.”

```
 void getData() {
   if ( radio.available() ) {
      radio.read( &sensorReceived, sizeof(sensorReceived) );
      newData = true;
   }
}
```

Here the data reception section of the receiver’s code is established. The initial if statement states that if the command *radio.available()* reads some sort of information then the following code is implemented. The *radio.read* command reads the available data sent from transmitting nrf24l01+ module. It then reassigns those values to the previously established array “sensorReceived” with correcting sizing. The bool variable “newData” is then redefined as true for use in the next void function “implementData.”

```
void implementData() {
   if (newData == true) {
      Serial.print("Data received ");
      Serial.println(sensorReceived[0]);
      armServo.write(sensorReceived[0]);
      Serial.println(sensorReceived[1]);
      clawServo.write(sensorReceived[1]);
      newData = false;
   }
   }
```

In the implementData loop the data which is obtained through getData is put to use on ‘Copy.’ In the newData loop it is established that if the radio is open and sending data that the bool variable newData is redefined as true. Now that this is the case the if statement here is run through. Here a confirmation print is made in the serial monitor along with the values of the established array. The correct respective values are also written to the appropriate servo motors using the *armServo.write()* and *clawServo.write()* commands. The bool variable newData is then set to be false to keep this loop from infinitely repeating itself with the same data as new data is transmitted and read.

Chapter 3

Results

This chapter presents the results obtained upon execution of the code described in the previous chapter.

The RobotC code used as commands for ‘Original’’s movement worked within a high degree of accuracy making ‘Original’ move as desired. The only necessary modification is made in its command to move downward. Its downward movement is driven by gravity as well as the available motor, therefore its code is adapted to keep the robot arm from aggressively hitting the metal frame.

The Ardiuno Unos are tested for their ability to send information using a combination of online basic tutorials and the Arduino software’s built in example codes. At first the two are plugged into a PC and simple text is sent to and from the Arduinos. The transceiver capabilities of the nrf24l01+ modules are checked using the *radio.isChipConnected()* command. Once working modules are acquired and attached their communication is tested using a simple text test similar to the previous one used for Arduinos. They are then tested in their abilities to send integers. Here things prove to be slightly off, but it is not an uncommon issue. Many forums have threads regarding the issue. Once integers are received and the other tests are all successful the code previously discussed and found in Appendix A is implemented.

Unfortunately, the use of the coding for ‘Original;’s movement was unused. Instead ‘Original’ is manually displaced to adjust for the complications that arose with the project. These are discussed in the next chapter. In the first set of trials ‘Original’’s arm moved upward and downward. ‘Copy’ performed the same action but there was was large time delay. After this, one of the nrf24l01+ modules burned out and had to be replaced. In the final round of trials ’Original’

was displaced starting at a large angle and moved downward. The claw was then opened. When ‘Original’ was displaced ‘Copy’ also moved. There was a slight time delay between the two most likely accounted for by the time it takes took the data to travel between the robots and activate the appropriate motors. The arm movement of ‘Copy’ moves at a constant speed regardless of ‘Original’’s movement and it is heard that the motor wanted to continue to push the arm of ‘Copy’ past its lowest position. Eventually it stops running. The claw movement of ‘Copy’ follows ‘Original’’s as well, however this is more difficult to assess due to the movements small nature and lack of smoothness as compared to the arm movement. Again, the motor continues running past the desired position, but eventually stops. The motors are unable to power upward movement, most likely due to a lack of battery power from the 9V batteries in use.

Figures 11 and 12 show time stamps of the movements of Copy and Original from two videos taken during the most recent testing showing the mirroring capabilities of the robots. Previous trials went unrecorded but many of the trials yielded similar results.

Figure 11. Time stamps for arm movement of Original (left) and Copy (right). Close inspection reveals a slight time delay between the two but accurate speed and position. Left to right and top to bottom the time stamps are as follows: t = 2s, 4s, 5s, and 6s.

