Ian P. Nolan
www.linkedin.com/in/ian-nolan1

North Bend, OR. 76706
(907) 942-9045 · ian.nolan35@gmail.com

SAMPLE EXPERIENCE

For the class Engineering Design I, also known as junior design, our class was split into teams and assigned the task of loading and delivering boxes of keychains.

Link to validation testing clip: https://youtu.be/jiaw06JLQcc

I served as the electrical systems engineering lead for team 5. I wrote a reflection at the end of that semester detailing which components I was responsible for, why they were selected, and what I learned from each part of the process of integrating that component into our system. That file is attached below. This is a deep reflection, so please use the table of contents to find the part(s) you are most interested in reading about.

Thank you for your careful consideration.

Design Responsible Engineer Report Version 1

Ian Nolan, Team 5 12/6/2024
Electrical System Lead

Table of Contents

List of components I am responsible for electrically integrating into our design:

Component name:	Quantity:	Function:
Linear Actuator	1	Provides linear force to move the keychain stack
PWM Speed Controllers	2	Moderates the DC motor speed
Servo motors	2	Position and orient empty and full boxes
DC motors	2	Drive conveyor belts
Ultrasonic Sensor	1	Detect empty box
Whisker limit switch	1	Detect new keychain
10k ohm resistor	1	Protect Arduino port
Wire fastening strap	1	Secure excess wire
All functions within our code	11	Simplify main code
M4 8mm Phillips head screws	5	Fasten gusseted corner brackets
M4 20mm flathead screws	17	Fasten vertical components to base

Table 1 – List of components

To fulfill my responsibility of integrating these components into our team's chosen design, I needed to understand the mechanical and material attributes of each component and its surroundings. After designing the entire system together as a team, I delegated the mounting of each motor and sensor to the mechanical engineers on our team. In addition to understanding the physical context of each component, I wrote and reviewed several functions, which I am also responsible for. These functions control the basic operation of the components which I am responsible for, as well as one additional function that controls the stepper motor. These functions were executed on our Arduino Mega 2560.

Wiring diagram from CDR

From this diagram, you can get an idea of the context of each component so that you have a reference for the following sections. This wiring diagram is the same as the one resented at our team 5 critical design review. Revisions on specific subsections have been made since then and they will be noted with specific components in the following sections.

