Hamzah Khan and Jeewan Naik
Engineering 155: Microprocessors
Prof. Spencer, Harris
9 December 2016

Two-Axis, Internet Controlled, Pneumatic T-Shirt Cannon

Abstract

This final project report describes an internet-connected two-axis, pneumatic t-shirt cannon that has shot 100-150 feet at pressures between 40 and 50 psi. The project divides into four subsystems: the pneumatic cannon, azimuthal control, polar control, and the control input interface. This document describes the design decisions made in the proposal and then changes made during the final implementation of each subsystem and the rationale for those decisions in such a way that the reader may clearly understand how to implement a two-axis cannon of his own. Furthermore, it describes the results of the project and potential improvements to its design.

Table of Contents

Introduction

The idea of a pneumatically powered cannon is not new; many sporting events use t-shirt cannons to distribute small quantities of t-shirts into large crowds. Furthermore, the maker community has published a number of articles and tutorials on creating such a cannon, and *Popular Mechanics* in particular has a well-regarded guide[1] that was further improved in an Instructables tutorial.[2] The tutorial largely guided the initial design of the pneumatic cannon (shown in Figure 1). However, the rest of the design subsystems were developed by the team. These subsystems included the "Lazy Susan"-style azimuthally rotatable base, the "See-Saw" polar control mechanism, and a live-streaming "click-to-aim" webpage. The relationships between the project components are shown in Figure 2.

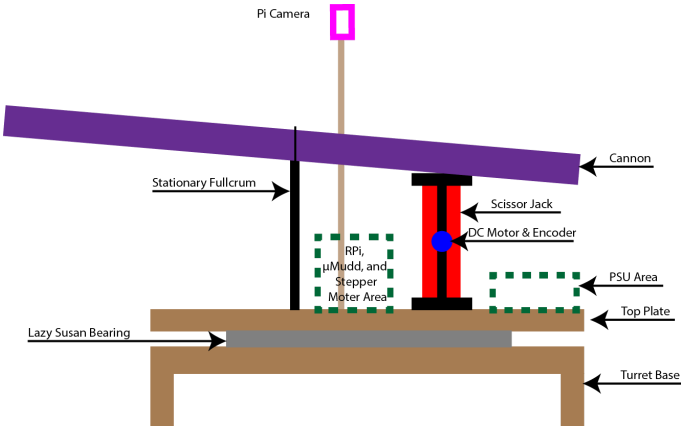


Figure 1: Full diagram of the initial design of a pneumatic cannon on a two-axis turret

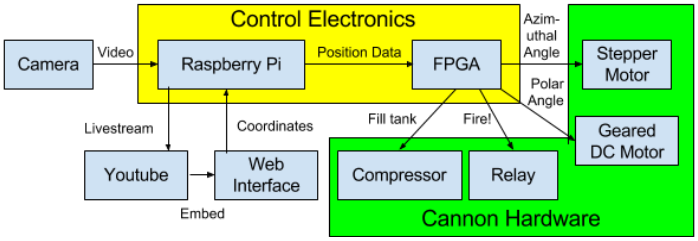


Figure 2: Initially proposed system design

Pneumatic Cannon Subsystem

The cannon is 6' long and consists (from bottom to top in Figure 3a) of an aluminum tire valve stem (Figure 3b), PVC air tank, solenoid relay (Figure 3c), and PVC barrel, as described by *Popular Mechanics*.[3] The air tank and barrel are sourced from schedule 40 PVC, which is rated at 220 psi, and the solenoid valve is rated to 150 psi. The tire valve interfaces with a 12V tire inflator[4] that can pressurize up to 130 psi, although it slows down significantly above 50 psi so the tank pressure will never exceed the part ratings. The solenoid is rated for 24VAC, but it can acuate with 12VDC input, although using DC

[1] http://www.popularmechanics.com/home/how-to/a7298/build-your-own-t-shirt-cannon/
[2] http://www.instructables.com/id/Self-Contained-Pneumatic-T-Shirt-Cannon/
[3] http://www.popularmechanics.com/home/how-to/a7298/build-your-own-t-shirt-cannon/
[4] http://www.homedepot.com/p/Husky-12-Volt-Inflator-HD12A/203356023

causes significant flyback signals through the power supply. The parts we used can be found in Appendix B.

