Speech To Text Activated Typewriter

Michael Xiao
Cornell University
Ithaca, NY
mfx2@cornell.edu

Yanir Nulman
Cornell University
Ithaca, NY
ywn2@cornell.edu

INTRODUCTION

The history of mechanical key-pressed input systems to compose lingual messages stems back centuries, with early innovations beginning in the 16th century and accelerating during the 19th century. It wasn't until the 1870s that the keyboards we know and love today grew popular, which was due, in part, to the advent and introduction of the typewriter [6]. For decades, the standard QWERTY keyboard, introduced by Christopher Latham Sholes, became the world standard due to its lower statistical probability of causing typewriters' arms to jam. The QWERTY layout became so commonplace that it remained the standard for the computer keyboard during the rise of the digital age.

Fortunately, computers no longer have mechanical arms that can get stuck as typewriters did. Therefore the QWERTY layout is no longer a useful structure, especially in an era that values efficient work speed. To address this baggage, researchers and companies such as Google and Microsoft have introduced Speech to Text (STT) technologies that monumentally increase the speed of language input on digital devices due to the fluidity and low-effort of speech.

Currently, STT is most often implemented without driving any mechanical movements or traditional computer keyboard inputs. A user simply speaks and the text appears on a display or is repeated to them via a voice assistant. The joy of hearing and seeing typing is becoming sparse in the age of "hands free" computing.

To address this epidemic, we are looking to introduce a solenoid driven, mechatronic typewriter which uses hands-free voice dictation and a cloud based STT technology as the input. Beyond serving as a novel integration of new and old technologies, the piece serves as an artistic statement of technology's past, present, and future.

RELATED WORK

Speech To Text (STT)

STT is widely accessible for both local and cloud-based processing. The technology has been in development since the 1970s but has rapidly improved in speed and accuracy in the past decade. Specifically companies such as Microsoft, Google, IBM, and Amazon, have achieved an accuracy as high as 97%. The technology has become adopted en masse through applications such as voice powered assistants, hands free texting in vehicles, and even reached areas such as military applications and accessibility use cases.

Keyboard Typer

Much of the related work in mechanical keyboard typers has been done by hobbyists and implemented in various forms. Travis Scholl on youtube published a video of a "Keyboard Robot" he built with a lego robotics kit[1]. The design includes a two axis system that moves to the location of the desired key to press and uses a gear system to punch into the key.

TY The Typing Robot is composed of an arm that uses a camera and light computer vision to obtain a relative position of the arm as it moves around the keyboard[2]. The computer vision approach allows for a dynamic swapping of different keyboards.

Callie A. of The Dalton School built a typing system similar to TY The Typing Robot in that it uses an arm to move around the keyboard's space, but does not use computer vision as a guide. Callie does, however, use Google's Speech To Text services to dictate to the system what to type[3].

ROBOETRY II is a two arm keyboard typing robot-poet system which selects words from a dictionary to write a poem and then recite the poem[4].

Autonomous Typewriters

Hobbyists have also worked on automatic typewriters, which require a bit more complexity because of their multi-level keypad system.

Brian Benchoff uses an array of arms on servos to press into the typewriter and another arm that is fixed to a sliding rail to slide the page back into starting position. The system is controlled by voice dictation on an arduino microcontroller[5].

Mike Szczys built a system that uses a string system underneath the typewriter to pull down the keys to press. Because the actuation system is underneath the typewriter, the typewriter appears to be typing on its own. The system was designed to be an art installation[6]. While this design was not compact, it became a point of inspiration for our work.

DESIGN GOALS

As mentioned above, the main goal of the system aims to fuse an "ancient" typewriter with modern STT technology. Beyond executing this bare minimum, we also hope that this integration is seamless, almost as if it were invisible. Hiding the mechatronic actuation of the keys underneath or behind the typewriter is a major design goal as it not only preserves the ability to manually type on the typewriter, but it contributes greatly to the overall "magic" of the self-typing typewriter. The hidden implementation is meant to mirror the "ghost piano" that has caught the public's attention from years before and presents it in a new and interesting medium.