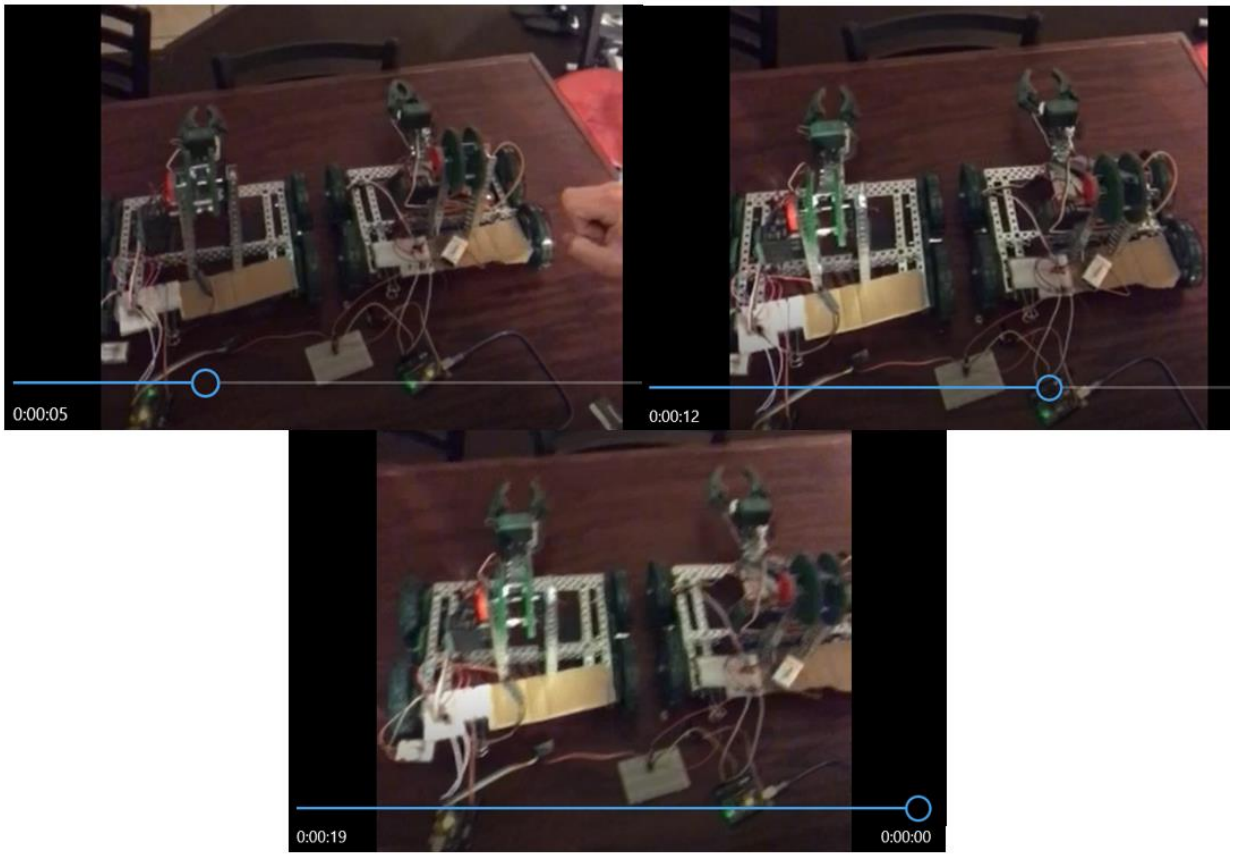

Figure 12. Time stamps of claw movement. Close inspection reveals that the claw of ‘Copy’ (left) was open to begin with. This may due to cross contamination of data when being transmitted, or some form of transmission error. The top left image is after manual displacement of ‘Original’ (right). In the bottom image one can see that ‘Copy’’s claw is now further open. Left to right and top to bottom the time stamps are as follows: t = 5s, 12s, and 19s.

The main goal of the experiment is to establish a connection between the two robots with a secondary goal of moving both the arm and claw within the same movement. In other words, the goal is to move two degrees of freedom on ‘Copy’ wirelessly using ‘Original’ as a parent system. The concept is proven possible, with some error which will be discussed along with the sources of such error below in Chapter 5.