The FPGA controls the solenoid valve and compressor using electromechanical relay circuits as described in Appendix C. When the FPGA drives the base of a 2n3904 transistor high (through a 1 k□ resistor) a 180 mA current from collector to emitter switches the relay and sends power to the component. By changing the time that the compressor relay is on, the FPGA can control the amount of pressure in the tank. Once the FPGA fills the tank and sets both the azimuthal and polar angles, the FPGA fires the cannon by enabling the solenoid valve for 2 seconds. A link to the FPGA code can be found in Appendix A and the RTL netlist viewer output is in Appendix D.

Because of flyback currents from the solenoid valve, the FPGA struggled to control the relays as described in the initial design. The main problem was that the flyback transients pulled the reset pin high, restarting the FPGA. Switching the relays from a different power supply didn't fully remove the problem, proving that it was flyback from the solenoid valve rather than the relay. A simple flyback diode would allow the ground to spike all the way to 12 V before turning on, so a second 12 V power supply (or a 18 VAC power supply) was needed to fully isolate the FPGA fro mthe valve.

Figure 3.a: The full cannon with the tank on the bottom and the barrel on top

3.b: The tire valve at the base of the tank that interfaces with the compressor

3.c: The sprinkler solenoid valve which fires the cannon

Azimuthal Control Subsystem

The base of the cannon rotates to adjust its yaw. The top plate of the base connects to the bottom plate by a Lazy Susan bearing (Figure 4). The initial design contains an axle hole in both parts of the Lazy Susan for a high-torque stepper motor. The motor enclosure is mounted to the top plate and it's axle slots securely into the bottom plate so the top plate rotates as the motor turns. With a stepper, the FPGA does not need encoder feedback to track turret rotation. The FPGA drives the azimuthal stepper motor with a high-current H-bridge, the L6205N from STMicroelectronics.

The FGPA stepper Verilog module takes an azimuthal position from the Pi and steps through a table with the correct H-bridge signals to turn a stepper at 700 pulses per second, which corresponds to maximum torque. A reset pin on the FPGA manually zeroes the azimuthal angle state. The FPGA module

can be found in Appendix A, and diagrams showing how the module works can be found in Appendix D and E.

However, during assembly, the stepper motor underperformed in its torque, so the final design replaces it with a custom encoder and 12V DC motor with a gearbox sourced from a power drill. The homemade encoder uses a series of electrical contacts around the outside of the bottom plate to pulse 3.3V when a contact shorts with an input lead as the wheel turns (Figure 5). The final design uses an instance of the same Verilog module and H-bridge as the polar control subsystem. The revised design met expectations until a short burnt out the only working H-Bridge.

Figure 4: The turret base with the lazy susan bearing and the motor axle mount in the center

