Robotic Flange Assembly

Antonio Buda, Cassidy Lyons, Viviana Gonzalez Pascual, and Alana Icenroad

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — In large-scale pipe assembly, there is an industry standard method of tightening bolts on pipe flanges which involves torqueing the bolts in a specified pattern through incremental torqueing levels. This is done by an individual operator using a handheld hydraulic tightening device and may takes several hours. This is physically and mentally demanding for the worker and can lead to workplace accidents as well as inconsistent tightening on the pipe. The device shown here will physically assemble the pipe in a consistent and predetermined pattern, alleviating strain on the worker and providing more efficient assembly.**

***Index Terms* — Autonomous systems, flanges, manufacturing automation, robot, safety, torque.**

I. Introduction

The Robotic Flange Assembly project is an interdisciplinary project sponsored by Siemens that consists of 17 members. The following text describes the Robotic Flange Assembly project as stated in the project document:

"Flanges are often used to attach pipes carrying fluids and gases to various power plant equipment and systems. Flanges are typically round rims welded to the end of pipes with a sequence of holes for threaded fasteners to attach two pipes together. A gasket is usually placed in between two flanged pipe ends to maintain a seal. The assembly of flanges are manually intensive processes requiring careful control so as to maintain a proper seal.

The goal of this project is to develop a scaled prototype robotic flange assembly assistant to demonstrate and better understand the opportunities associated of the system. Benefits of the proposed system include reduced manual effort and improved quality." [1]

The final product is designed to automate the assembly of flanged pipes with minimal worker involvement. The process for proper flange assembly requires a strict set of guidelines to adhere to; following the proper torque sequencing pattern and following the proper bolt tightening process from the ASME standards.

This specified pattern consists a distinct order that requires precise and repeated movement. The delivery of torque is not all at once for each bolt and the amount of torque delivered depends on a time and percentage of the overall specified torque. The ASME [2] specified pattern for tightening is used to apply torque in stages. The staged torque will vary depending on how many bolts a flange has. For example, torque may be applied to each bolt on the flange in three stages where 30, 60, and then finally 100 percent of the torque is applied. This ensures that bolts on the flange are evenly and properly tightened. Displayed in Figure 1 below is the ASME specified torque pattern that the Robotic Flange Assembly will be following.

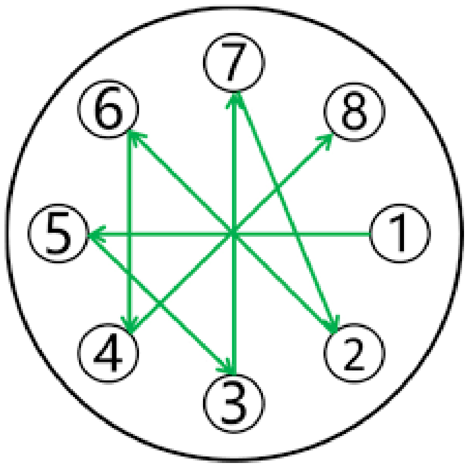

Fig. 1. ASME Torque Sequencing Pattern.

The Robotic Flange Assembly system has thus been divided into three distinct subsystems to achieve these goals; the Torque Drive Subsystem, the Articulation Subsystem, and the Carriage Subsystem. We divided the Cobot into three subsystems that center around the functions that each motor satisfies. Starting with the initial alignment setup of the Cobot on the pipe to the first bolt, the Torque Drive Subsystem will begin first and start the tightening process of the first bolt. Next, the Articulation Subsystem will engage and disengage the socket with the bolts. Finally, the Carriage Subsystem that uses a chain attachment to fasten the Cobot around the pipe will circumnavigate around the pipe in the specified torque sequencing pattern to each bolt on the flange.

II. System Components

The Robotic Flange Assembly is comprised of several of important system components. The best way to exhibit the system is in terms of the system components. This section provides a semi-technical introduction to the main individual components that make up the Robotic Flange Assembly.

A. *Microcontroller*

The main component of the Robotic Flange Assembly is the Arduino Mega 2560 microcontroller (ATmega 2560). The ATmega 2560 microcontroller was chosen for our system because of the compatibility advertised with every motor controller that was considered for this system. Moreover, it was also chosen for its high memory capacity and abundance of digital and analog pins that include pulse width modulation for controlling the speed of the motors. The Arduino IDE is used to program C/C++ code to develop and debug the logic incorporated across all subsystems as we integrated all moving parts.