Overall, such a system uses the simplicity and ease of use of voice as the input, yet preserves the joy of mechanical typing. Moreover, as mentioned before, we're looking at the project as an artistic take on how we are witnessing, in real time, the antiquation of the mechanical keyboard as an input. Smartphones are being driven by voice, swipe to type keyboards, and other input explorations. Smart TVs reject physical keyboards entirely and move away from the QWERTY standard. This project is the intersection of where digital input is going and where it used to be.

HIGH LEVEL DESIGN

To actually execute upon the rather broad goals set out in the design goals section, it was important that the project was split up into individual electromechanical segments for individual exploration and eventual reintegration. This technique helps us diverge and assess the options to best execute very specific motions and also allows for much easier division of labor.

As a general overview, the typewriter sits upon a small pedestal (~4 inches tall) in which the mechanisms of the self-typing system are housed. This is a "black box" effect in which the true mechanisms are hidden from the vision of the user/observer. Within this box are 28 different tendons that are attached to the arms of the keys we plan to use (26 letters, period, space bar). Each of these tendons act as pulleys that are routed through a bar that redirects the tendons parallel to the plane of ground and backwards away from the keys. Our original design intended to connect the tendons to a row of spring-tensioned pistons which, when pulled by an actuator, will pull the tendon to strike a key. Though this worked, it was very slow as the stepper motor would have to move a large distance, often using several rotations, in order to reach from one key location to the other. Furthermore, if we wanted to speed this up, we would need to use several actuators that would continue to increase the complexity and form factor of the piece.

Unfortunately, initial prototypes of this system proved to be too slow for the intended "magic" of the art, and we changed to a design in which a solenoid sat on a servo which pivots to the right lever on an arch that corresponds to a key press. This arch has arms that have tendons connected to it. The solenoid then punches the lever arm which in turn yanks on the tendon to pull the key arm down over a fulcrum and strike a key.

In addition to figuring out the key presses, we also aimed to automated the carriage return after the typewriter reaches the end of a line. This involves both sensing when this needs to be done and actually pulling the carriage over in a sleek and hidden way.

While we were building the final integrated system, we learned that the paper roller knob on the left side is threaded and unscrews itself when the paper is rolled due to typing progression. After the first unscrew, it rendered the reset lever useless, and we had to resort to manually resetting the carriage return by hand. The software design counts how many characters are typed and stops when at the end of a line and waits for the manual reset to the beginning of a new line to resume typing. Because of the poor construction of the typewriter, we decided to forego the automation of this piece to prevent the self deconstruction of the device and to maintain the sleek aesthetic of the current model.

The last piece of architecture exploration that needs to be completed is the crucial speech-to-text component of the project. We began the project by exploring using Google Cloud services to compute speech recognition online. This would be done directly on the ESP32 connected to WiFi/ However, during the budgeting of our project, we decided to not spend money on Google services and decided to use Apple's free services on the iPhone and communicate with the ESP32 via bluetooth. This also eliminated the need for a microphone, which would have needed tuning as well.

Figure 1 shows a cardboard prototype showing the tendon architecture and visualizing the footprint of the device.

What operation would look like can be seen in video 1. As seen, the mechanism sits below the typewriter such that the keys can still be used manually.

Figure 1: Typewriter on cardboard prototype

DESIGN AND CONSTRUCTION

In this section, we describe our experiences and findings from the first prototype of the device along with how we addressed these issues in our final model. In this initial prototype, we aimed to construct a linear multiplexing system using a belt to pull a solenoid lever arm to hit the tendon arms. Below, we describe the key functional units from our construction.

Key Actuation

Pressing the keys of the typewriter is probably the most important functionality that the rest of the system relies upon for successful operation. This proved to be a very interesting problem to solve as the constraints required balancing several parameters. First, the actuation had to be fast enough to produce the required impulse to actually strike the letter upon the tape. However, making an actuator too fast sacrificed power needed to actually push against the spring resistance of the key. Furthermore, consistency is also very important to this particular functional unit because it will be constantly actuating throughout the use of the device. Lastly, frequency was also a very important consideration. This differs from speed in that it represents the "reload time" between each keystroke. Having a high typing frequency can allow the device to better follow up with what has been said in the STT microphone.

With these initial design parameters, we brainstormed solutions and found that a solenoid with a lever arm would provide an effective means to accomplish our goals. Our resulting 3D printed mechanism is shown below in figure 2 and is demoed in video 1 in the folder below. The design consists of 3 3D printed pieces: 2 that screw into the solenoid and a lever arm attached to the solenoid shaft. The longer end across from the fulcrum gives more reach to the system and allows for even faster strokes. This system was very promising so we planned ahead and added holes such that the 8mm rod bearings could be secured as well.