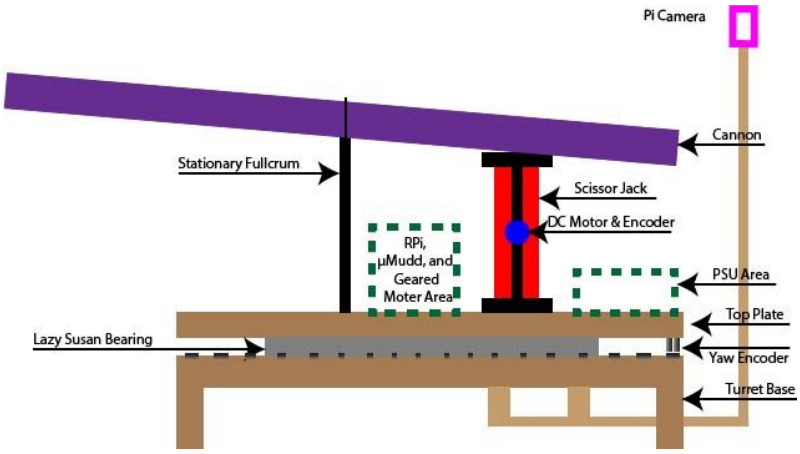


Figure 5: The full design with changes from the initial design in Figure 1 shown

Polar Control Subsystem

The pitch of the cannon is changed by rotating the axle of a car jack (see Figure 5). The weight of the cannon balances on a fixed fulcrum so that the motor does not lift the cannon, but only needs to overcome the friction of the car jack. The DC motor has 99:1 gearing, which provides enough torque for this application. The main constraint with this design was that the head of the car jack screw moves relative to both the base and vertical axes: the motor and encoder can't mount to the table and must mount to a moving joint on the car jack (Figure 6) . As shown in Figure 6, we mounted both the encoder and the DC motor directly to the axle so they could all move together. The DC motor mount was made with a thick piece of sheet metal screwed into the front of the motor and formed to hold onto the jack. The hinged bar setup below the encoder allows the axle of the encoder to rotate while the enclosure doesn't rotate (allowing it to keep accurate track of position). In practice, the hinged bar and wheel was changed to a thin piece of metal that could slide easily in the track because that's what was available.

In hardware, the DC motor is powered by a TB67H400AHG H-Bridge mounted to a custom breakout board. The H-Bridge is wired in its high-current configuration which provides bi-directional control for a single motor. The encoder shorts a pulled-down input pin with the 3.3 V rail 24 times per rotation. The FPGA module can be found in Appendix A, and diagrams showing how the module works can be found in Appendix D and E.

On the FPGA side of the subsystem, the polar motor uses the exact same module as the azimuthal control system. That is, the FGPA receives a desired position from the Pi and turns the motor on until it receives the correct number of pulses from the encoder. The FPGA keeps track of the scissor jack position and also watches a reset pin which allows the current position to be set back to zero. The encoder bounces for 1-5 ms and the motor turns at 100 rpm so the module needs to sample at a speed between 5 ms and 25 ms for adequate debouncing.

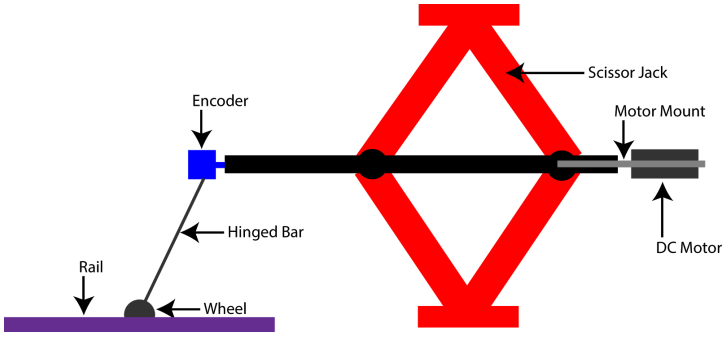


Figure 6: Diagram of the initial design of the car jack polar control system. In the final design the hinged bar and wheel was switched to a flexible piece of metal.

Control Input Subsystem

The control input subsystem encompasses receiving the input from the user, deriving motor positions and compressor time from the input streams and sending the data to the FPGA. To get input from the user, the cannon uses a stationary Raspberry Pi Camera Module[5] securely attached to the bottom plate and the Pi hosts a web page (Figure 7b). The webpage contains an embedded YouTube livestream from the camera using the raspivid and ffmpeg video-encoding libraries. Unfortunately, the livestream has a 20 - 30 second delay due to network problems and YouTube's streaming software. A JavaScipt EventListener registers click positions on the livestream and maps the coordinates to possible positions for the azimuthal and polar motors (Figure 7a). In addition, the force input approximates the distance of the shot by setting the enable time of the compressor, which in turn sets the pressure in the tank. This data is sent back to the Raspberry Pi using a CGI script, and the Pi sends the translated values to the FPGA over SPI so that the cannon aims, fills, and fires as described in previous subsystems. All of the code can be found in Appendix A.