B. *M18 Motor*

The M18 Motor inside the Milwaukee Power Drill is the main motor responsible for supplying power to two out of three subsystems in the Robotic Flange Assembly. This motor was selected for its lightweight capability to provide the power and stability necessary to move the Cobot around the pipe and provide torque.

Some of the features of this motor include that it comes with an 18V red Lithium-ion battery, it has an RPM value of 1800, and it can produce up to 40 ft-lbs. of torque.

C. *Stepper Motor*

The Nema 17 Stepper Motor is the primary motor responsible for driving the radial direction of movement that engages the socket with the bolt in the Articulation Subsystem. This stepper motor was chosen because of its 200-steps per revolution precision. This precision is extremely important because we must ensure that the same distance traveled is repeated to and from every bolt. Connected to the stepper motor will be a lead screw with a nut inside, allowing for movement forwards and backwards since the stepper normally moves in a rotating radial direction.

Some of the features of this low-cost, bipolar stepper motor include having a step angle of 200-steps per revolution at 1.8° per step for smooth motion. This stepper motor was purposely built for high-holding torque, allowing for the ability to incrementally step to the next position. This results in a simple positioning system that doesn't require an encoder, and it makes stepper motor controllers very simple to use.

D. *Optical Encoder*

The E3-2000 US Digital Optical Encoder was the encoder chosen for the Robotic Flange Assembly. This optical encoder is designed to easily mount to and dismount from an already existing shaft to provide digital feedback information. The optical encoder is used in the Carriage Subsystem on the main motor shaft to determine where the position of the bolts are located on the flange.

Some of the features include being able to supply 2,000 cycles per revolution and can easily interface with the ATmega2560 MCU and tested with the Arduino microcontroller. Furthermore, it is simple to assemble and compact in size, making it easy to mount to the main motor shaft of the Cobot. Furthermore, it is simple to assemble and compact in size, making it easy to mount to the main motor shaft of the Cobot.

E. *Touchscreen LCD*

The 2.8” Touchscreen LCD by Adafruit was the LCD chosen for the Robotic Flange Assembly. The Cobot will integrate a small user interface function that will be used for the initial set up of the Cobot. This 2.8" LCD was chosen for its touchscreen capabilities and its compact size. The operator working alongside the Cobot is expected to provide minimal input, so it is not necessary to have a large sized LCD.

Some of the features include that it has an 8-bit digital interface and an operating voltage of 3-5V, making it compliant to be used with any microcontroller. Additionally, this LCD has a controller built into it with RAM buffering, so that almost no work is done by the microcontroller.

III. System Concept

The purpose of the Robotic Flange Assembly is to design a Cobot that works alongside the worker to tighten the bolts on flanges. The Cobot runs based of the integration of three separate subsystems, all controlled through the main printed circuit board in the back of the device. The Torque Drive Subsystem gives a controlled torque on the bolts of the flange and feeds information on the torque to MCU. The Articulation Subsystem moves the torque drive subsystem forwards and backwards in the assembly so the torque socket can meet the bolt. The Carriage Subsystem provides the motor and sprocket holding the chain that will move the Cobot around the pipe in a controlled sequence.

The system will rely on two 18V power supplies shared across the device that are stepped down to voltages of 9 and 5 where necessary. As well, the carriage and torque drive motors will each rely on one Cytron motor driver, while the Articulation Subsystem will be driven with SparkFun’s Easy Driver motor controller.

The simplest way to explain the concept of the system is to demonstrate how the system is running using a flowchart. The flowchart will illustrate how the subsystems are interacting with each other and show their sequence of arrival. Breaking the system down into a pictured diagram will allow for a better overall understanding of the entire system.

