Electrical and Computer Engineering Department
California Polytechnic State University, Pomona

CAL POLY POMONA

SENIOR PROJECT REPORT
2nd Degree S-Curve Motion Controller with Definable Kinematics (CZMC)

By:
Mahan Bastani
Atsushi Domoyo
Daniel Kizito
Nolan Chang
Muhammed Abdal
David Avalos

Advisor:
Dr. Mohammed El-Hadedy Aly

April 23, 2019

Table of Contents

Chapter A: Introduction and Abstract

A motion controller is a control system in charge of the moving parts of a machine. A motion controller can range from a simple fan controller to multiple axis CNC machines, 3D printers robots, and more. Industrial motion controllers are usually application specific and cannot be easily re-configured to suit multiple purposes, as well as being hard to obtain for the maker community and closed-source. Open-source motion controllers have their own slew of disadvantages, mainly stemming from being micro-processor based which results in the controller being only able to do one task at a time which limits pulse generation and the number of axises, as well as a distinct lack of safety measures. These controllers also traditionally use a trapezoidal motion profile which tends to create high oscillation in the movement of the controlled device. The main goal of this project is to create an open-source motion controller that can solve most of the problems of both industrial and open-source motion controllers by implementing different features and using a kinematic based, optimized motion profile with the S-curve algorithm. An S-curve's defining point features two acceleration periods, followed by a period of no acceleration at its max velocity, and finally two decreasing deceleration period. Compared to a trapezoidal motion algorithm that uses linearly increasing acceleration and deceleration, the S-curve motion controller has much less jerking (oscillating) movement which results in smoother motion overall.

Chapter B: Background

The most important element of the project is implementing the S-curve model to improve the motion of the stepper motor. The S-curve profiles' acceleration and deceleration periods contain two polynomials resulting in a slower initial acceleration and deceleration. Compared to the linearly increasing acceleration and decreasing deceleration of a trapezoidal motion curve, the overall motion will have less of a jerk and be smoother. As can be seen in the figure below, the S-curve velocity model makes jerk (the derivative of acceleration) a finite value whereas in the trapezoidal model it is infinite.