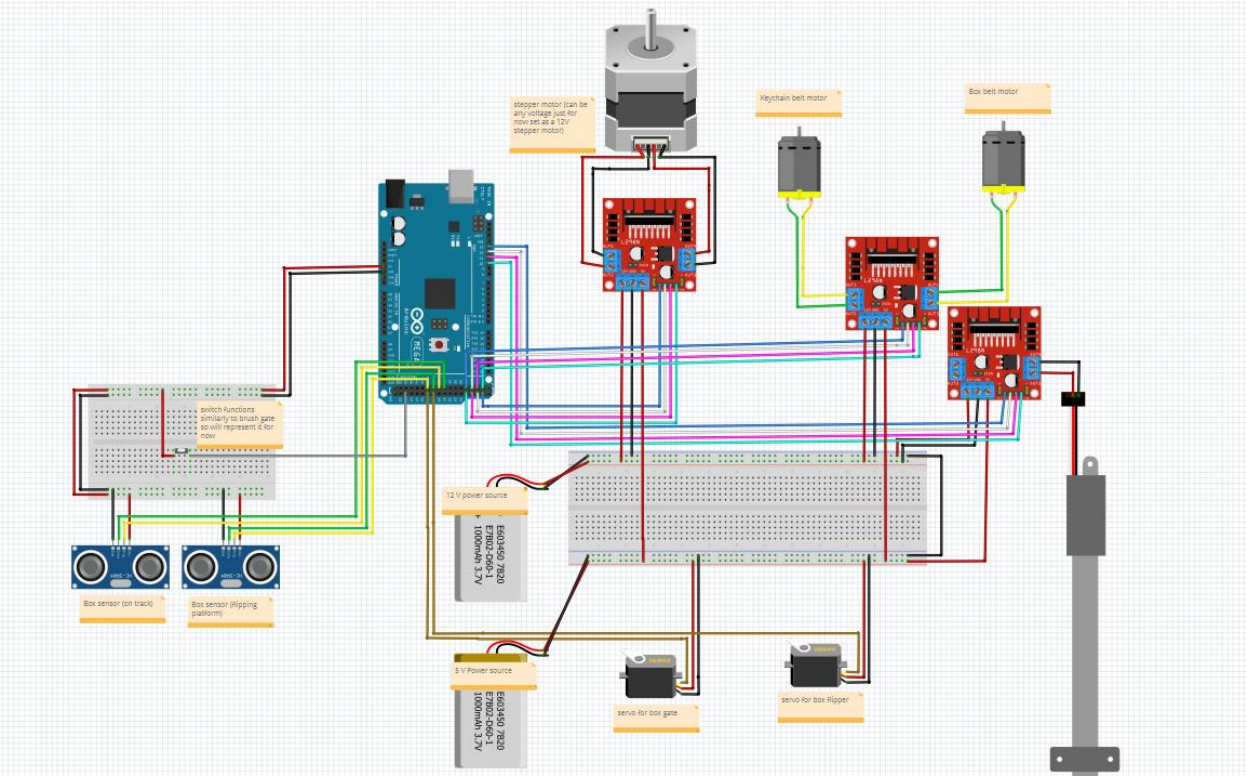

Figure 1 – Broad wiring diagram overview

Component details and cost

- **Linear actuator**
- BES part number: I-1
- Quantity used: 1
- Unit cost: $31.99
- Total cost: $31.99
- Updates to BOM, drawings, or diagrams: None necessary

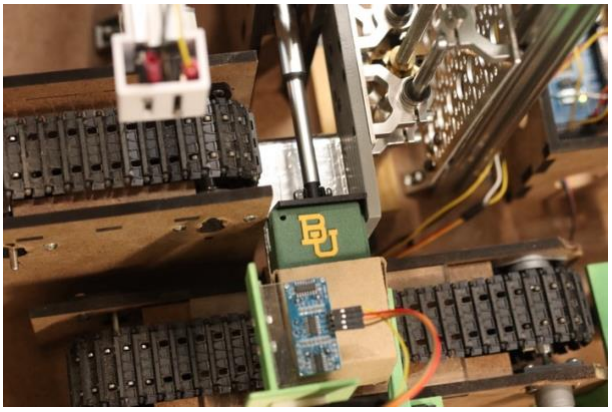

Figure 2 – Linear actuator photograph

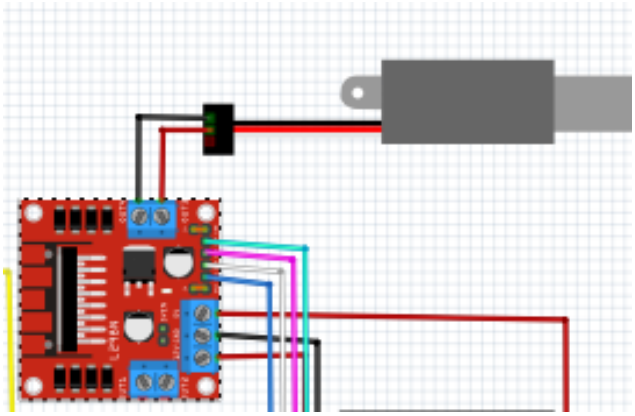

Figure 3 – Linear actuator driven by L298N diagram

- **PWM speed controllers**
- BES part number: I-2
- Quantity used: 3
- Unit cost: $2.50
- Total cost: $10.00
- Updates to BOM, drawings, or diagrams: Added PWM speed controllers (green) to DRE wiring diagram