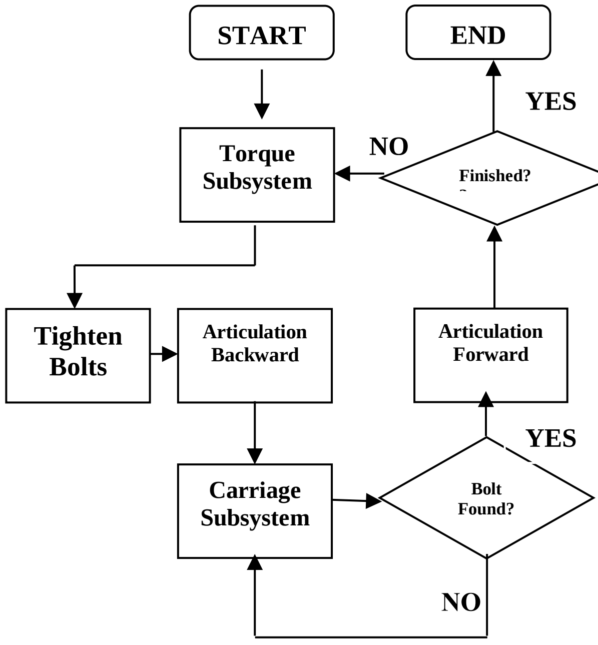

Fig. 2. Flowchart of Subsystem Flow.

Displayed in the flowchart above, it is easy to understand how the Robotic Flange Assembly is operating. First, begin with the start which will be the pressing of the button START on the LCD by the worker. Then, begin the process of tightening the first bolt with the Torque Drive Subsystem. Once that process is completed, move backwards on the Articulation Subsystem and then begin the Carriage Subsystem. If the position of the bolt was not found, keep repeating the Carriage Subsystem functions. If the position of the bolt was found, begin moving the Articulation Subsystem forward. If the process has finished and tightened the bolts on the flange by 30%, 60%, then 100%, then the process is finished. If the process is not finished, keep repeating the same steps until finally finished.

IV. Subsystem Detail

The Robotic Flange Assembly system is broken down into three main subsystems; the Torque Drive Subsystem, the Articulation Subsystem, and the Carriage Subsystem. This section provides a deeper, more technical look into the subsystems that make up the Robotic Flange Assembly. It will discuss the main roles and objectives of each subsystem, the order each subsystem goes when the system is running, and the main components that are comprised in each subsystem.

A. Torque Drive Subsystem

For the Torque Drive Subsystem, the main objectives consist of tightening and loosening bolts autonomously, using a motor to provide a desired torqueing force on the bolts in the flange. The motor will also have to apply torque in stages so that the flange is uniformly tightened and that it provides an even seal. The purpose of this subsystem is to ensure that all bolts on the flange are tightened evenly and properly according to the ASME bolt tightening standards and torque sequencing patterns.

The Torque Drive Subsystem is the first subsystem to run on the Cobot. This subsystem relies on the Articulation Subsystem to move towards the bolts on the flange so that torque can be applied.

The main components that are used in the Torque Drive Subsystem include the Milwaukee Power Drill's M18 Motor, the Cytron Brushed DC Motor Driver, and programmed logic on the Arduino. The M18 Motor was chosen for this subsystem because it has a stall torque of 40 ft-lbs., giving us our set torque value. The Cytron Brushed DC Motor Driver was chosen to drive the M18 motor because this will allow us to control the speed of the motor by pulse width modulation. Additionally, this driver was chosen because it has a max amperage rating that is twice as large as the motor's continuous current consumption under load which is recommended when selecting a motor driver. Since the Torque Drive Subsystem has a high demand for torqueing capabilities, this motor and driver are the perfect choices for applying the target torque on the bolts.

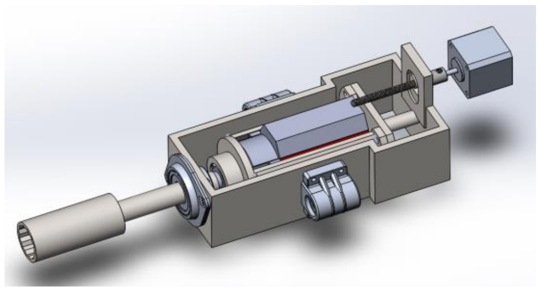

Fig. 3. Torque Drive Subsystem main body.