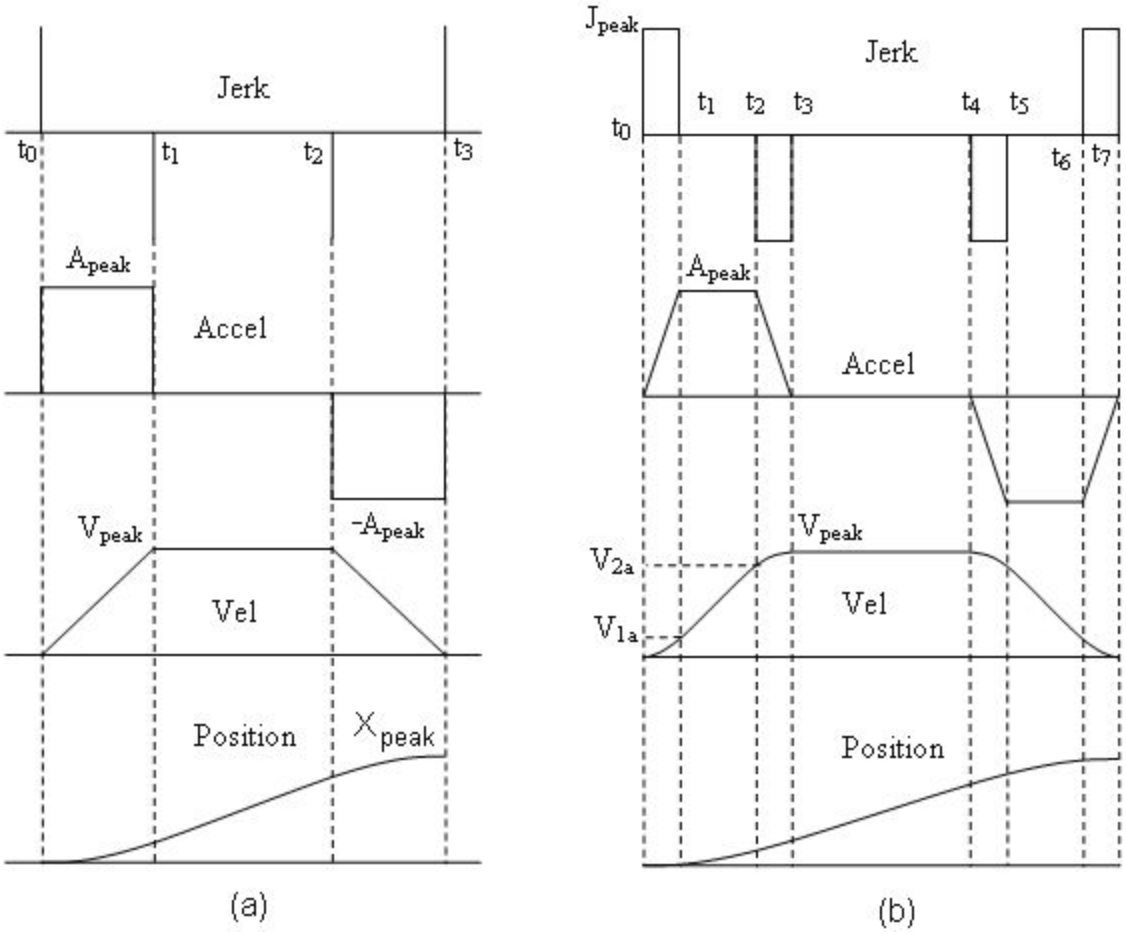

Fig. 1: Trapezoidal and S-curve models

Chapter C: Requirements and Specifications

Required Features

- The code must implement a 2nd degree polynomial S-curve algorithm
- A PWM module has to be able to convert values into pulses to be compatible with drivers
- The controller has to demonstrate a smooth motion profile using the S-curve algorithm and should have less jerk than a trapezoidal motion curve with the same specifications
- The controller has to be able to calculate complex kinematic calculations in real time for multiple axis of motion
- The calculations and step generations in all the axis needs to happen simultaneously for the motion to synchronize
- There has to be control measures that ensure maximum safety for user and constantly updates the safety status
- There has to be an interface for the user to communicate with the controller in real time and a queue to store additional instructions entered by the user.
- The code has to be open-source
- The project has to be easy to expand

Additional Features

- We need to be make updates to the firmware over the air.
- We need to be able to diagnose any issues remotely.
- We need to be add additional features to the controller in the future as time permits
- The motion controller operates on multiple axes of motion

Chapter D: Design

For this project, the PYNQ FPGA board was chosen because we needed an FPGA as well as an embedded processor. The FPGA would allow for real-time, high speed parallel calculations for trajectory correction, high frequency pulse generation, and safety logic. The FPGA would also allow us to have multiple axis motion with user definable kinematics, a key characteristic of the project. The embedded processor part of the PYNQ made the project easier to expand and allow the team to incorporate parts of existing infrastructure. The PYNQ board provided us with a dual ARM cortex processor as well as an Artix-7 FPGA and allowed us to run Linux, a lightweight and serviceable operating system.

Initially there were plans to use two seperate pieces of hardware: a Nexys 4 DDR board consisting of an Artix 7 FPGA as well as a Microblaze soft core processor or ARM core processors on a ZedBoard. The PYNQ board was ultimately chosen because it was easier to use because of the Python overlay architecture and the PYNQ's incorporation of both an FPGA and embedded PC.

The user initially interacts with the interface through the Linux operating system which sends instructions and parameters to the configurator, motion planner, S-curve, and supervisor blocks through high speed AXI buses. The configurator and motion planner provides the S-curve with the instructions for the movement requirements. The core of the project's design comes in the S-curve block which computes the desired motion from the positional specifications provided by the user. The motion profile generated by the S-curve is sent to a FIFO buffer, which passes the profile to the supervisor block . The supervisor block is in charge of a variety of features such as the step counter, pulse generation, homing logic, as well as safety. The supervisor block generates step pulses which are sent to the motor driver and the desired motion is achieved.

For the source of motion, we chose to use a motor on a rail for demonstration and testing purposes.