[5] https://www.amazon.com/Raspberry-Pi-Camera-Module-Megapixel/dp/B01ER2SKFS/ref=sr_1_fkmr0_3?s=pc&ie=UTF8&qid=1481312930&sr=1-3-fkmr0&keywords=respi+cam+8mp

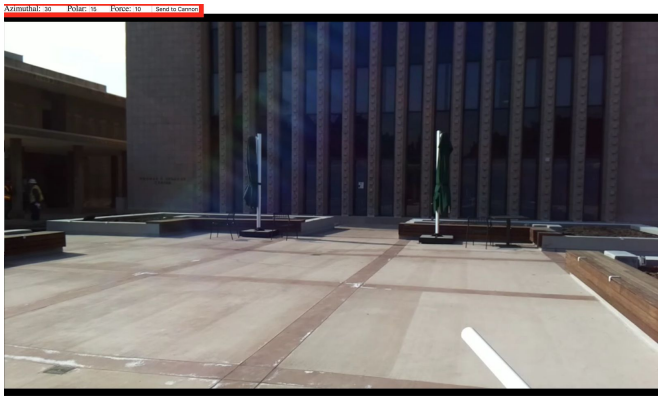


Figure 7a: The form for the three cannon parameters
Figure 7b: A screenshot of the Pi-hosted webpage with the value form highlighted

Results

The control interface and data management, including the Raspberry Pi camera's live stream to YouTube and embedding, work as intended. The point-and-click interface accurately maps points on the screen to motor positions on both axes. Furthermore, the Raspberry Pi accurately receives and transmits the data to the FPGA, which also simulates and runs correctly on hardware. With regards to pneumatic cannon functionality, the cannon shoots the t-shirt at a distance of approximately 100-150 feet at pressures between 40 and 50 psi. The distance was less than expected, but well within project goals. In addition, the cannon demonstrated polar angle control given webpage input. Unfortunately, the failure of the stepper motor to provide adequate torque and the subsequent H-Bridge burnout caused an inability to demonstrate azimuthal control, although all the code and circuitry was there to support it. Two main issues plagued development, especially with regards to the motors: high currents and back-EMF from motor and solenoid use. Many of the electronics for the project come with current protection, which both protected vital equipment from shorts and shut off vital equipment when inductive equipment, including motors and the solenoid valve, were enabled. The back-EMF from these electronics also reset the FPGA at undesirable times.

Future Extensions

Now that proof of concepts for each of the components have been completed, there are a few features which will be easy to add in the future. On the software side, the website targeting can be combined with an artificial intelligence face detection system to send t-shirts to people (if the user can estimate how far away they are from the stream). Also, with a different streaming service, the 30 second delay in the video can be eliminated. The biggest mechanical improvement is including a good gearbox between the yaw motor and the base so an H-Bridge doesn't need to deliver so much current. Finally on the electrical side, now that the 12 VDC limit is lifted, a faster compressor will reduce the delay between Pi input signals and firing. Also, a separate AC power supply for the solenoid valve will eliminate the back-EMF problems that kept resetting the FPGA when the cannon was fired.

Conclusion

This final project report describes the development process for an internet-connected two-axis, pneumatic t-shirt cannon that can shoot 100-150 feet at pressures between 40 and 50 psi. The cannon can be subdivided into four subsystems: a "*Popular Mechanics*"-style pneumatic cannon, "Lazy Susan"-style azimuthally rotatable base, the "See-Saw" polar control mechanism, and a live-streaming "click-to-aim" webpage. While the control input interface worked exactly as intended, other subsystems relied heavily on inductive motors and valves, which introduced high currents and back-EMF, each of which caused equipment to turn off due to current protections and unintended device resets. While the issue remains across each subsystem, the polar control works properly in most cases without issue, except that the position resets when the cannon is fired. High currents caused key components in the azimuthal control subsystem to fail, so the subsystem was not ready to test in time. Lastly, the pneumatic cannon worked properly, except for the solenoid valve control, which reset the FPGA with back-emf whenever it was fired. However, these issues are easily solvable new H-Bridges and an AC power supply for the solenoid valve.