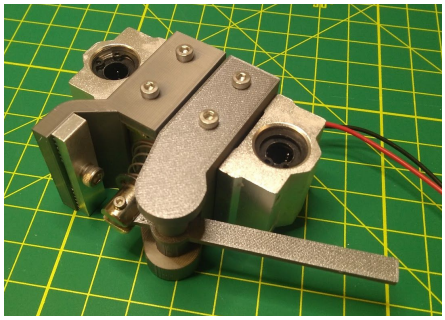

Figure 2: Solenoid with lever arm

As we moved on to testing with the actual typewriter, we soon found that though this solenoid was powerful, it still was not able to generate the necessary impulse for fully striking a key to produce a visible letter. This led us to change hardware to a larger solenoid, operating at 6A and 12V. This solenoid is shown below in figure 3 and was much bigger than our previous design as seen in scale with Michael's hand. This new hardware was more powerful but increased the form factor of our linear rail system greatly, pushing us further from the implementation.

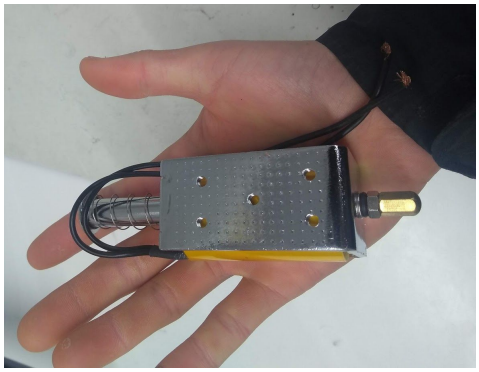

Figure 3: New 72 W solenoid

Multiplexing the Actuator

The main goal of multiplexing the actuator was simply moving the actuator horizontally across a single axis. Drawing inspiration from other plotting devices, we initially decided on using a belt driven system relying on a set of 8mm guide rails to limit motion to just one axes. Luckily, we were able to find some in the parts left over

from last year to test with and bought the belt and pulley pieces.

As our parts came in, we began to design our system consisting of two side plates supporting the three linear rails to house the solenoid movement and the swing arm pistons that connect directly to the cables. One side plate contains mounting brackets for both the stepper motor and the limit switch. The stepper motor drives the system via a pulley attachment. The limit switch is a fancy button that is used to calibrate the system, serving as a zero point when the carriage activates the switch. The other side plate contains a notch for latching in another pulley to tension the belt upon. Between these two plates are three different rods. the arms are separated by small bearings. Figure 3 shows a CAD model of the side plate. Figure 4 shows the assembled unit. Figure 5 shows a more closeup view of the carriage and limit switch. Video 2 shown in the video folder below highlights the system moving and calibrating via the limit switch.