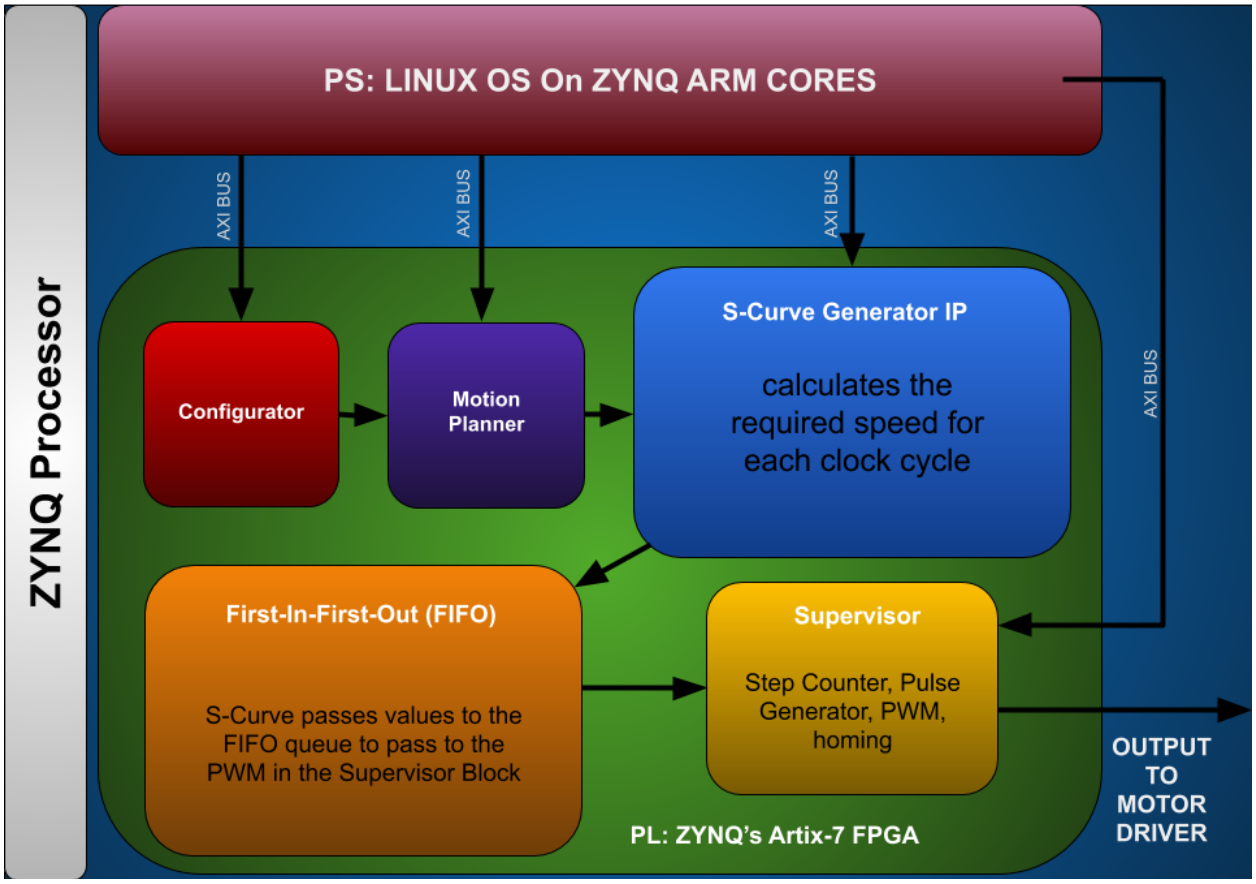

Fig. 2: Project Block Diagram

Chapter E: Test Results

Initially, we were having problems getting the S-curve block to work as expected. After testing using an Arduino, we learned that the formula we were using for the S-curve (taken from a research paper) needed to be adjusted. After doing some hand calculations and adjusting the formula, the following S-curve was produced on the Arduino.