Chapter 4

Error Discussion and Possible Implications of this Research

In this chapter, the errors of the study are discussed to provide helpful guideline for future improvements while recreating, replicating, or building one's own project with similar purpose. When running the experiment, a large issue that is encountered is the ability to receive the desired integer values across the system. Specifying the data type as DEC during transmission and reception helps with this. The problem is a consistent one discussed across forums and after multiple trial runs and reading across many forums it is clear that having a working nrf24l01+ module is important here and ensuring the use of right data pins is even more important. The SPI library in use must be checked because the use of the proprietary SPI pins for radio connection will complete or hinder the communication. For Ardunio UNOs the Slave Select pin is designated as 10 normally with the MOSI, MISO, and SCK pins defined for 11, 12, and 13.[14] In this experiment getting correct integer values consistently did not occur but data is being consistently transmitted and received resulting in movement of the robots. After reading many forums from the Arduino page[15] it was determined that this error may have come from an issue within the hardware itself. Similar projects with code setup similar to this experiment were run successfully and yielded accurate results in the past.

Hardware selection is important not only for communication purposes but also for operation. There are two different types of servo motors one can use to operate movement. Different servos run off different styles of code. Some servos are run according to position (these are the simplest to work with and are highly recommended). They operate simply by writing a degree value between 0 and 180, with 90 as the center, to the motor. No math is involved and the position is absolute instead of being relative to the current position the motor is at. Another type

of servo motor is the type which operate with raw pulse duration and are referred to as continuous-rotation servos. These servos look for a command telling them how long to run. The time value is written between 1000 and 2000 microseconds. Here things get confusing because there are projects and recommendations which suggest both ways should be used to drive the VEX servo motors even though they are continuous rotation servos. The best trial run during this experiment mapped the potentiometer values between 0 and 180 using position as the desired position. There are projects which show code in which both styles of command writing work equivalently well.

The last consideration that must be addressed is the power supply of everything and the wiring connections. The servo motors can be run off a 5V power supply[16] but the nrf24l01+ modules must be run at a 3.3V supply or risk being burnt out. In this experiment a 9V battery is used to power the Arduino and motors however over time the batteries deplete and by the end of the time allotted for the experiment the voltage supply is depleted enough that the motors are unable to run. The wiring through the breadboards also seem imperfect. Sometimes the wires would need to be plugged into different ports in the breadboard or moved around in order for connections to be made. Soldering is highly recommended but it is semi-permanent so the circuitry should be accurate. Otherwise a good breadboard and better wires are recommended to ensure complete and accurate connections.

Looking past the error, this proof of concept leads to some interesting implications. There are many areas in which this concept could be used. As mentioned in Chapter 1, the use of wireless technology is already a major tool in the modern age. Between telerobotics, communication, and many other forms of use, wireless is an integral part of the world we live in today.

The ability for two different robots to communicate on a wireless link provides us with many possibilities. First off, the use of this concept at an industrial level may increase efficiency

levels in the modern assembly line. The assembly line is used to speed up production and if mass production is the goal of a company the option to drive multiple assembly lines with one now arises. This idea of mass production is not limited to macro good production but can also be applied to things such as drug production or mass food production.

Next, one can apply the concept to things in the medical field. Drug production has already been mentioned, but now the concept will be used for something more similar to the idea behind telerobotics. This is where the established link and movement possible is not used for production, but rather for performance. In telerobotics the link allows a surgeon to perform some action miles away from the field while a robot performs surgery in the field. This can prevent a loss of crucial time in certain situations, and keep medical personnel from having to enter dangerous zones. The concept proven here continues to support the idea of telerobotic surgery.

Another performance application revolves around prosthetics. An amputee who loses their leg may get a prosthetics but unfortunately neural integration is still in its infant stage of development. Prosthetic movement is thus not very well developed. If one could attach potentiometers to a leg brace and use that to drive the motion of a leg prosthetic with a simple time delay, more complicated movements may be achievable. This time delay could then be adjusted for different movement types such as running and jumping. Attach an AI which could learn patterns and control algorithms, and a non-neurally integrated prosthetic that is complex enough to handle basic human function may be achievable.

A final area, and one of continuous interest in which digital twins are generally used for, is the area of optimization. Generally, the optimization is handled in simulation and adjusted for in practice. However, one cannot always make the adjustments manually as shown in simulation

so by bringing the optimization problem to life we can avoid adjusting something impractical to manually adjust.