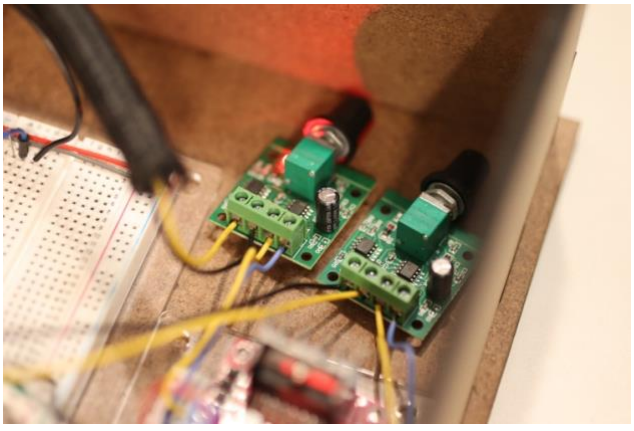

Figure 4 – PWM speed controllers photograph

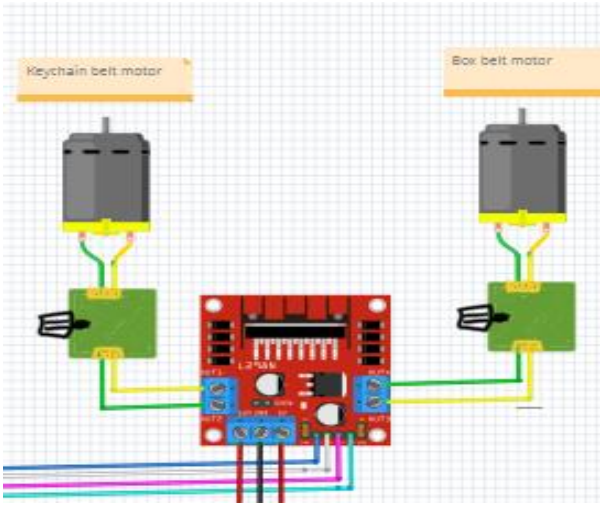

Figure 5 – PWM speed controllers (green) diagram

- **Servo motors**
- BES part number: I-4
- Quantity used: 2
- Unit cost: $5.00
- Total cost: $10.00
- Updates to BOM, drawings, or diagrams: None necessary

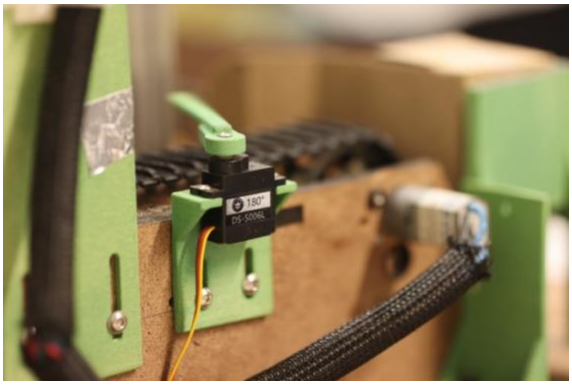

Figure 6 – Gate servo motor photograph

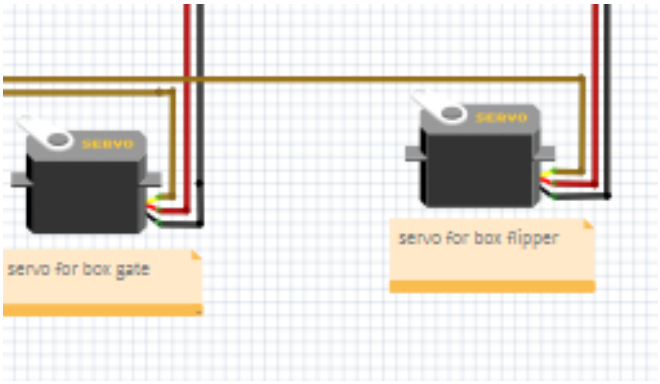

Figure 7 – Servo motors (tan line for signal) diagram

- **DC motors**
- BES part number: I-5
- Quantity used: 2
- Unit cost: $12.99
- Total cost: $24.98
- Updates to BOM, drawings, or diagrams: Added PWM speed controllers (green) to DRE wiring diagram

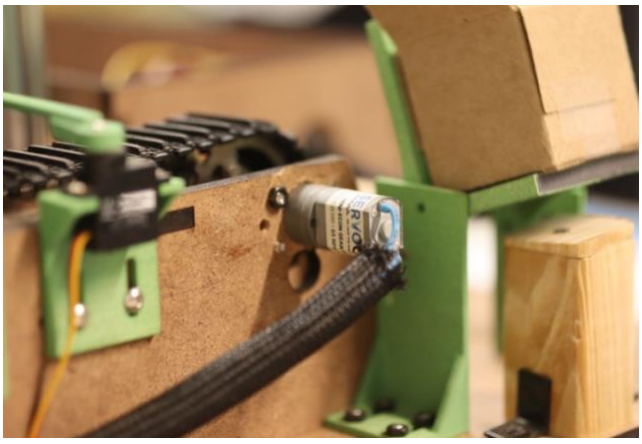

Figure 8 – Box belt DC motor photograph

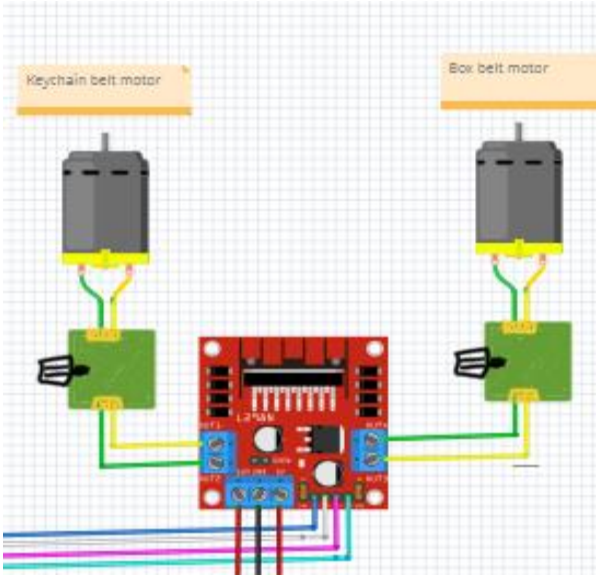

Figure 9 – DC motors diagram

- **Ultrasonic sensor**
- BES part number: I-3
- Quantity used: 1
- Unit cost: $0.25
- Total cost: $0.50
- Updates to BOM, drawings, or diagrams: None necessary

