**SYSTEM ALTERNATIVES**

**FPGA**

* Can be expensive
* Difficult learning process
* Includes a lot of unnecessary components for this project since its a development board
* Precise
* Development time intensive
  + Memory management
  + Build infrastructure

This control system consists of a field programmable gate array (FPGA). This control system would be the most accurate due to the optimization capability of an FPGA. The system should rotate to the set positions with the most precise accuracy. Compared to the other alternatives this control system is the closest to the analog level due to the development code environment being in Verilog HDL.

This system would have the most optimized capability of the other alternatives, as the analog to digital conversion process is low on an FPGA. The research of prototyping this robotic positioner on an FPGA is limited but doable. The signals from the encoder can remain close to an analog value and can be utilized easily when implemented. The system itself would be more expensive to prototype based on what development parts are built into the FPGA.

However, the memory management and infrastructure is not built in on an FPGA. The development time for this control system would be intensive and not viable within our time frame for a completed prototype. This FPGA control system does not allow for future implementations by the client and the development will be hard to code.

**PLC**

* Most expensive choices
* Easy to implement
* Makes sense if everything else in the factory uses PLC but literally only 1 machine uses PLC in house
* Client have no plans of automating everything in the future

This control system uses a programmable logic controller (PLC). The use of PLC’s in an industrial environment is mainstream and widely supported. As for the development the PLC programming language is straightforward and supports memory of a recall position. The PLC also supports additional indicators such as a running and stop light indicator. These components provides a safe working environment for the operator.

However, the PLC is the most expensive control system of the alternatives. This system would have the most in-house support based on local suppliers in Saskatoon. This solution will also require more research, development, and prototyping time to develop the user application. The use of the PLC is not beneficial for the client due to no future plans of automating all or another machine in the shop. The PLC will not be utilized to its full function since the shop only has one other PLC controlled machine.

**Microcomputer**

* Problems with analog to digital conversion
  + Precision problems
* Requires additional sensors for the ADC
* SD card will wear out over time
  + Will require the client to reupload OS and program to new SD if broken

This control system consists of a microcomputer to drive the rotations for the robotic positioner. The user would press a button on the control panel and a signal is sent to the control box which houses the microcomputer.

The microcomputer control system would allow for a simple development time since the code used can be high level for this functionality. The microcomputer will have more than enough processing power to rotate the positioner in specific positions.

However, this control system does not have the most accurate analog-to-digital conversion. This system would require additional sensors to provide an accurate representation for the rotations at specific degrees. In addition, if the SD card were to fail the entire operating system and rotation program would have to be re-uploaded to a new SD card to fix this solution. This solution would also require more research, development, and prototyping time to develop the user application, and configure the precision of the rotations. The microcomputer will require a heatsink to dissipate heat due to the enclosure and industrial environment.

**Microcontroller**

* A simple microcontroller will be able to do the job
* Rapid deployment of new code during prototyping and can be adjusted as necessary to refine the outcome
* Allows for easy physical button configuration
* Research and development for this implementation should be relatively straightforward, given the background and experience of the design team
* Code development process may also take longer than anticipated, as the process of implementing all the required and target functionality may be more difficult than anticipated
* Allows for a wide range of controls for the rotator
* Ease of use/learn
* Abundant amount of GPIO pins
* Inexpensive

This control system employs a microcontroller as the control box unit. Data is acquired from the encoder and is converted to a digital signal which is then processed by the microcontroller. The control panel will have the physical buttons of which and how much the rotation will be that sends a pressed signal to the microcontroller and rotates the motor.

The microcontroller allows for a wide range of control over the rotational process. One such option is the recall position which will rotate the positioner back to the original home position and will always have this position set as home. This will prevent operators from having to recalibrate the home position each time they turn on the robotic positioner. Another rotation position would be rotating 45 degrees clockwise or counter-clockwise. The application of having specific rotations provides the ideal tool for the robotic positioner.

If the microcontroller is used in the final solution, the main consideration will be compatibility and ease of development. Choosing a microcontroller with an abundant amount of library sources online would aid in the development process. The microcontroller in this solution would be required to provide adequate GPIO pins for the encoder, push buttons and the motor.

This system alternative can be implemented relatively inexpensively, as a simple microcontroller is needed for the control box. Employing a microcontroller also allows for rapid deployment of new code during prototyping and the position settings can be adjusted as necessary to refine the outcome. The microcontroller also allows us to use physical push buttons which are easy for the operators to use. The research and development for this implementation should be relatively straightforward, given the background and experience of the design team. Should the project be reproduced, the code should be exactly the same with minor changes to the angles of rotation.

However, this solution will require the encoder to determine the appropriate signals sent and read by the analog-to-digital converter on the microcontroller. The code development process may also take longer than anticipated, as the process of implementing all of the required and target functionality may be more difficult than anticipated.

**Comparison of Alternatives**

**VERSION 1**

To compare the proposed system alternatives, we employed a weighted decision matrix. The five main categories are Cost, Development, Compatibility, Memory Management, and Reliability. Each of the categories were assigned an overall weight out of 100%. The categories were ranked from highest to lowest in the following order: Cost, Development, Compatibility, Memory Management, and Reliability. The high-level decision matrix is shown below.

ADD DECISION MATRIX HERE

Each category is then further broken down into subcategories and again each subcategory is given a weight out of 100%. Points were assigned to each subcategory ranging from 1-4, with 0 being the worst performer and 4 being the best. These numbers are assigned to each alternative independently. The points were assigned based on the group discussion and research into the different methods.

**VERSION 2**

**DETAILED BLOCK DIAGRAM - SOFTWARE (DON’T NEED IT ANYMORE)**

**Insert block diagram**

The software block diagram consists of five main states: e-stop, power, standby, reset, and running state. The system will operate for the majority of the time in standby and running state. The standby state is a safety feature that requires the operator to hold down a button to enable the system to be in running state.

The system will require the E-stop button to be released to enable the system to go into the power state. Once the power switch is on the system will go into standby state where the operator can depress a button to jump into the running state.

In the running state, the operator is able to press any of the rotational buttons on the control panel. This will allow the operator to rotate the robotic positioner either clockwise or counter-clockwise and at specific angles. In addition, the home button will rotate the positioner to the original calibrate start position where the frames can be mounted on the robotic positioner. When the system exits the running state to return to the standby state, the operator would have depressed the button.

If the E-stop were pressed during the standby or running state it would jump into the E-stop state and would wait there until the E-stop button is released. Once released the system will go into the reset state and will require the operator to press the reset button to enable functionality again. After the reset button has been pressed it will enable the system back into the standby state.

**SUBSYSTEM FUNCTIONALITY (DO I NEED THIS?)**

**Microcontroller**

In this system, the microcontroller is the decision unit. It takes in signals from the control panel’s buttons and the encoder for the motor’s position. As discussed in the previous section the microcontroller will implement a number of software functions to govern the system.