The proof of concept provided in this thesis may entail many fields where wireless communication is applicable. Although wireless is a popular tool there is very little literature on the subject. What does exist are online tutorials and videos to show how to code in Arduino and wireless. This research study is designed to provide literature with a method of coding wireless communication using Arduino. In addition, it presents recurring issues during operation of experiments involving analog twins. The study and code are successful in moving ‘Copy’ in response to ‘Original’’s movements. The study reveals many flaws in the coding of the robots. This study allows for the surfacing and confrontation of these errors and flaws such that other researchers might be able to troubleshoot and solve the current issues quicker. This idea is already applied in many areas but it could be further explored in industry with the production assembly line, medicine and drug delivery, Orthotics and Prosthetics, physical therapy, and many others.

Appendix A

The Code Programmed for each Robot Involved in the Study

ROBOTC Program for the Parent Movement of 'Original'

```
#pragma config(Motor,  port6,           rightMotor,    tmotorNormal, openLoop)
#pragma config(Motor,  port7,           leftMotor,     tmotorNormal, openLoop)

//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

//+++++++++++++++++++++++++++++++++++++++++++++| MAIN |+++++++++++++++++++++++++++
task main()
{
  wait1Msec(2000);
  motor[rightMotor] = -50;
  motor[leftMotor]  = 50;
  wait1Msec(1000);
  motor[rightMotor] = 50;
  motor[leftMotor]  = 0;
  wait1Msec(1000);
  motor[leftMotor] = -10;
  wait1Msec(700);
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Wireless Communication for the Parent 'Original' – The Transmitter Code

```
// Tx - transmitter


#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>
```

```
#define CE_PIN   9
#define CSN_PIN 10

const byte slaveAddress[5] = {'R','x','A','A','A'};
const int analogInPinArm = A0;
const int analogInPinClaw = A1;

RF24 radio(CE_PIN, CSN_PIN);

int sensorToSend[2];
//================

void setup() {

   Serial.begin(9600);

   Serial.println("Tx Starting");

   radio.begin();
   radio.setDataRate( RF24_250KBPS );
   radio.setRetries(3,5);
```

```
    radio.openWritingPipe(slaveAddress);

}

//====================

void loop() {
    sensorToSend[0] = analogRead(analogInPinArm);
    sensorToSend[1] = analogRead(analogInPinClaw);

    sensorToSend[0] = map(sensorToSend[0], 0, 1023, 0 , 180);
    sensorToSend[1] = map(sensorToSend[1], 0, 1023, 0 , 180);
    send();
}

//====================

void send() {

    bool tx;
    tx = radio.write( &sensorToSend, sizeof(sensorToSend) );
```

```
    Serial.print("Data Sent: Arm- ");
    Serial.print(sensorToSend[0]);
    Serial.print(" Claw- ");
    Serial.print(sensorToSend[1]);

    if (tx) {
      Serial.println(" received");
    }
    else {
      Serial.println("  Tx failed");
    }
}

//================
```

Wireless Communication for the Child ‘Copy’ and the Movement of ‘Copy’ After Data Reception –Code for the Receiver

```
// Rx - receiver

#include <SPI.h>
#include <nRF24L01.h>
```

```
#include <RF24.h>
#include <Servo.h>

#define CE_PIN   9
#define CSN_PIN 10

const byte thisSlaveAddress[5] = {'R','x','A','A','A'};
Servo armServo;
Servo clawServo;

RF24 radio(CE_PIN, CSN_PIN);

int sensorReceived[2]; // this must match dataToSend in the TX
bool newData = false;

//===========

void setup() {

    Serial.begin(9600);
    armServo.attach(5);
    clawServo.attach(6);
```

```
    Serial.println("SimpleRx Starting");
    radio.begin();
    radio.setDataRate( RF24_250KBPS );
    radio.openReadingPipe(1, thisSlaveAddress);
    radio.startListening();
}

//=============

void loop() {
    getData();
    implementData();
}

//==============

void getData() {
    if ( radio.available() ) {
        radio.read( &sensorReceived, sizeof(sensorReceived) );
        newData = true;
    }
}
//=================
```