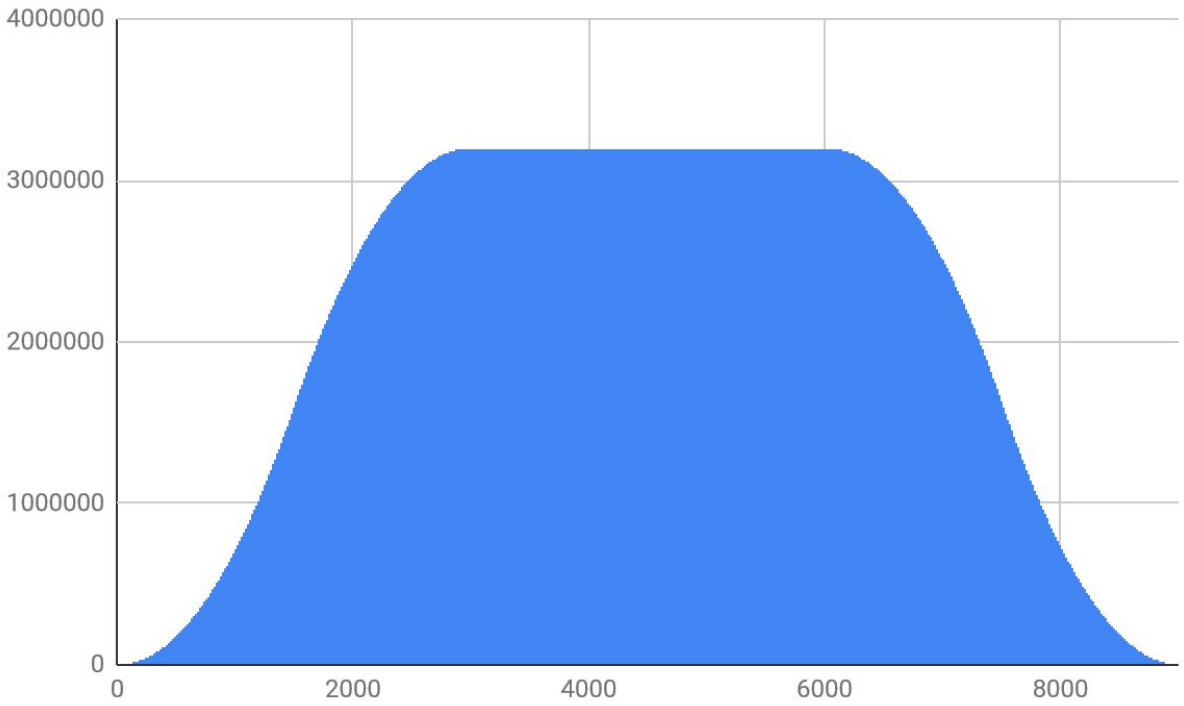

Fig. 3: S-curve Test using Arduino

The next step was to reproduce the same results on the PYNQ board, as can be seen below. Once that had been accomplished, the next step could be taken to work on PWM block, motion planner and so on.