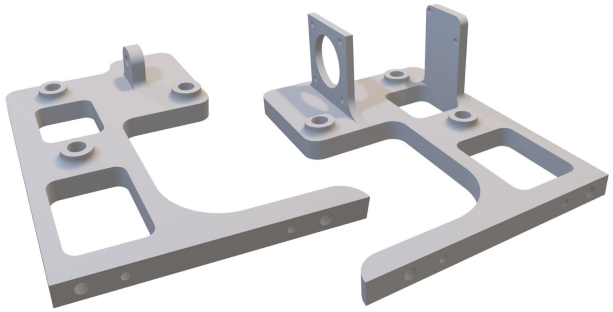

Figure 4: CAD models of side panels

Figure 5: Belt Linear Actuator

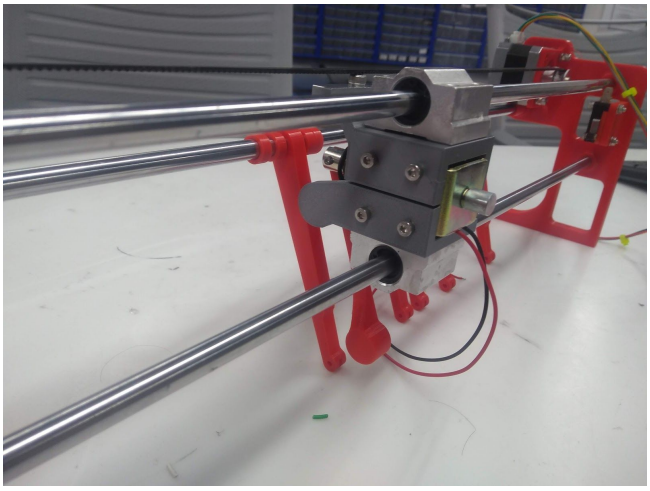

Figure 6: Closeup on limit switch and components

The main shortcoming with this initial system is that it is a little slow, and one of our design goals was to ensure that the typewriter felt magical. This is something that could only be achieved with a faster typing speed than what we could not each with the current system. Therefore, we switched to a radial design which uses a solenoid on a servo and punches levers on an arch. Rather than moving a large distance linearly across the rail, we can simply twist about a single axis to access our 28 tendons. The main drawback of this approach is the complexity of tendon routing and more precise multiplexing necessary to distinguish between individual letters. This addressed several of our issues including substantially faster typing and minimizing the footprint of the mechanism.

As an overview, we used a 3D printed arched arm holder. This model is shown below in figure 7. As seen, there are 28 slots for arms to fit inside. Our first challenge was finding something that could hold all the arms together in a rounded path. We overcame this by creatively utilizing a 0.25 OD flexible tubing segment to route through the holes, creating a rounded axis of rotation for the arms to sit in.

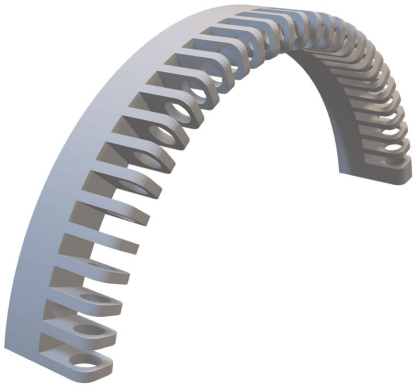

Figure 7: arched arm holder

While routing our tendons, we also found it was very difficult to correctly tension our strings such that they were taught enough to achieve the required distance of movement while being loose enough to allow the solenoid to pick up some momentum before the strike. Furthermore, the close quarters of all the arms made it very difficult to tie the strings. This led us to create a tendon arm that looks like figure 8 shown below. We used a laser cutter to cut this piece so that it was fast and easy to create 28 copies of. The hole allows the string to be routed through the arm and the two notches around the sides allow us to wrap the string around the sides to tension the tendon rather than tying a really tight knot. We found that after 3 wraps and a small piece of tape, the friction was secure enough to hold the string to the arm even with the solenoid actuating.

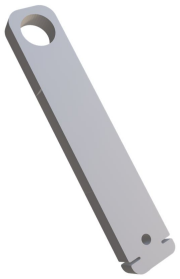

Figure 8: tendon arm

Lastly, in this functional unit, we had to configure the solenoid with the servo so that the servo could rotate the solenoid module. To do this, we first laser cut a circular guide that houses the solenoid that constricts its motion to twisting about one axis. This is shown in figure 9 below with just one arm in the arm holder. As seen, the solenoid sits between two round plates. These plates are constricted by rings on the bottom. The servo mounts to the top of this structure via an attachment point and a set of screws. The servo sits in a attachment piece that allows it to be secured statically to the top surface of the casing. This was a very tricky piece to design as it required a lot of tolerancing and exact measurements within the z axis.

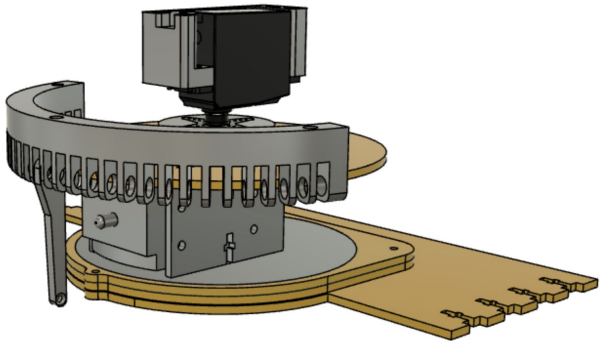

Figure 9: Servo multiplexing system

As we moved forward with this design, we also had to begin to consider the framework of the device and how components would fit together. We had to keep in mind how to structure our pieces in order to make wires accessible and the different layers on the internal part of the device. This will be further discussed in the integration section. Video 3 in the video folder shows some early testing with this setup.