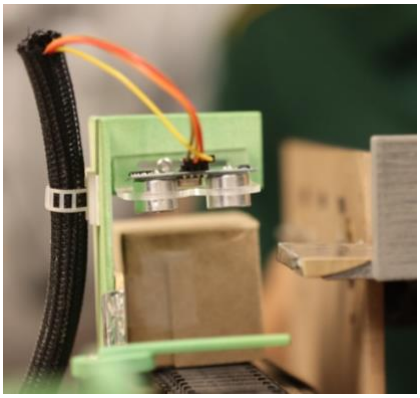

Figure 10 – Ultrasonic sensor photograph

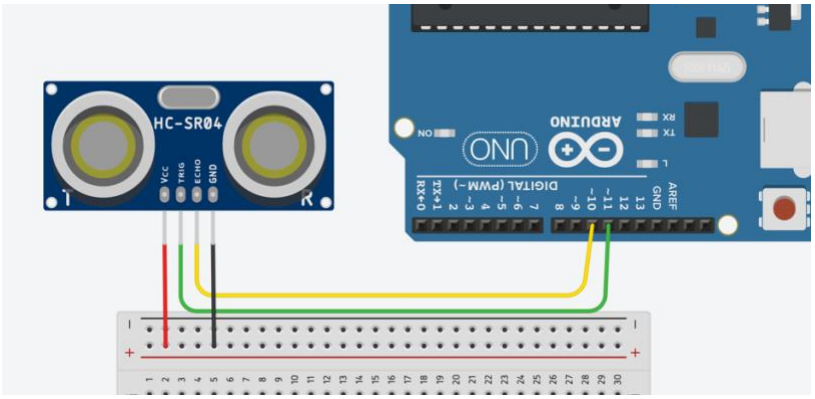

Figure 11 – Ultrasonic sensor diagram

- **Whisker limit switch**
- BES part number: I-6
- Quantity used: 1
- Unit cost: $17.01
- Total cost: $17.01
- Updates to BOM, drawings, or diagrams: Revised wiring diagram to clarify

Figure 12 – Whisker limit switch photograph

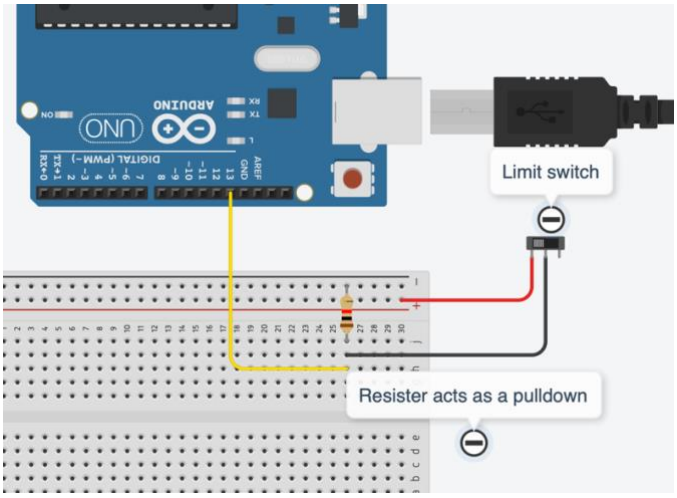

Figure 13 – Limit switch diagram

- **10k ohm resister**
- BES part number: I-10
- Quantity used: 1
- Unit cost: $0.19
- Total cost: $0.08
- Updates to BOM, drawings, or diagrams: Revised wiring diagram to clarify