Appendices

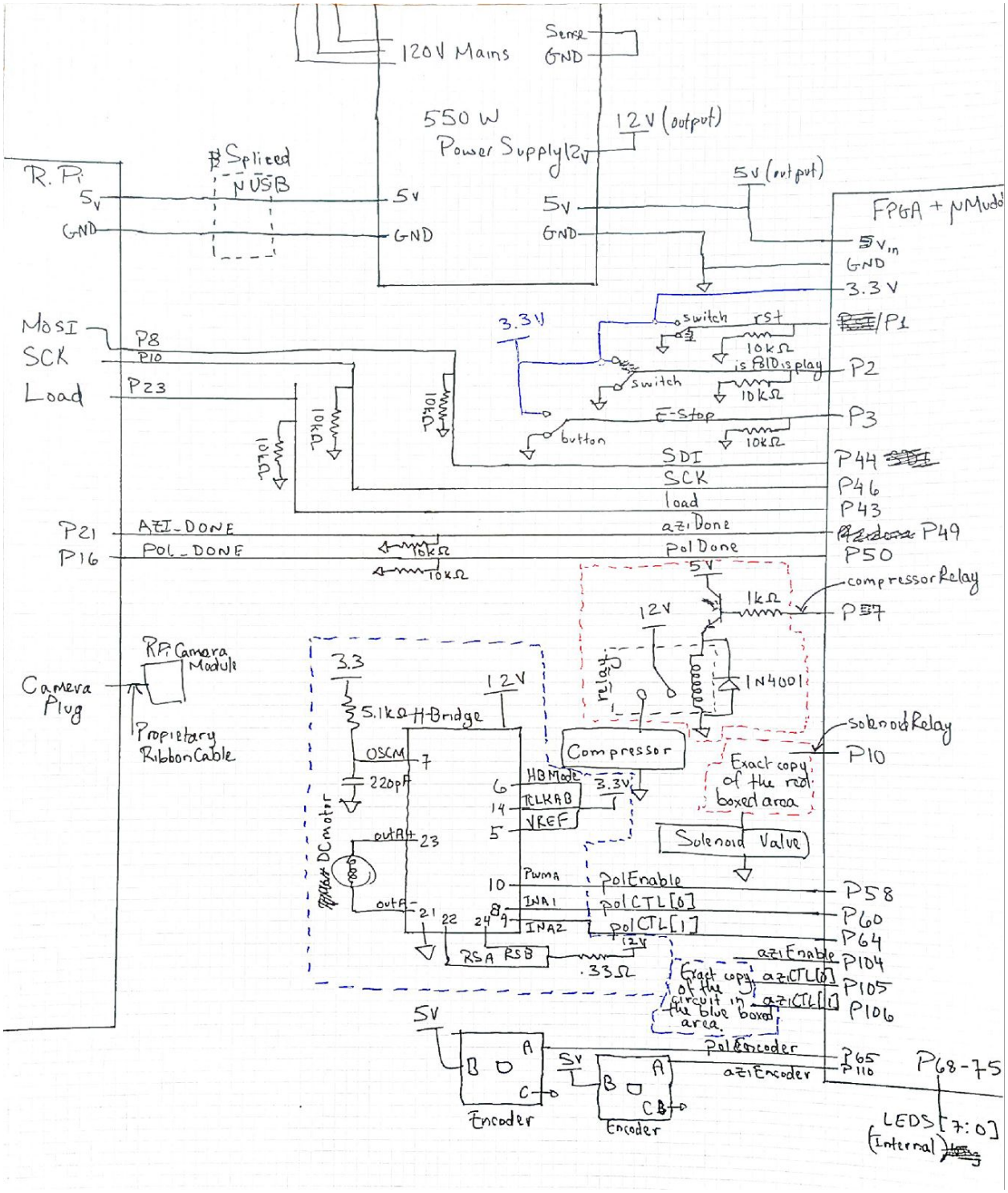
A. Source Code

All source code is hosted at https://github.com/khanh111/shirt-cannon.