Carriage Return

Returning the carriage is another very important part of the system as it is responsible for resetting a new line to be typed on. We originally had planned to use a belt system as indicated above to achieve this long linear motion, but as mentioned earlier, the platen knob (paper rolling knob) which is attached to the carriage return lever broke during integration and testing and removed the capability to automatically return the carriage. This made automating this with actuators a rather difficult task at the end.

Because of this, we instead resorted to pausing the typing and waiting for a manual return of the carriage and paper scrolling. In terms of sensing when the carriage should return, we have a simple counter variable implemented that counts the number of keys that have been pressed since the last return. This works because each press on the typewriter is equally spaced, meaning regardless of what letter is pressed, the horizontal spacing will always be the same. Each time the actuator is used (letter, space bar, or period), we can simply increment the variable and compare it to a limit to sense when the carriage return needs to happen. In our testing with 8.5" width paper, we found this limit to be 70 characters. This pause in typing was implemented on the software side of the ESP32 microcontroller we used for the system. In the future, we could also add a microphone to sense the bell of the typewriter such that it is adaptable to any width of paper. We would have to be careful to reject noise into the microphone however.

Speech to Text and iOS App

We decided to take advantage of the ESP32's bluetooth capabilities and built an iOS application that connects to and communicates with the device. Building an iOS app was also advantageous because we could use Apple's built-in Speech to Text services for free[7].

In order to enable speech to text services from Apple, permission to use the microphone and speech recognition by the device owner is necessary. Once granted, the microphone stream is sent on a separate thread to a speech recognition request which returns and updates the interpreted speech as a string with other metadata encoded such as confidence level and timestamps. Originally, we had planned to use the timestamps to be able to manage keyboard editing, but decided to disable speech recognition when the user would use the keyboard as an input.

The app is comprised of two way bluetooth serial communication which was adapted from Arduino's bluetooth libraries for the ESP32 side and helper files adapted from Jindřich Doležy's github repository[8]. The app sends character bytes to the device and the device sends a progress integer back to the app.

Additionally, the app displays the connectivity with the ESP32 via two modes: dark and light. Dark mode (gray text on a black background) is indicative of no connection to the device, and light mode (black text on a white background) is indicative of a connection.

The app and device follow a communication protocol in which the app sends one character from the current spoken (or typed) message and the device will return how many characters have been typed by the automatic typewriter system. Once the iOS app has received the progress made by the typewriter, it will send another character if possible. Only characters which are able to be typed out by the typewriter are sent. Therefore all characters are lowercase and the only punctuation sent is periods (question marks and exclamation points are converted to periods).

The iOS app displays the progress made by the typewriter by using two colors for the text. Characters that have yet to be typed are light gray and those that have already been typed are black. A custom UITextView component was made to dynamically manage the view heuristics based on the progress of what was typed. This was done in order to abstract out the text management from the main view controller and into the UI component itself. The visuals of this feature and other heuristics are highlighted in Figure 10.

It was of great importance that the app contain little to no functionality other than to control the typewriter. We wanted the interaction to feel as though it was with the typewriter, rather than with the phone. Therefore, there are no buttons or additional functionality other than the text and speech input. The simple black and white UI and Courier New font work to enhance the effect of feeling like the users are interacting with the typewriter.

The app also has a shake to reset function in which the text clears, the speech to text resets, and the app sends a reset key to the device to restart the ESP32 all upon shaking of the iPhone.

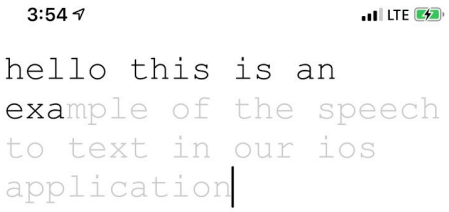

Figure 10: App Screenshot

Integration

After getting all these functional units working, we started integration. The integration consisted of two main portions: hardware and electrical system.