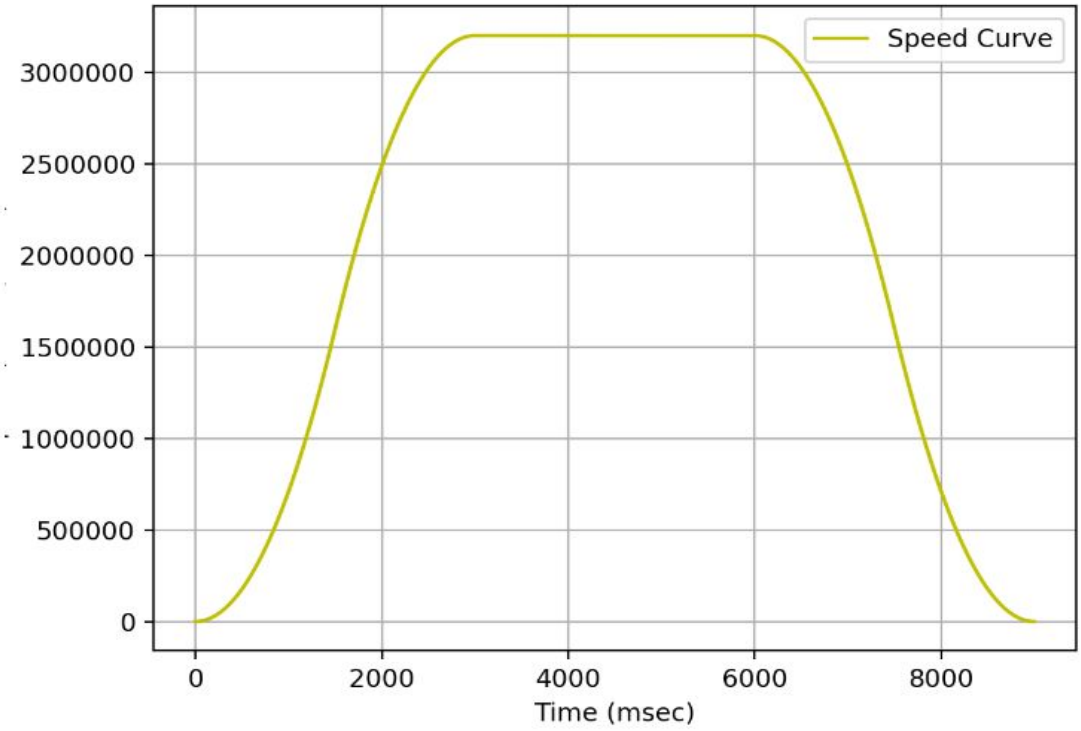

Fig. 4: S-curve Test using PYNQ

The power usage on the PYNQ is currently about 92% of available power at 1.671 W, which is relatively high, however, only about 318 mW is used by the motion controller's logic with the rest of the power being consumed by the operating system and I/O.

Total On-Chip Power: 1.819 Watts
CZMC Logic: 318 mWatts

Figure 5: Power usage on PYNQ

The total resources utilized by the motion controller is currently less than 30% of the available resources available on the PYNQ FPGA board, which has a relatively low amount of resources and available power compared to some of the other FPGAs on the market. The low resource usage in this controller allows for future feature additions without worrying about reaching the limit of resources available.