B. Articulation Subsystem

For the Articulation Subsystem, the main objectives consist of having to move the necessary components towards the bolt head as it is being tightened and move components away from bolt head after proper tightening has been completed. With the setup of the initial height being implemented, the purpose of this subsystem is to engage and disengage the socket with the bolt.

Between the three listed subsystems, the Articulation Subsystem falls in the middle. This subsystem relies heavily on the function terminations coming from the Torque Drive Subsystem and the Carriage Subsystem. When the Torque Drive Subsystem terminates, the Articulation Subsystem moves away from the bolt and then begins the Carriage Subsystem process. When the Carriage Subsystem terminates, the Articulation Subsystem moves toward the bolt to provide linear movement for the Torque Drive Subsystem. This process is constantly repeated until all bolts are properly tightened on the flange by ASME standards.

The main components that are used in the Articulation Subsystem include the Nema 17 Stepper Motor, Sparkfun's Easy Driver Stepper Motor Driver, and programmed logic on the Arduino. Since the Articulation Subsystem doesn't demand the high torque capabilities that the other two subsystems require, the stepper motor is the perfect choice for moving the subsystem back and forth.

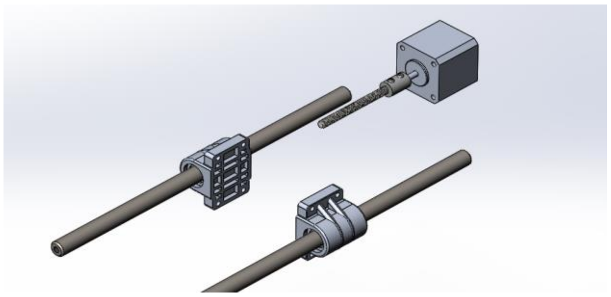

Fig. 4. Articulation Subsystem Parts

C. Carriage Subsystem

For the Carriage Subsystem, the main objectives consist of circumnavigating around the pipe and maintaining proper control to the motor in order to implement a specified torque sequencing pattern. The Carriage Subsystem shall also strive to maintain a static position while tightening bolts, withstand the torque produced by the motors in the system, and prevent slippage while on the pipe. The purpose of this subsystem is to ensure that we are following the specified torque sequencing pattern when circumnavigating around the pipe, all while moving to a specified location when instructed.

The Carriage Subsystem is the third subsystem to run on the Cobot. This subsystem is the main support for all the components and the main body for our prototype. The Carriage Subsystem relies on the Articulation Subsystem to move away from the bolts on the flange so it can begin its process of circumnavigating around the pipe and go to the next bolt in a specified pattern.

The main components that are used in the Carriage Subsystem include the Milwaukee Power Drill's M18 Motor, the Cytron Brushed DC Motor Driver, an optical encoder, and programmed logic on the Arduino. The Carriage Subsystem uses the same motor and motor controller that the Torque Drive Subsystem uses because both subsystems demand high torque capabilities. The optical encoder is used for the Carriage Subsystem because it will provide feedback about the position of the motor shaft. This will allow us to use this data to move to predetermined positions that will correspond to the location of each bolt. Having an encoder is critical for this subsystem because without it, the subsystem would not know where each bolt is located on the flange, therefore resulting in an increased error and miss rate in our system.

V. Hardware Detail

Each of the main major system components outlined in Section II, System Components, along with a few more additional components, will now be explained in further technical detail in this section.

A. Microcontroller

The Arduino Mega 2560 (ATmega 2560) has 256 KB of flash memory, 8KB of SRAM, 4 KB of EEPROM, and a 16 MHz clock speed, the highest speed grade amongst other Arduinos. It has an operating voltage of 5V and can receive input voltage values from 7V-12V. Each pin on the ATmega can provide or receive a maximum of 40 mA and has an internal pull-up resistor of 20-50 kOhms. The ATmega also has the function of operating at ultra-low power mode, running on a 1 MHz clock speed, operating voltage at 1.8V, and a current value of 500 microamps.

The ATmega 2560 has 54 digital input/output pins and 16 analog inputs. Some pins on the Arduino have specialized functions including pulse width modulation (14 digital pins), external interrupts, SPI communication, and many more. Since the Arduino is the heart of the Robotic Flange Assembly, all electrical components will be accurately wired throughout corresponding pins on the Arduino board.