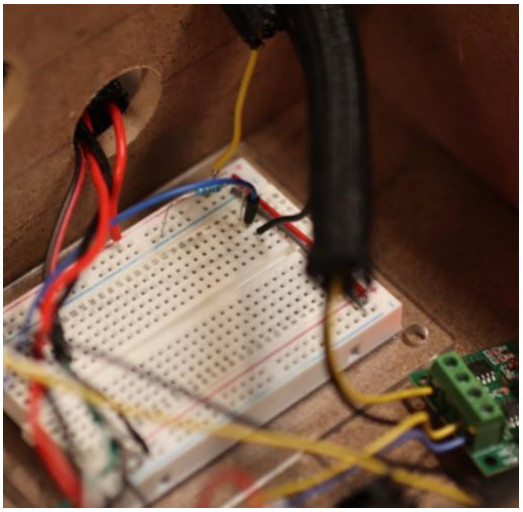

Figure 14 – 10k ohm resistor photograph

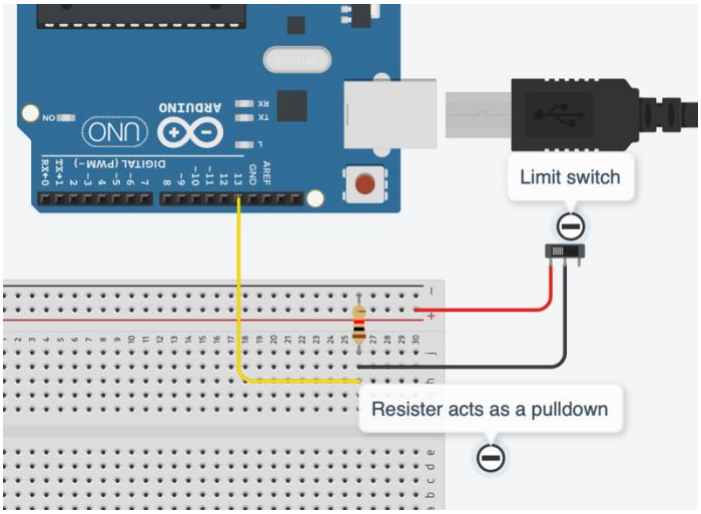

Figure 15 – Pulldown resistor diagram

- **Wire fastening strap**
- BES part number: I-7
- Quantity used: 1
- Unit cost: $0.00
- Total cost: $0.18
- Updates to BOM, drawings, or diagrams: None necessary

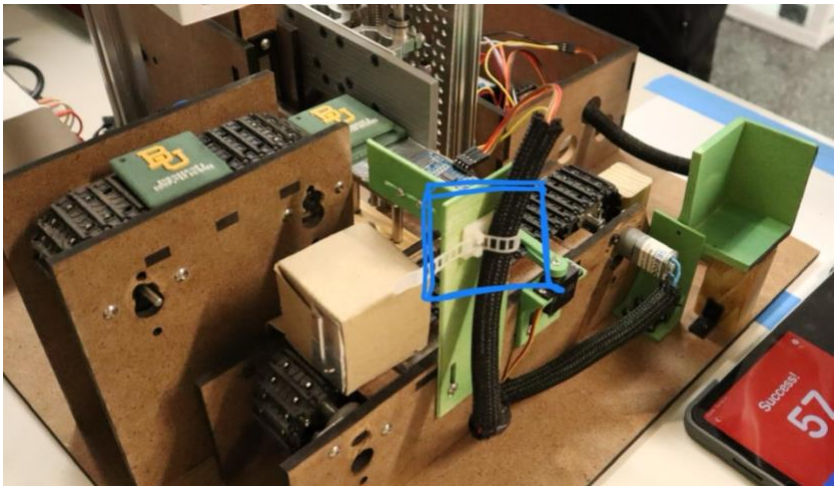

Figure 16 – Wire fastening clip (blue) photograph

- **M4 8mm Philips screws**
- BES part number: I-8
- Quantity used: 5
- Unit cost: $0.06
- Total cost: $0.30
- Updates to BOM, drawings, or diagrams: None necessary

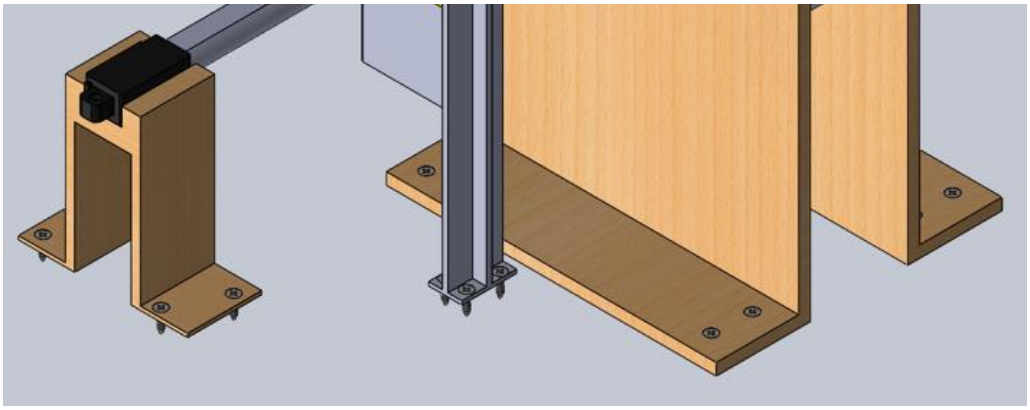

Figure 17 – M4 8mm screws CAD (fastening base of limit switch to gusseted corner bracket)