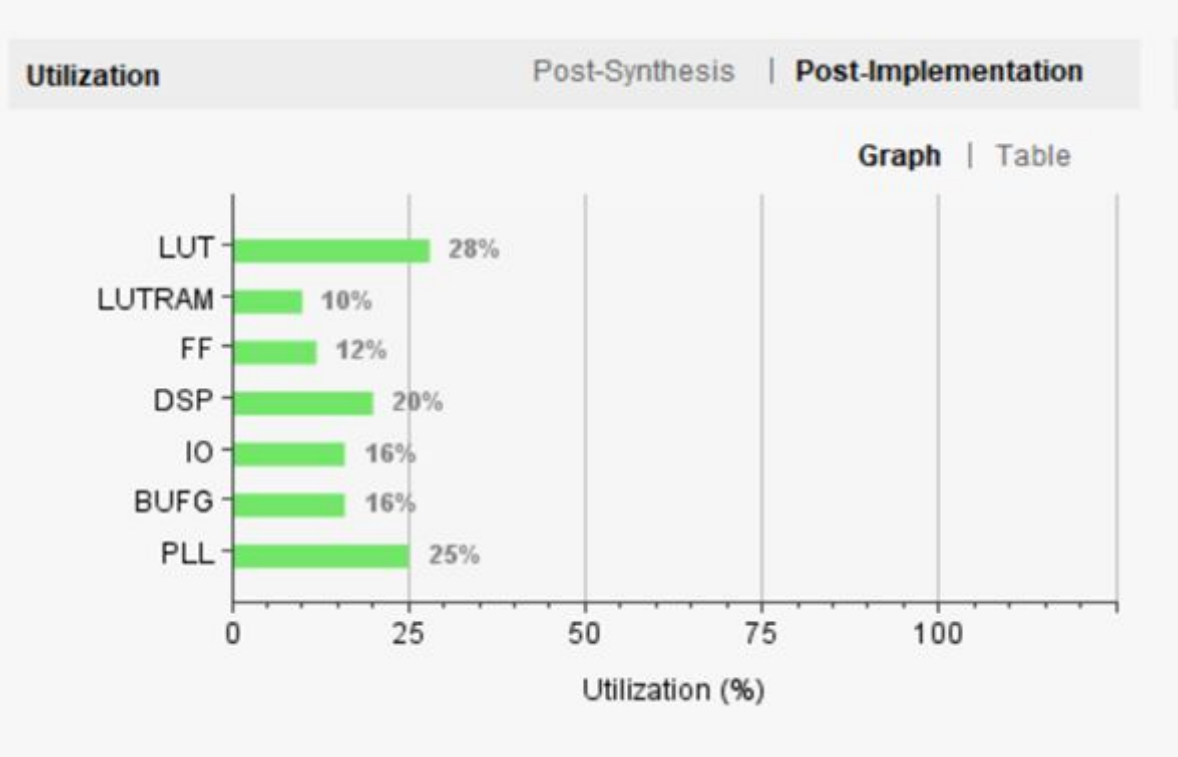

Figure 6: Device usage on PYNQ

Chapter F: Standards and Constraints

Primary Constraints

- Our biggest constraint has been time. It took us a long time to settle on a project idea. Since there were 15 people working with the advisor, there had to be separate projects. Almost everyone was uncertain about what area they wanted to work on. These issues delayed the start of the project by almost 2 months. That significantly reduced the time we had to complete the project. Also, since all of us have been taking other classes, the amount of time we have been able to commit to the project has been limited.
- Most of the members in our team were limited by the knowledge from previous coursework. This was a first-time project. Therefore, we had to research everything from scratch and figure out how to meet the requirements of our system. We had to ensure that we can move forward in spite of these limitations. We had to share whatever documentation or resources we had.
- We had to be able to communicate effectively to make sure that we didn't spend too much time working on a problem that someone has already solved.

Engineering Standards

- The engineering standards were important to ensure consistency and efficiency throughout the project. For example, we had to make sure our code for any part of the project could be used with little or no modification when we used different input parameters.
- A relatively common implementation process was used for the project, starting with research and analysis. Then the design stage began which was followed by prototyping. Integration and testing led to some changes in the design. That was followed by construction of the new design and further testing until we arrived at our final product.
- Documentation was of utmost importance to ensure that team members had material to reference and learn from. Different research papers could be compared to find the best approach.
- Different blocks of the project need to be connected and needed to work synchronously. Therefore careful naming of variables was important.

Public Health, Safety and Welfare Impact

- Public health should not be affected directly by our finished motion controller. It is a small scale project moving small payloads. Therefore there are no real risks associated with it. To make sure the system is safe to use, it was tested multiple times to make sure that all the components worked synchronously without any problems. A safety stop routine was added in case of emergencies. If applied to more large scale control systems, we will be able to achieve smoother motion with little or no jerk. That would enhance safety and public welfare.

Global Impact

- Since this is an open source, configurable motion controller, this can be used for small scale projects by students and researchers alike. This may allow users to add functionality and reduce cost, furthering improving the quality of future systems.

Societal Impact

- The motion controller can be modified and used for small scale projects by students and researchers. This may help enhance their understanding of how similar systems work. Since this is an open source project, our work can be used for other projects in the same field or in different ones, thus helping enhance technology.

Environmental Impact

- This project was not intended to address any environmental concerns. This project does not use any toxic materials. Materials used are quite common in various other fields. It does not contribute any significant amount of pollutants to the environment.

Economic Impact

- Since this motion controller is a cheaper option compared to more expensive and application-specific industrial controllers, it can be used by students and researchers for small scale projects in robotics or 3D printing etc. This can lead to improvements in our product as well as projects in those areas. These projects could be of economic benefit for startups who choose to work with similar technology.

Ethical Consideration and Professional Responsibility

- There were no major ethical considerations that needed to go into this project.