Hardware integration revolved around the design of our frame. The internal skeleton of the frame can be seen in figure 11. The frame consists of 4 separate compartments. The compartment in the front (pictured top right of picture) allows strings coming down from the keys to be routed. The middle compartment houses the servo and solenoid setup described in the multiplexing section of the report. This compartment has two layers. The bottom layer has a constricting base that allows the solenoid to spin on one axis. The top layer puts the arm holded arch at the right height such that the tip of the arms are just a few millimeters above the ground. The left and right compartments in the back are for storage of electronics and can be accessed through laser cut windows that allows wire to reach all parts of the device.

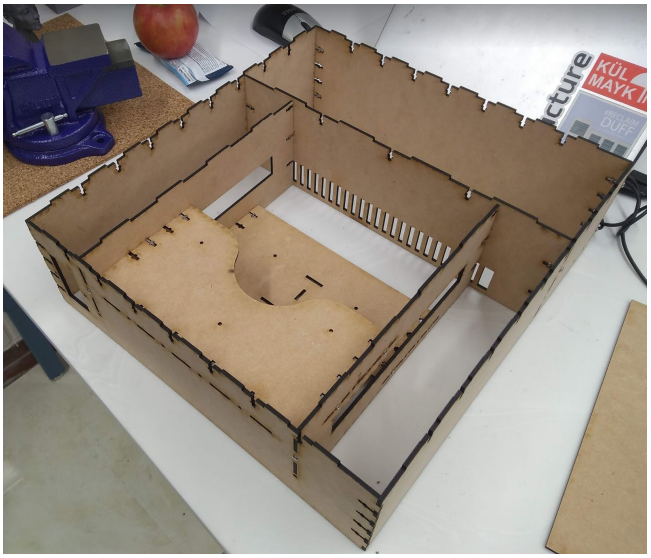

Figure 11: Typewriter skeleton

Figure 12 shows how tendons are routed. Coming from the top, the tendons wrap around a rod suspended in the frame that twists the vertical tendons into horizontal motion. These tendons are routed through the grill, which keeps the strings orthogonal to their original motion downwards from

the individual key that each tendon is attached to. Lastly, the tendons are routed to respective arms. The routing from the bottom of the device can be seen in figure 13. This routing took a substantial amount of time as we had to properly tension and attach 28 different strands. This marked the hardest part of our project - making a solution that worked for 3 units work for 28 units. Everything had to be done 28 times making the project repetitive and monotonous. This made it especially important to spend time contemplating the most efficient and effective solution before executing it.

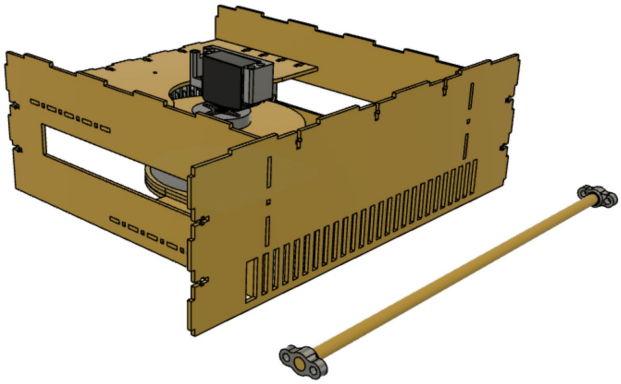

Figure 12: Front view of support rod and grill

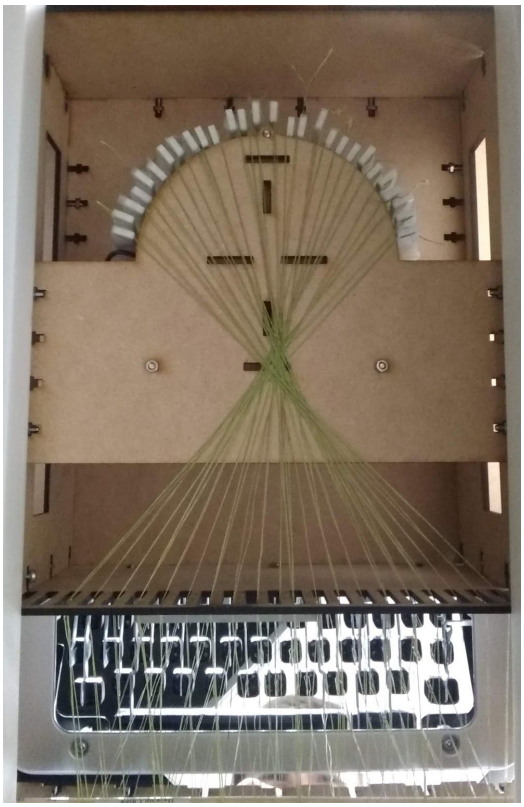

Figure 13: Tendon routing from below the device

This configuration of laser cut walls also made our system very modular. We connected all pieces using T-slots and M3 hardware such that it is easy to remove the outer casing for fine tuning. Furthermore, the orthogonal bracing made the frame very sturdy, crucial for supporting the heavy weight of the typewriter without buckling.