- **M4 20mm flathead screws**
- BES part number: I-9
- Quantity used: 17
- Unit cost: $0.12
- Total cost: $2.04
- Updates to BOM, drawings, or diagrams: None necessary

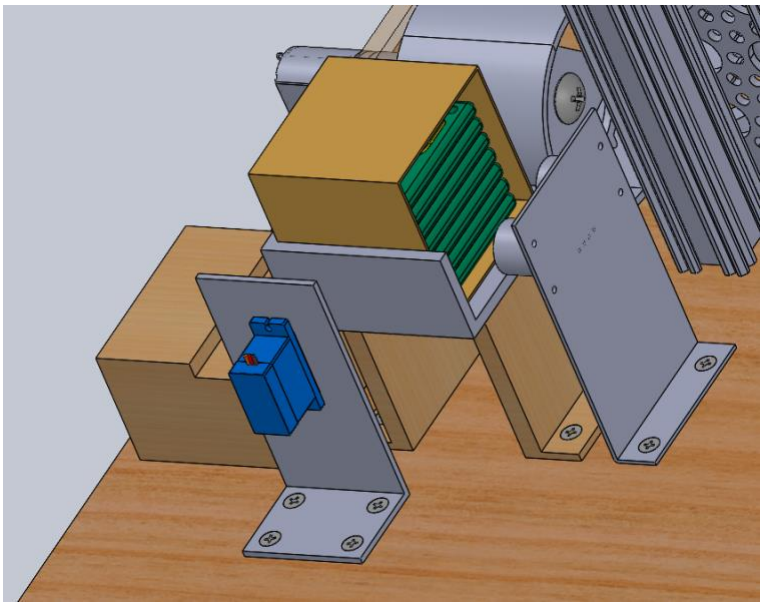

Figure 18 – M4 20mm screws CAD (fastening vertical structures to base board)

Component selection, justification, and reflection:

Linear actuator

Our team decided to use a linear actuator in our final design to accomplish the task of moving a stack of seven keychains a box. Out team arrived at this component choice after much deliberation and revision. The very first idea that was proposed to accomplish the task of moving the stack of keychains was a linear actuator, however, for several days during the development process our team reviewed options such as a scotch yoke which could also provide manageable linear motion. The two main reasons we passed over the scotch yoke were because of the unnecessary complexity of sourcing the correct diameter and the complexity of mounting and aligning so many components. One difficult metric that came into play when attempting to source a wheel for the scotch yoke was the fact that we needed a linear travel distance of precisely four inches. In the end, the linear actuator was selected because it proved to be capable and convenient for sourcing, mounting, and programming to accomplish our task in ways that no other option could surpass.

The linear actuator performed very well during out hour-long system verification test. It met or surpassed all of our expectations and fulfilled its role of moving keychain stacks in collaboration with the rest of our design and in fulfillment of the design attributes we had outlined for it. Although we did face unforeseen issues with the head of the actuator getting caught on our keychain loading platform during the retracting motion, the linear actuator was, in the final analysis, very simple and convenient to not only program but also to physically align with how we needed the keychains to be moved. From this, I have learned that a simple motion component can save lots of trouble both electrically and mechanically when it is compared with a mechanically complex component.

PWM speed controllers

Two PWM speed controllers were used to modulate the speed of the conveyor belts. The PWM speed controllers were used in order to bypass the complication of programming PWM on our Arduino which was communicating with our conveyor belt DC motors through motor controllers. The PWM speed controllers were a perfect fit for our application that required no tradeoffs to be made and we are grateful that Professor Weaver suggested considering them.

Before components arrived from Amazon, a third PWM speed controller was intended to be part of our design. Once the PWM speed controllers arrived, however, we found out that they were not capable of driving a motor in either direction. This became a problem when we attempted to connect our linear actuator to a PWM speed controller, as our design had specified. The PWM speed controller would slow the extension of the linear actuator, as intended, but would not allow any current to pass through in order to retract the linear actuator. For this reason, we removed the PWM speed controller from the linear actuator in our final design.

The PWM speed controllers performed flawlessly during our verification test, accomplishing their design criteria with no issues. Many team members remarked at how convenient it was to be able to moderate the motor speed while doing preliminary testing. The choice to use these PWM speed controllers surpassed our expectations and reinforced the lesson that simple, distinct components are often preferable to complex, integrated ones.