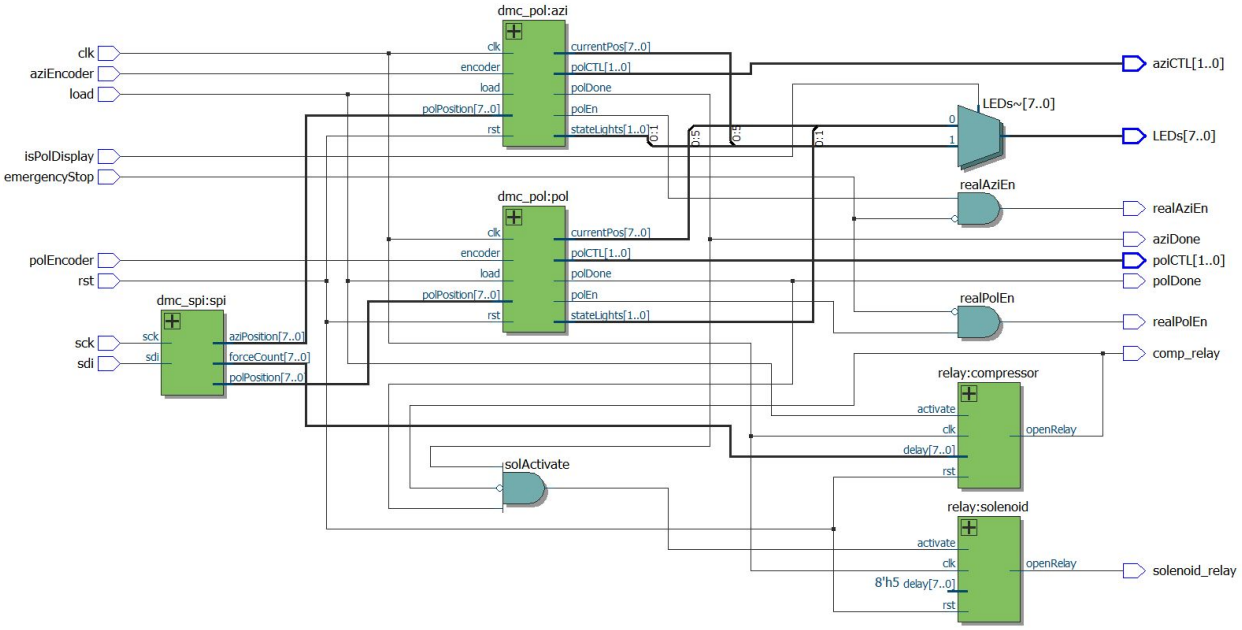
B. Parts List

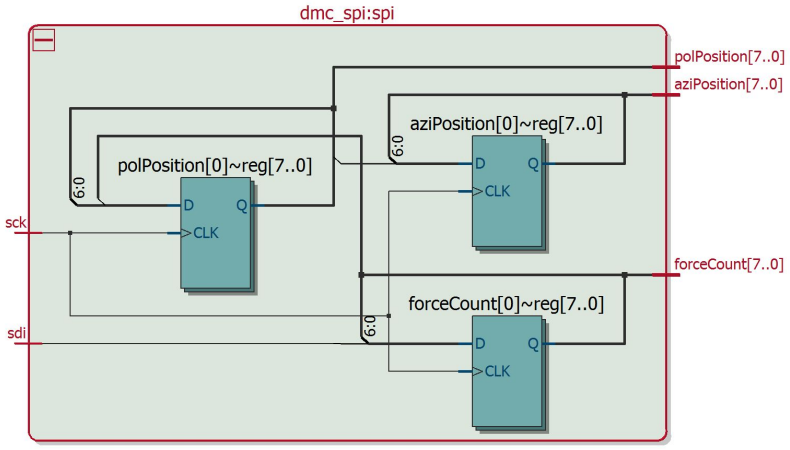
Part	**Source**	**Price**
PVC Pipe & Fittings	Lowe's	$70
PVC Glue	Lowe's	$8
Sprinkler Solenoid Valve	Home Depot	$24
Tire Valves	Amazon	$9
Wood	Lowe's	$30
550W PSU	FabStudio	$0
Pi Camera	Amazon	$25
Compressor	Home Depot	$25
MuddPi Board	Sam	$0
Raspberry Pi	MicroPs	$0
Tire Jack	Amazon	$15
DC Geared Motor	Pololu	$22
Stepper Motor	Pololu	$16
12 V Drill Motor	Harbour Freight	$48
Encoders - 2x	Digikey	$2
TB67H400AHG H-Bridge - 2x	Digikey	$12
L6205N H-Bridge - 2x	Digikey	$12
ProtoBoards - 2x	FabStudio	$0