Chapter G: Project Planning and Task Definition

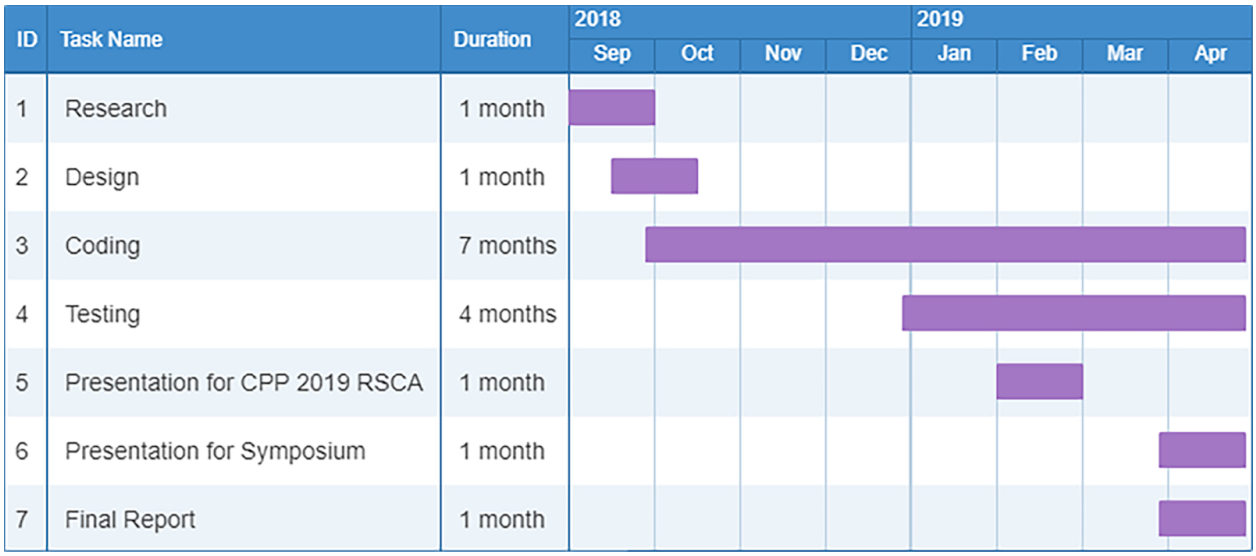

Fig. 7: Gantt Scheduling Chart

Task Distribution:
All members were involved in the initial research phase of the project and everyone learned how to work with the PYNQ before the motor was acquired

Mahan - created the concept and design of the project and took on a leadership role, worked on the motion planner and provided team members with assistance on coding of the project, worked on the testing of the PYNQ and motor and most of the different coding blocks, as well as created the case for the prototype and provided most of the prototype parts, presented at the College of Engineering showcase

Atsushi - primarily involved in the coding aspect of the project, worked on the S-curve, PWM, safety status controller, and homing logic as well as testing the PYNQ and motor, presented at the College of Engineering showcase

Daniel - worked together with Atsushi on the brunt of the coding for the project on the S-curve, PWM, and status controller, created the motion planner and helped with the homing logic, presented at the College of Engineering showcase

Nolan - wrote some of the initial code for the S-curve and PWM, created the poster presentation and abstract for 2019 CPP RSCA and the symposium, helped create the final report, presented at the College of Engineering showcase

Muhammed - Researched how to create a FIFO in Vivado HLS, wrote and organized the final report

David - Helped out with the final report

Chapter H: Conclusion

All of the major goals of the project have been accomplished. A fully operational system for single axis motion has been achieved. A 2nd order S-curve motion profile has been implemented in Vivado HLS. The PWM pulse generator has been highly optimized in RTL. A trajectory planner has been implemented successfully in Python. A safety emergency stop supervisor has been programmed in RTL. Homing logic has been developed as well as a DMA(Direct Memory Access) based planner buffer. All of these steps have led to a fully operational motion controller.

There are plans to further optimize the system and add more functionality in the near future. One of the goals is to implement higher order S-curve motion profile with look ahead trajectory planning. There are plans to synchronize multiple axis with kinematic/ inverse kinematic calculator. Also combining some logic blocks will lead to optimized usage of resources.

Looking further down the line, we would like to design a core board along with multiple baseboards for different applications. We would also like to create an Application Programming Interface (API) before writing native Linux drivers. The final goal would be to test and launch the motion controller on KickStarter.

References

Create a custom PYNQ overlay for PYNQ-Z1. (2018, March 15). Retrieved from http://www.fpgadeveloper.com/2018/03/create-a-custom-pynq-overlay-for-pynq-z1.html

Overlay Tutorial¶. (n.d.). Retrieved from https://pynq.readthedocs.io/en/v2.1/overlay_design_methodology/overlay_tutorial.html

Nguyen, K. D., Ng, T., & Chen, I. (2008). On Algorithms for Planning S-Curve Motion Profiles. *International Journal of Advanced Robotic Systems, 5*(1), 11. doi:10.5772/5652

Yang, G., Ye, Z., Pan, Y., & Ma, Z. (2012). The Implementation of S-curve Acceleration and Deceleration Using FPGA. *TELKOMNIKA Indonesian Journal of Electrical Engineering, 11*(1). doi:10.11591/telkomnika.v11i1.1897

E. Cigan and A. Cozma, "FPGA-Based Systems Increase Motor-Control Performance | Analog Devices", *Analog.com*, 2018. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/fpga-based-systems-increase-mc-performance.html

"howitworks [Smoothieware]", *Smoothieware.org*, 2018. [Online]. Available: http://smoothieware.org/howitworks.