Servo motors

There were two servo motors incorporated into our final design. Servo motors were chosen for these applications because, unlike DC or stepper motors, the exact degree of rotation can be precisely defined through programming. No tradeoffs were necessary because, for these two applications, only a limited range of motion was required. This made using servo motors the obvious choice for these design functions.

The first servo motor opened and closed a gate which positioned the empty box consistently for the keychains to be loaded. This first servo was a later addition to our design, after we decided to trim down the number of sensors we wanted to use. Our initial design did not include any physical barrier to the empty box moving past its target location. Through team brainstorming and many revisions, we agreed that a physical gate would be a prudent replacement for the beam break sensor and complicated programming that we were intending.

The second servo motor we used was intended from the very beginning. This servo motor performed the function of orienting the fully packed box to be upright within the delivery zone.

DC motors

Two DC motors were used in our design. Each motor powered a conveyor belt. Both motors performed exactly as we expected them to. They also integrated perfectly with both our conveyor belt sprockets and the motor drivers we had selected to power them.

DC motors were chosen for the task of powering the conveyor belts because of how simple the required motion was. Not only is the motion simple, but it’s also imprecise. One big advantage the DC motor has compared to a stepper motor during a task like this is the DC motor did not reach nearly as high of temperatures as a stepper motor would have. This is something we noticed during our verification test. The stepper motor that we used became too hot to touch while our DC motors were running just fine for the same amount of time.

DC motors were chosen for their simplicity and reliability. We specifically chose 66rpm motors because we calculated the necessary rotational speed of the conveyor belt sprockets necessary to cause a keychain resting on the conveyor belt to move at the desired rate of 7m/s. These calculations helped immensely, narrowing down our component search to a few high-quality options instead of hundreds of miscellaneous

options with unknown rotational speeds. They perfectly integrated into our design and accomplished their purpose for our system, moving belts that transport material.

Ultrasonic sensor

One ultrasonic sensor was used in the final assembly of our design. The decision to use an ultrasonic sensor was motivated by multiple factors. Primarily, we chose to use an ultrasonic sensor because we were familiar with how to use it before we even started designing the system. In hindsight, I wish more research had been done on other means of detecting empty boxes because the ultrasonic sensor proved to be very unreliable, which made integrating it into our system quite difficult.

To overcome the challenge of the unreliability of the ultrasonic sensor, two things were done. Firstly, we removed the necessity for a second ultrasonic sensor that was planned to detect the presence of full boxes that were ready to be delivered. This ultrasonic sensor was replaced with a section of code run by our Arduino that systematically timed the delivery of the finished package. The second way we overcame the ultrasonic sensor's unreliability was by using the ultrasonic sensor's unreliability. After revising our program, way that a box is detected by our Arduino is by receiving a consistently absurd distance measurement from the ultrasonic sensor when a box is present.

From the surprising difficulty of integrating this component into our system, I learned three things. First, ensuring good quality research is done at the beginning can reap bountiful rewards later on. Second, consistent components are preferable to accurate components. I say this because a consistently inaccurate reading from an ultrasonic sensor can still be used to accomplish that component's purpose within the system. Third, there are there is an abundance of programming and integrating resources on forums and websites throughout the internet. I will certainly be using these newfound resources in the future.

Whisker limit switch

The decision to implement a limit switch into our design was made after lots of deliberation. Several ideas for how to accomplish the task of detecting a new keychain were offered up for consideration. The initial plan was to use a beam break sensor. That option was discounted because our choice of conveyor belt tracks allowed for too much error in keychain positioning. Another suggestion was to use an ultrasonic sensor measuring from above. This option was discounted because of the known unreliability of the ultrasonic sensors and the thickness of the keychain being too small to read consistently.

During one of our design reviews, Truett suggested that we consider using a physical switch to detect the keychains. This idea was immediately adopted by our team because it had no considerable tradeoffs and had promise to be the most reliable option we had considered up to that point. The positioning of the switch was reconsidered

several times. In the final design, the switch is positioned above the keychain conveyor belt with its tip dragging along the belt. Integrating the switch in this orientation worked flawlessly. It cooperated very well with the rest of our system and reliably communicated when a new keychain was present.

10k ohm resistor

This component, although small, was instrumental to the success of the limit switch that detected our keychains. The purpose of this resistor was to protect the digital read pin of the Arduino when 5 volts were directed into it. This resistor acted as a pull-down resistor in the circuit. A 10k ohm resistor was chosen because it is a well-known resistance, commonly used for internal pull-down resistors in other microprocessors. This component performed its task of protecting our Arduino flawlessly by redirecting a significant portion of the current sent from the limit switch into ground.