Lastly, to add one last piece of functionality to our device, we implemented a set of manual controls. This was done by adding a toggle switch that would switch the mode of the device. In STT mode, the typewriter would function as designed, allowing the user to speak to the phone app to control its typing. If the switch is pressed, the device can be toggled into semi-manual mode, allowing the user to use the potentiometer and the button on the front of the device to control the angle of the servo and the actuation of the solenoid, allowing the user to try to hit keys on their own. This added one more layer of intractability in our system. To interface this with the frame, we used a drill press to drill mounting holes in the front of the device. The results are seen in figure 14.

Figure 14: button controls at the front of the device

After the frame and casing was manufactured, it was time to fine tune all the electrical systems to operate together. The electronics include a 12V power supply mounted to a power switch for safety. We also had to integrate the high power solenoid, a strong non-continuous servo, a switch, a button, and a potentiometer. As mentioned, above, we utilized the Feather ESP32 microcontroller due to its internal bluetooth and very fast clock speed. However, this choice of microcontroller gave rise to several issues. First, the relay we had to actuate the solenoid ran on 5V so the logic on the controller would not actuate the hardware. Secondly, the servo we had also runs on 5V. To solve these problems, we proposed three separate solutions. We could have an intermediate Arduino Uno microcontroller to process commands from the ESP32 to actuate these devices, we could use a level shifter, or we could use the UA7805 to step down the 12V down to 5V for use. We ended up choosing the third solution due to our familiarity with the approach and bought a new 3.3V logic relay for the solenoid. This resulting circuit can be seen below in figure 15.

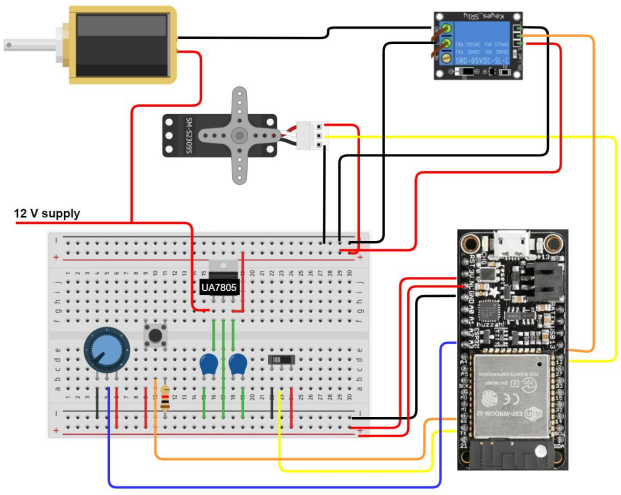

Figure 15: Electrical Schematic

Finally, after seeing that our circuit worked on a breadboard, we wanted to make our electrical system more robust. To do this, we moved the entire circuit onto a solderboard so that we do not have to deal with wires coming loose or shorting across different terminals. This was time consuming but paid off as our electronics are sleek, safe, and reliable. Finally, we used wood screws to mount all of our electronics to a base plate that the frame was also mounted to. This view of the electronics can be seen below in figure 16.

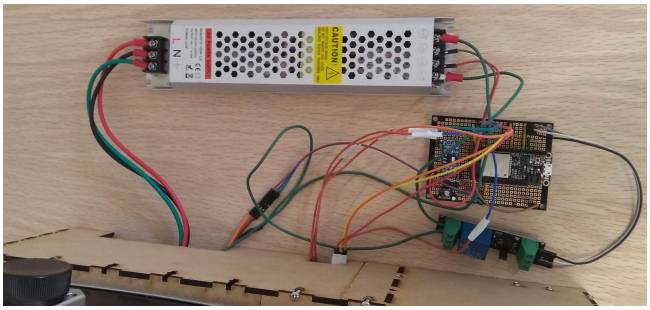

Figure 16: Electrical setup

Figure 17: Completed STT Typewriter

FUTURE WORK

Through our prototyping and final integration we learned of many opportunities to improve the project. First, to improve the typing speed even further, we could include a second servo-solenoid unit to actuate the pulleys. With the addition of such a modification, it would also be useful to reroute some of the pulleys in a manner that groups commonly hit characters together so that the servo would need to rotate less to hit characters that are statistically likely to follow one another.