B. M18 Motor and Driver

The M18 Motor that comes from the Milwaukee Power Drill delivers 500 in-lbs. of torque and up to 1800 RPM. It comes with two external red lithium-ion batteries that operate up to 18V; these motor batteries will be the main power supply for the Robotic Flange Assembly, delivering more runtime, power, and speed than standard lithium-ion batteries. These motor batteries are also optimized to work per charge and work over pack life, offering 3.0 amp-hours of runtime. Moreover, these motor batteries run cooler, preventing the overall system from overheating.

The Cytron Brushed DC Motor Driver is the motor controller responsible for driving the two M18 Motor's in the system. This motor controller has an operating voltage of 5V-30V and has a maximum current of 80A peak (30A continuously). Some additions of this motor controller include having reverse polarity protection, a PWM generator, operation between 3.3V-5V for logic level input, and speed control PWM frequency for up to 20 KHz. We use PWM for this motor controller because of power efficiency, controlling the speed of the motor, and having a controlled circuit.

The three main pins used on the Cytron Brushed DC Motor Driver are direction, pulse width modulation, and ground. The two motor direction pins are set to pins seven and ten and the motor enable pins are set to pins six and nine for running on the pulse width modulation signal.

C. Stepper Motor and Driver

The Nema 17 Stepper Motor is the perfect motor when it comes to positioning and repeatability. It is a bipolar 4-wire stepper motor that can be connected to the ATmega 2560 to obtain precision motor control. Some of the main features include having a step angle of 200-steps per revolution at 1.8° per step for smooth motion. The rated voltage is 12V, but it can operate up to 36V and the rated current is 2A per phase. The high-holding torque is 64 oz. in, 45 N*cm, allowing the stepper to incrementally step the next position. The stepper has a 5mm diameter shaft which allows for our lead screw to be attached directly on to it, and the motor width is only 42mm, making it compact in size and easy to mount onto the Robotic Flange Assembly.

Sparkfun's Easy Driver Stepper Motor Driver is the motor controller responsible for driving the Nema 17 Stepper Motor. This stepper driver is simple to use and compatible with anything that can output a digital 0V-5V pulse. The stepper driver requires a 6V-30V power supply to the motor and it can power any voltage of stepper motor.

This stepper motor driver provides flexibility and control over the Nema 17 Stepper Motor. Connecting the 4-wired stepper motor is very straightforward; the four wires that breakout from the stepper motor simply require connections to the A+ (black wire), A- (green wire), B+ (red wire), and B- (blue wire) leads that go to the corresponding phase outputs on the motor drive. These four wires are connected to pins 35, 37, 39, and 41 on the ATmega 2560. Pins 35, 37, and 41 are digital pins, and pin 39 is a digital pin that provides PWM for the motor to control the speed of the motor. The stepper motor driver also uses the step and direction pins. The step input uses a low-to-high transition and advances the motor one increment. The direction input will determine the direction of rotation of the motor.

D. Optical Encoder

The E3-2000 US Digital Optical Encoder is the component that determines where the position of the bolts are located on the flange. It can supply 2,000 cycles per revolution and can easily interface with the ATmega 2560 with an operating voltage of 5V. The optical encoder is designed to easily mount to and dismount from an existing shaft to provide digital feedback information.

This optical encoder utilizes a 5-pin standard connector, where the index output pin is optional. The first pin starts with Channel B, then VCC, then Channel A, then index, then ground. Channel A and Channel B are the two output channels that issue square waves in quadrature when the encoder shaft rotates. The square wave frequency indicates the speed of shaft rotation, whereas the A-B phase relationship indicates the direction of rotation. Channel B does not need to support interrupts whereas Channel A does. This is because when the encoder is mounted to the motor shaft of the Carriage Subsystem, the interrupt on Channel A is used to determine if the encoder position should be incremented or decremented (depending on the direction of rotation).

E. Touchscreen LCD

The 2.8" Touchscreen LCD by Adafruit is a resistive touch LCD that will be used to integrate a small user interface function that will be used for the initial set up of the Cobot. This display has a controller built into it with RAM buffering, so that almost no work is done by the microcontroller. It has an operating voltage of 3V-5V and an 8-bit digital interface.