```
void implementData() {
    if (newData == true) {

        Serial.print("Data received ");
        Serial.println(sensorReceived[0]);
        armServo.write(sensorReceived[0]);

        Serial.println(sensorReceived[1]);
        clawServo.write(sensorReceived[1]);
        newData = false;

    }
    }
```

Appendix B

Materials and Equipment

Table 2 presents the materials used during this experiment. The items' names, quantities, and general descriptions are located in the respective column.

Table 2. Material items, quantities, and brief descriptions of all equipment used in this research

Item	*Quantity*	*Description*
Arduino Uno	2	A microcomputer system used to read and control the movements of each robot.
nrf24l01+ modules	2	Wireless transceiver modules used to relay data between robots
VEX Clawbot	2	VEX robots with multiple degrees of freedom (only the claw and arm are used in this experiment however).
VEX controller	1	Programmable controller using ROBOTC to move Original.
VEX servo motors	4	Motors to control each robots arm and claw.
VEX potentiometers	4	Potentiometers used to read angle outputs of Original and check outputs of Copy.

VEX battery back	1	Powers Originals movements as coded in ROBOTC.
9V battery	2	Powers Arduino Unos and nrf24l01+ modules, therefore powering signal transduction and Copy's movements.
10 µF Capacitor	1	Filters noise from nrf24l01+ modules and helps to control power supply.
Wires	As needed	Signal transduction.
ROBOTC software	N/A	Software for VEX Clawbot control (with controller).
Arduino software	N/A	Software for Arduino control (VEX Clawbot control without controller).

BIBLIOGRAPHY

1. Tse D, Tse D, Viswanath P, Viswanath P. *Fundamentals of Wireless Communication 1*.; 2004. doi:10.1109/TIT.2008.2009814
2. Ballantyne GH. Robotic surgery, telerobotic surgery, telepresence, and telementoring: Review of early clinical results. *Surg Endosc Other Interv Tech*. 2002;16(10):1389-1402. doi:10.1007/s00464-001-8283-7
3. Anvari M, McKinley C, Stein H. Establishment of the world's first telerobotic remote surgical service: For provision of advanced laparoscopic surgery in a rural community. *Ann Surg*. 2005;241(3):460-464. doi:10.1097/01.sla.0000154456.69815.ee
4. Yang W, Yoshida K, Takakuwa S. Digital Twin-Driven Simulation for a Cyber-Physical System in Industry 4.0 Era. 2017:227-234. doi:10.2507/daaam.scibook.2017.18
5. Rosen R, Von Wichert G, Lo G, Bettenhausen KD. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*. 2015;28(3):567-572. doi:10.1016/j.ifacol.2015.06.141
6. Boschert S, Rosen R. Digital twin-the simulation aspect. In: *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and Their Designers*. ; 2016. doi:10.1007/978-3-319-32156-1_6
7. Wang X, Chevalot N, Monnier G, Ausejo S, Suescun Á, Celigüeta J. Validation of a Model-based Motion Reconstruction Method Developed in the REALMAN Project. *SAE Tech Pap 2005-01-2743*. 2005;2005(01-2743):9. doi:10.4271/2005-01-2743
8. Ali Z, Rahim A, Nawaz SS. Design and Implementation of a Low Cost Wireless Sensor Network using Arduino and nRF24L01 (+). 2016;01(5):307-309.
9. Elfasakhany A, Yanez E, Baylon K, Salgado R. Design and Development of a

Competitive Low-Cost Robot Arm with Four Degrees of Freedom. *Mod Mech Eng*. 2011. doi:10.1007/978-3-319-24543-0_5

10. VEX. *GUIDE FOR BUILDING THE CLAWBOT*. http://www.sydt.com.tw/download/VEXEDR/VEXEDR_Clawbot Build Instructions.pdf. Accessed January 3, 2019.
11. Arduino Wireless Communication - NRF24L01 Tutorial - HowToMechatronics. https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/. Accessed March 25, 2019.
12. s3003-servo-motor. http://www.shopathomes.co/home/248-s3003-servo-motor.html. Accessed March 25, 2019.
13. *Reference Potentiometer Potentiometers Overview*. http://cdn.robotc.net/pdfs/natural-language/hp_pot.pdf. Accessed March 25, 2019.
14. Arduino. SPI Library. https://www.arduino.cc/en/Reference/SPI.
15. Arduino. Arduino Forum. http://forum.arduino.cc/index.php. Accessed March 28, 2019.
16. ROBOTC. How to connect a VEX motor to the Arduino - ROBOTC API Guide. http://www.robotc.net/wikiarchive/Tutorials/Arduino_Projects/Mobile_Robotics/VEX/Connecting_A_VEX_Motor. Accessed March 28, 2019.