Another important addition to the system would be a working carriage return that does not require human intervention. A belt and rail system would be one method of implementing the carriage return, but would also require a pulley system to activate the carriage return lever. This addition would add to the "magic" that we intended to create with this project.

It would also be helpful to brainstorm a more efficient way to tension and tie our tendons to the arms. The status quo requires us to set the device across two tables and work from beneath it much like how an auto shop works on cars. This is extremely tiring and significantly slowed our progress as we had to spend so much time wiring the tendons. Figure 18 shows Yanir hard at work performing this task.

Figure 18: Yanir wiring tendons

To ensure project feasibility within the scope of this class, we narrowed the range of "hittable" characters to lowercase letters and the period punctuation mark. Subsequent work on the project should include expanding this set of characters to all possible characters supported by the typewriter.

Enabling automation of the shift functionality of the typewriter further expands the set of characters that could

be typed, but poses a challenge because it requires more force to actuate than any of the other characters.

Lastly, we are not fully satisfied with the reliability of the solenoid hits on the arms. Often times, when we try to hit a letter, it will also hit a neighboring letter, failing to actuate the actual key strike. This solution has several solutions. We could find a more precise servo motor; the current one often will go to slightly different positions even with the same code with the tight granularity in the project. Another solution that would help with this is adding an attachment to the tip of the solenoid edge that allows the tip to catch just one arm while pushing the adjacent ones aside. We could also integrate a bottom layer of arm holders such that the spacing between arms is more consistent as there is some horizontal wiggle room that makes some spacing tighter than others.

Though we were pretty impressed and happy with our results in this project, there are many steps for further improvement and we hope to put more effort into the project next semester as we find more time to work.

PARTS LIST

A detailed parts list for the project can be found here:

https://docs.google.com/spreadsheets/d/1bISCvK-7QLbU-KKgEN4fP5oyISpZfHnrEcDQbU2vrPQ/edit?usp=sharing

VIDEOS AND DOCUMENTATION

Videos referenced in the text, additional videos highlighting the mechanism, code, and other files referenced throughout the text can be found in a Google Drive folder here:

https://drive.google.com/drive/folders/10brgs3JIbIE2VTG_ixcaPvk0HnG9hVxx?usp=sharing

REFERENCES

1. Travis Scholl. 2010. Keyboard Video. Video. (2010). Retrieved September 22, 2019 from https://www.youtube.com/watch?v=qExIeZtt8KA
2. Elecia White. 2018. Ty the Typing Robot. Web. (4 May 2018). Retrieved September 22, 2019 from https://hackaday.io/project/157946-ty-the-typing-robot
3. Callie A. 2016. Typing Robot. Web. (29 July 2016). Retrieved September 22, 2019 from https://bluestampengineering.com/student-projects/typing-robot/
4. Steph Horak. 2013. Roboetry II: A typing robot poet. Video. (31 January 2013). Retrieved 22 September 2019 from https://www.youtube.com/watch?v=u3gQnFP8o1g
5. Mike Szczys. 2013. Automating a Mechanical Typewriter. Web (11 April 2013). Retrieved 22 September 2019 from https://hackaday.com/2013/04/11/automating-a-mechanical-typewriter/
6. Brian Benchoff. 2017. The Voice Recognition Typewriter. Web. (17 February 2017). Retrieved 22 September 2019 from https://hackaday.com/2016/02/17/the-voice-recognition-typewriter/
7. Apple Speech Framework , 2019. Web. Retrieved 9 October 2019 from https://developer.apple.com/documentation/speech
8. dzindra. 2018. BLE-iOS-demo. 2018. Web (28 November 2018). Retrieved 20 October 2019 from https://github.com/dzindra/BLE-iOS-demo