For using the 8-bit digital interface, it is necessary to have eight digital data lines and four or five control lines to read and write to the display, making this a total of twelve data lines. Since this LCD is resistive touch, it allows for the user to use the LCD's touchscreen functionalities for the user interface function being implemented.

F. PCB Finalization

The Electrical Engineer's main task was to design, assemble, and test a printed circuit board (PCB). Besides software (detailed in section VI) being the one instance to give the Cobot meaning, energy and control that a PCB provides is the one to give it life. Due to mechanical aspects of the COBOT, like motor selection and overall dimensions that continued to undergo major changes in our first and second semester, finalizing the PCB proved to be quite challenging. We struggled to incorporate a PCB that would meet the demands for proper motor control and provide meaningful connections for a physical design that remained undecided. The design the PCB was made in parallel with component selection and rather than waiting to test our components, the PCB and components were ordered at the same time. Other than the footprints chosen for the input pins being too small, our first PCB appeared to be a success. Nonetheless, going through a second revision was needed because we had to accommodate amplification for a voltage signal that was coming from a strain gauge that had not been tested due to the incompleteness of the custom machinery for the physical component it required. In the end the small signal (7mV) was accompanied by a large amount of noise generated by the vibration of the motor and became indistinguishable. Fortunately, this did not require additional revisions to the PCB. .

The final PCB includes some unused analog and digital breakout pins in case we can incorporate an alternative to the stain gauge or to simply serve as backup. The main component of the PCB is the Atmega2560 microcontroller unit (MCU) and it drives the software in this system. This microcontroller was chosen for having more than fifty input and output pins. Having an abundance of input/output support allows our team to remain flexible and accommodate changes to the system when unforeseen problems arise and provides an easy transition in future growth. In addition, the ATmega2560 can support an external 16MHz oscillator crystal, which was desired to provide optimal performance. To communicate with this microcontroller, we added an Arduino In Circuit Serial Programming (ICSP) header. Without the ICSP header we would be unable to translate our programming software to the microcontroller driving our PCB. It is important to note the PCB must include a reset button for the MCU for efficient troubleshooting. To support onboard components that have a lower voltage rating than the Lithium ion 18V batteries that will be supplying power two voltage regulators were included to step down the voltage. A DC-to-DC converter was used for this since it allows us to step down the voltage from 18V to 5V for the MCU to provide the 5V to the optical encoder, LCD, and the (external) stepper motor driver. A second voltage regulator was added for the main purpose of supplying the peak voltage excitation of 9V to the strain gauge. However, due to the excessive noise distortion that deterred us form implementing the strain gauge, the 9V regulator no longer served a purpose. Lastly, it is very important to note that the 9V regulator was eventually repurposed, over heated, fixed, and then modified and is, at this time, now being used as an additional 5V regulator that supplies a constant 5V to the stepper motor driver.

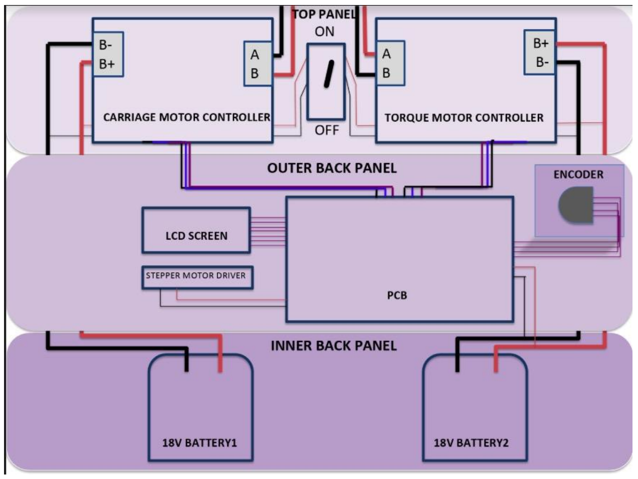

Fig. 5. System Block Diagram.

In the figure above you can see the inputs and outputs of our printed circuit board and how the components are arranged on the final design. The final design has three areas that are dedicated to electronic components, all on the back side of the system.