Academic Vita of Nathaniel D Graham

nqg5152@psu.edu

Education
Major(s) and Minor(s): Biomedical Engineering, Mechanical Engineering
Honors: Mechanical Engineering

Thesis Title: ANALOG TWINS: DEVELOPMENT OF WIRELESSLY COMMUNICATING ROBOTIC TWINS WITH MULTIPLE DEGREES OF FREEDOM

Thesis Supervisor: Aman Haque, PhD

Work Experience:

Artificial Heart and Cardiovascular Fluid Dynamics Laboratory 06/2018-05/2019
The Pennsylvania State University, University Park, PA
Led projects in biomedical engineering laboratory research. Led a team of three individuals and learned how heart disease causes pathology through the design and creation of a translucent silicone model combined with PIV analysis of said model.

Teaching Assistant
The Pennsylvania State University, University Park, PA 08/2017-05/2019
Physiology Laboratory
Supervisor: John Waters and John Neisser
Prepared lesson plans for teaching physiology and laboratory techniques. Taught students of ages above and below my own, while adapting to how each one learns individually. Utilized multimodal learning to teach students. Operated equipment to analyze physiological readings and prepared chemicals for use. Trained in proper animal care and use for amphibians, and child abuse safety. Directed frog dissections.

Academic Grader
The Pennsylvania State University, University Park, PA *08/2017*-05/2019
Supervisor: Math Department
Supervisor: Biomedical Engineering Department
Selectively judged and graded specific problem sets for students to best prepare them for future questions and classes. Kept track of and organized papers and files for student grades.

Food Service Provider
Scalo Northern Italian Grill, Albuquerque, NM 06/2017-08/2017
Supervisor: Neta Domiguez
Performed tasks such as bussing, serving, and hosting. Provided customer service specific to the needs of each individual customer. Opened store, and cleaned and closed store at night in compliance with all company standards. Learned how to approach a wide variety of individuals in many different settings to best satisfy their expectations.

Construction and Landscaping
Realty One of New Mexico, Albuquerque, NM 05/2016-08/2017
Supervisor: Jennifer Graham
Performed tasks such as designing, sanding and painting wood, and rock removal, treatment, and placement. Learned many. practical lessons such as attention to detail, patience, and time management.

Grants Received

- Federal Pell Grant | 2018-2019
- Penn State Academic Grant from Schreyer Honor College for Summer Research | 2018

Awards:

- Academic Excellence Scholar | 2015-2019
- Marley Trustee Scholarship | 2018-2019
- Pre-Eminence in Honors Education Scholarship | 2017-2018
- Regal Family Trustee Scholarship | 2017-2018
- Schumacher Honors Scholarship | 2016-2017

Community Service Involvement:

THON 08/2015-05/2019

For the last four years I have participated in Penn State's student-run philanthropy committed to enhancing the lives of children and their families who have been impacted by pediatric cancer. I have been a participant, member, liaison, organization's chair, and dancer. Throughout the fundraising window from September to February we solicit money by online, in-person, and corporate donations along with raising awareness across multiple states in person, and the world electronically. One of our main goals is to also provide support to families in need, not only via money but also emotionally. My organization, Club Gymnastics, is currently paired with two families, the Damesheks and the Newswangers. We dedicate time and energy to making sure their lives are as enjoyable as possible. The Damesheks are a bereaved family (their daughter Emilia passed away form cancer in May of 2016), and the Newswangers are a family of survivors (their son Weston is now in remission).

YMCA Gymnastics Clinics 2017-2019

Once a month my organization, Club Gymnastics, takes time out of our weekend to go and volunteer coach young girls gymnastics. It gives them a great experience and peers to look up to as we teach and work with them.