No computations and very little circuit analysis were necessary for the choice of a 10k ohm resistor to be made. We relied on the body of knowledge already established on the internet and provided by other microchip manufacturers to make the decision to use this resistor. I learned that effective engineering can be done without having to derive each aspect of the circuit or system from the ground up. Using known circuit designs can save lots of time and effort while still fulfilling the circuit's function flawlessly.

Wire fastening strap

One strap was used to constrain wires that were running wildly from the ultrasonic sensor. This component was chosen because it is easy to attach, since it comes with M3 adhesive preset on the back. The only trade-off made in choosing this component, as opposed to regular zip ties or another fastening method, is that this piece appears unattractive, in my estimation. Despite this flaw, the fastening strap performs its function flawlessly and was a good addition to our final design.

With that being said, I do think our system could do without this single strap. It came in a pack of 50, which is far more than we needed. I do think the use of this single strap did not justify purchasing a pack of cable fasteners. Unfortunately, I must conclude that this purchase was a waste of resources. I will be sure to have a plan for each item I request to purchase in the future.

M4 8mm Philips head screws

These 8mm screws were used in a unique location, attaching the Go Rail gusseted corner brackets to the MDF base board. These screws were particularly difficult to source since the shape and size of the head needed to be small enough to allow space for an M4 nut to share the corner bracket from the base-side. After searching through several varieties of M4 screws, only these ones had a head that is low enough profile to

fit in the Go Rail brackets. After lots of searching, these fasteners worked flawlessly and were able to fulfill their role of keeping our Go Rail standing vertically for the entire verification test.

M4 20mm flathead screws

These screws were selected to screw upward through our MDF base, making contact with a nut on the top side that would hold all of our aluminum extrusion, Go rail, and sensor and motor mounts in place. These screws were perfect for accomplishing this function because their head is flat, allowing them to lie flush on a table after counter-sinking with a size 12 bit. These fasteners met our design intentions by holding each structure erect for the entire verification test.

Code functions

Programmed on Arduino Mega 2560 are several functions that enable our main code to run seamlessly. Many of these functions are very small and simple. Some of them are more complex. There are two functions that I would like to highlight during this review: readUltrasonic and checkKeychain.

readUltrasonic is a function that was primarily sourced from arduinogetstarted.com. The reason I think this function is important to highlight is because it reenforces a lesson that has been taught several times so far this semester; there are many resources and people available online and in person to assist me in my education and application of electrical and computer engineering. Researching and incorporating this function was significant to me because I have never had the freedom to use online resources in this way before.

checkKeychain is a function that reads the state of the limit switch pin and outputs whether there is a keychain detected or not. The reason this function is worth highlighting is because I am proud of how well it worked during our system verification test. It performed flawlessly. I think that some of the success of this function is due to its crude debouncer that double-checks whether or not there is truly a keychain present after a short delay.

Concluding reflection

The process of integrating the motors into the overall design went well. The DC motors were particularly straightforward to program and attach to the conveyor belts. Their associated PWM controllers made modulating their speeds a breeze. The servo motors took a little bit of adjusting and hand-holding to get to the perfect angles we needed them to be in a smooth and controlled manner, but they worked out very nicely in the end. The stepper motor, which I am not responsible for, had more hiccups than we were wanting, but it was good to learn as we troubleshooted and asked for council about our problems. The linear actuator was excellent and performed without fault.

Integrating sensors into our design was somewhat tedious. The ultrasonic sensor was far too unreliable at sensing when an empty box had arrived, even after doing as much programming for reliability as we could through the software. The whisker limit switch, on the other hand, performed well and was simple to use and program into our design.

Overall, the simpler components integrated better into our system than complex ones. I think this is due to our team having a better understand of exactly what to expect out of the simple components. On, off; true, false. These are Inputs and outputs that are easy to understand and easy to program with.

For reflections and considerations for individual components, please review the component selection, justification, and reflection section.

With all of that said, if I were to do a similar project again, I would try to use as many binary inputs and outputs as possible and use external tools such as the PWM speed controllers to modulate any gray-areas in between on and off. When it comes to sensors, I would be more discriminant in my research, checking to see if there is a sensor that is better tailored to my exact needs. I would also be less closed to using mechanical sensors such as buttons and limit switches. The limit switch we used was exactly what we needed, and I’m sure using more of them could really help the reliability of a system like ours.

As the electrical system engineering lead, I could not have been more pleased with my team. The way that we effectively collaborated on every aspect of this project was gratifying to be a part of. I hope to work with them again soon.