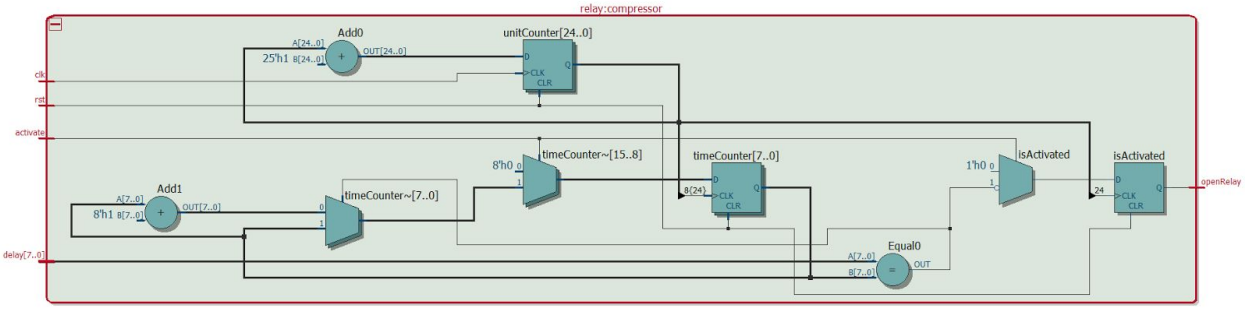
C. Circuit Diagram

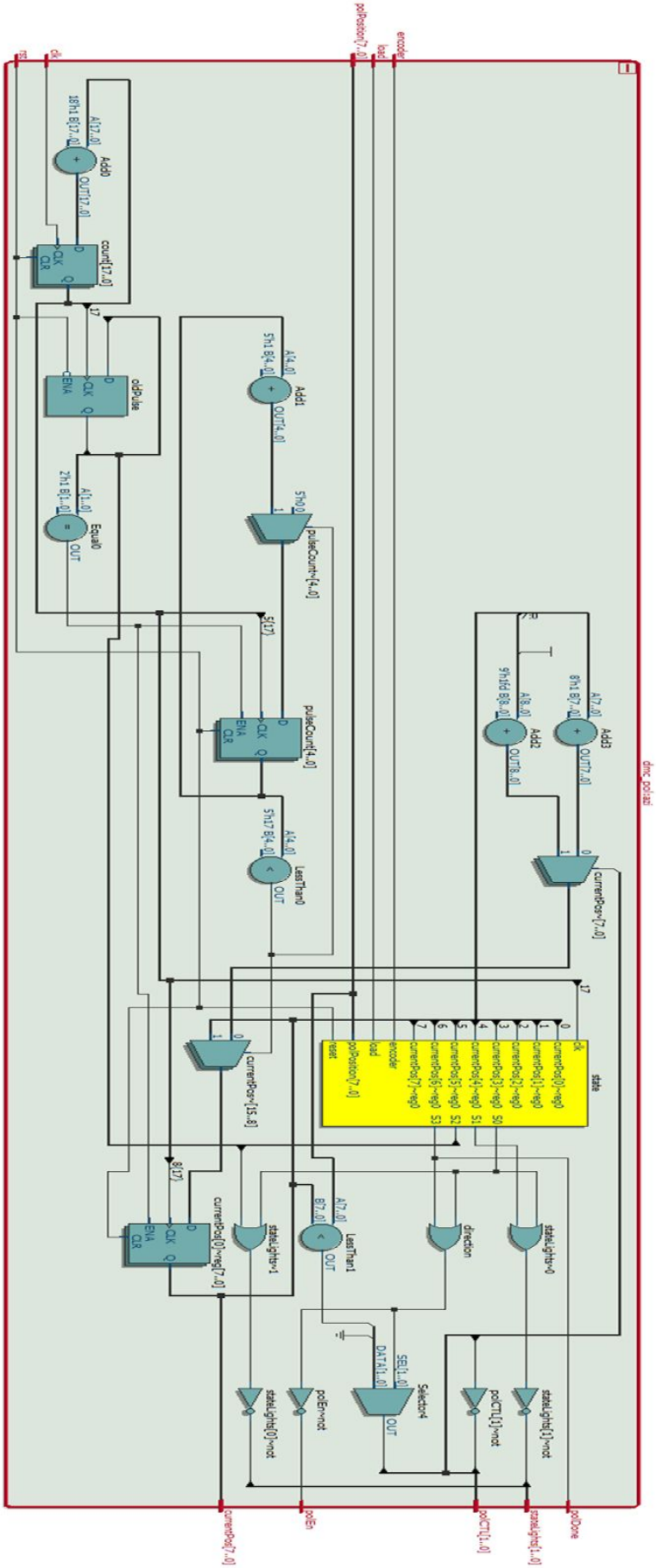


D. RTL Viewer


dmc_pol:azi
clk
aziEncoder
load
isPolDisplay
emergencyStop
polEncoder
rst
sck
sdi
dmc_spi:spi
dmc_pol:pol
currentPos[7..0]
polCTL[1..0]
polDone
polEn
stateLights[1..0]
polPosition[7..0]
aziPosition[7..0]
forceCount[7..0]
LEDs~[7..0]
realAziEn
realPolEn
relay:compressor
relay:solenoid
solActivate
activate
delay[7..0]
openRelay
8'h5
aziCTL[1..0]
LEDs[7..0]
aziDone
comp_relay
solenoid_relay


dmc_spi:spi
polPosition[7..0]
aziPosition[7..0]
forceCount[7..0]
polPosition[0]~reg[7..0]
aziPosition[0]~reg[7..0]
forceCount[0]~reg[7..0]
sck
sdi
D
Q
CLK
6:0


relay:compressor
Add0
Add1
unitCounter[24..0]
timeCounter~[15..8]
timeCounter~[7..0]
timeCounter[7..0]
isActivated
Equal0
clk
rst
activate
delay[7..0]
openRelay


dmc_polari
rst
clk
encoder
load
poiPosition[7..0]
Add0
count[17..0]
oldPulse
Equal0
Add1
pulseCount%[4..0]
pulseCount[4..0]
LessThan0
Add2
Add3
currentPos%[7..0]
CurrentPos%[15..8]
state
currentPos[0]~reg0[7..0]
stateLights%1
LessThan1
direction
stateLights%0
Selector4
stateLights[0]%not
poiErr%not
poiCTL[1]%not
stateLights[1]%not
currentPos[7..0]
poiErr
poiCTL[1..0]
stateLights[1..0]
poiDone

E. DC Motor State Machine Diagram