VI. Software Detail

The overall success of the Robotic Flange Assembly is its ability to circumnavigate around the pipe in an ASME torque sequencing pattern and tighten the bolts on the flange. The incorporated software that was written for this system will be able to complete the desired tasks.

The software which controls this system is written in C code and is approximately 350+ lines long. Since we are using an Arduino Mega 2560 as our microcontroller, the void setup() C function is called when a sketch starts. This function is used to initialize variables, pin modes, baud rate, start using libraries, et cetera. In this section of the code, we declare our instances of the libraries being used such as the PID library, the CytronMD library, the stepper library, et cetera. Additionally, the array of the bolt positions are declared and initialized to the starting position. All the major variables that are used with the microcontroller and motors are also declared here as well.

```
activateTorqueMotor();
    articulationSystemBackward();
        startCarriageMotor();
            articulationSystemForward();
```

Fig. 6. Primary Software Functions.

Displayed in Figure 6 above are the main functions that are being called in the Robotic Flange Assembly. To follow the order in which the subsystem processes begin, we start with the first function activateTorqueMotor(). This function is responsible for triggering the torque motor to spin to tighten the bolts on the flange. The set RPM of the motor is initialized to 30 and incorporates a delay before the next function is called.

Once the activateTorqueMotor() function is complete, it will then call the articulationSystemBackward() function. This function is solely responsible for disengaging the socket with the bolt. The RPM of the motor is initialized to 200 and set to iteratively move a fixed set of revolutions that equal to approximately two inches. Since the Articulation Subsystem is moving backwards at this point, the steps per revolution is being decremented to allow for the backwards movement.

Once the articulationSystemBackward() function is complete, it will then call the startCarriageMotor() function. This function is responsible for using an encoder and PID (proportional-integral-derivative) control to determine the position on the motor shaft for where the bolts are located on the flange. Since the encoder is not absolute, the location will not always be spot on and PID helps with getting that accurate location. The RPM on the motor is set to 35 and then incorporates a delay once a bis found. Once the Carriage Subsystem finds a bolt, it will proceed with the next function.

Once the startCarriageMotor() function is complete, it will then call the articulationSystemForward() function. The articulationSystemForward() is the same as the articulationSystemBackward() function. The only difference now is the Articulation Subsystem will be engaging the socket with the bolt to allow linear movement for the Torque Drive Subsystem.

While it is too complex to formally document the entire codes functionality here such as how the encoder is working or understanding how the proportional-integral-derivative control method is being used, it is extremely important to understand how the main functions call one another in a sequence while running. These sequences are what produce the order of the subsystems and allow for the Robotic Flange Assembly to follow ASME standards.

VII. Conclusion

This document provides a technical description of the Robotic Flange Assembly project, an interdisciplinary project of seventeen electrical, computer, industrial, mechanical engineering and computer science students. The resulting prototype is a collaborative robot that moves around a specified pipe size and tightens flange bolts in a predetermined sequence for an even seal. This device is meant as a safer and more consistent alternative to hand-tightening bolts on a flange. The device as it stands today is meant as a prototype for future groups to improve upon and redesign. As well, this document is meant to provide an insight into the project through the lens of the electrical and computer engineering team, and as such focuses more on the electrical and computer hardware aspects necessary to drive the device.

Acknowledgement

The authors wish to acknowledge the assistance and support of their sponsors at Siemens: Gerald J. Feller, Paul Zombo, and Matt Johnson. As well, the authors would like to thank their advisors and for their continuous guidance throughout the project. This includes Kurt Stresau and Mark Steiner (ME), Mike Conroy and Mark Heinrich (CS), Lei Wei, Samuel Richie, and Michael Young (ECE), and Luis Rabelo (IE). Finally, the authors would like to thank Bonnie Marini, whose constant support and leadership guided us through this last semester.

References

[1] L.Wei EEL4914 Microsoft Word document titled “2018-19 I-Design Projects 8_07 update.docx” College of Engineering and Computer Science, University of Central Florida Orlando, FL, Aug. 9, 2018.

[2] HardHat Engineer. Flange Bolt Torque Sequence and Torque Table - A Complete Bolt Tightening Procedure. [Online]. Available from: https://hardhatengineer.com/ flange-bolt-torque- sequence-table/ [Accessed 7th April 2019].