Ricky Jones
jonesr30@vcu.edu
https://www.ricky-jones.com/

About Me

General Information

Hi there! My name is Ricky Jones and I am a high school student that's interested in programming, video editing, engineering, 3D modeling, and photoshop. I go by the pseudonym 'DucksIncoming' online, and most media I produce is under that name.

Interests/Hobbies

The majority of my hobbies are related to computer science and creation. I enjoy building and programming engineering projects, most of which are in the following section of this portfolio. Additionally, I enjoy video editing, graphic design, 3D modeling and printing, and more. I try to diversify my hobbies and interests across a variety of disciplines, however my main focus is programming and engineering.

Skills

I have experience in a variety of programs and programming languages. I primarily use Visual Studio Code for my programming needs and I am knowledgeable in Python, Java, JavaScript and Typescript, C#, and I am currently learning React.js. I've spent a great amount of time working in Blender with 3D modeling over the past six years and become proficient at its use. I have been working in Photoshop for around three years and I am also currently learning Adobe Illustrator. Finally, I have done video editing work in DaVinci Resolve, Adobe Premiere, and Adobe After Effects.

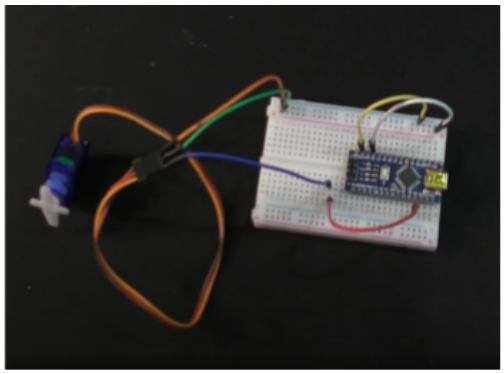
Contact

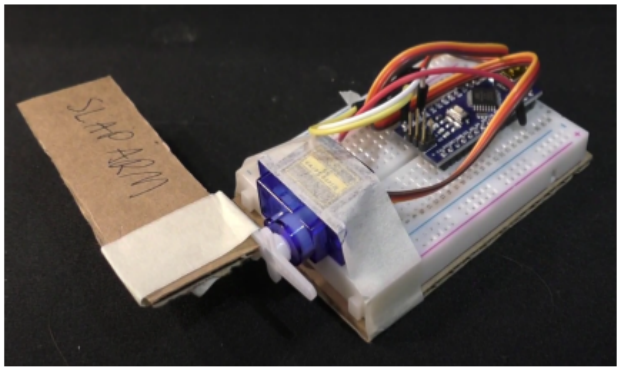
You can reach me by email or view my other contact options on my website linked above. I'm more likely to respond via email, however If you would like to call/text me, please email me requesting my number first for privacy reasons. You can also direct message me on social media like LinkedIn if you'd like, however I will likely not be as quick to respond. If in doubt, just stick to email, thanks!

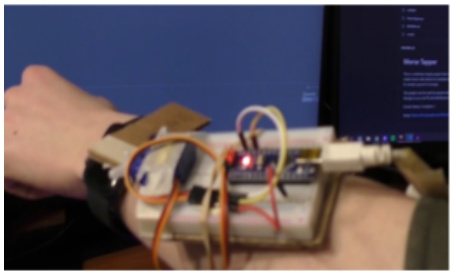
Please note: This is not the most recent version of this document, it will be updated shortly. Please email me about any further information you need.

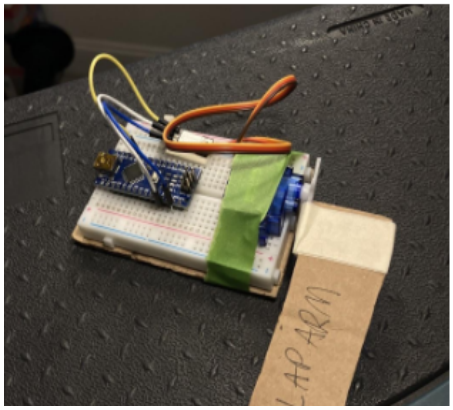
Ricky Jones
jonesr30@vcu.edu
https://www.ricky-jones.com/

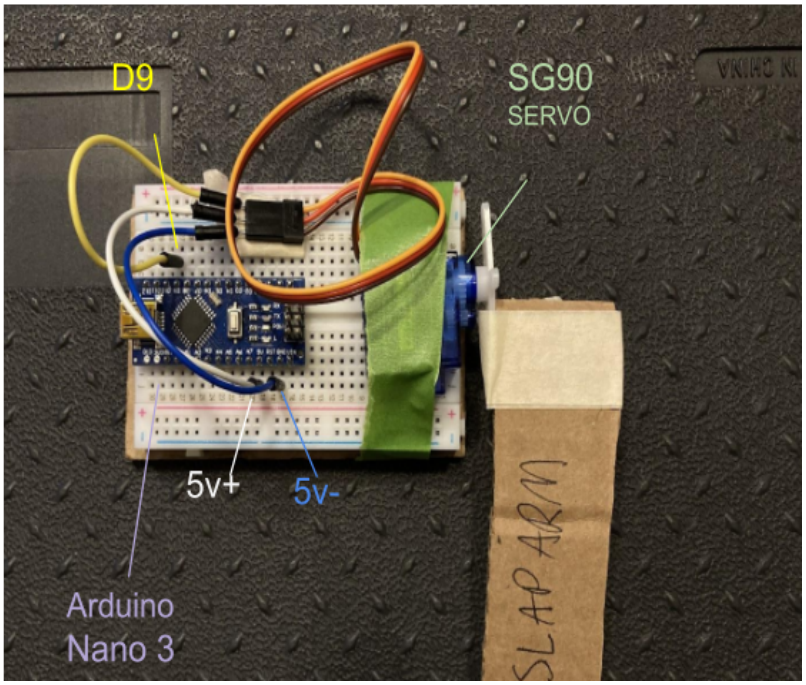
Morse Tapper











Objective

Construct and program a mechanism that translates speech spoken into a microphone into understandable morse code tapped on the arm of the person wearing the mechanism. This project could have applications for people who are hard of hearing, deaf, or have trouble processing speech as it is spoken, especially as it does not require other speakers to understand a separate form of communication such as sign language. It's performance is dependent primarily on its ease of use and reliability.

Ricky Jones
jonesr30@vcu.edu
https://www.ricky-jones.com/

Technical Explanation

This project was made using Python and Visual Studio Code. In order to produce the results expected, there are three primary components that must be considered and fabricated. The first of these components is the speech recognition module. In order to translate the speech spoken into the microphone into morse code, the spoken speech must first be converted into text for the computer to parse. Using the Google Cloud's speech recognition API, along with a Python library that allows easy access to its functionality, this component can be implemented after some settings are tweaked. The second component is the morse code translator that translates the speech from the microphone into morse code. Because morse code operates using individual letters rather than entire words, a function that intakes a single character and returns the morse code equivalent can be implemented for each individual character of the returned speech from the speech recognition module. Finally, the physical tapping mechanism must be built as the final component. The tapping mechanism is built using an arduino nano with a breadboard that is connected to a SG90 servo motor. This motor receives commands from the Python program dictating the position it must rotate to. The morse code output from the morse code translator can be passed into a handler function that dictates the timings for the servo motor given standard morse code timing and a global variable that alters its overall speed while retaining the proportional differences in certain timings. Finally, using the pyfirmata library to interface with the microcontroller, these commands and timings can be sent to the morse code tapper. Following this, the tapper mechanism can be mounted to the arm of a wearer and the project will be fully functional.

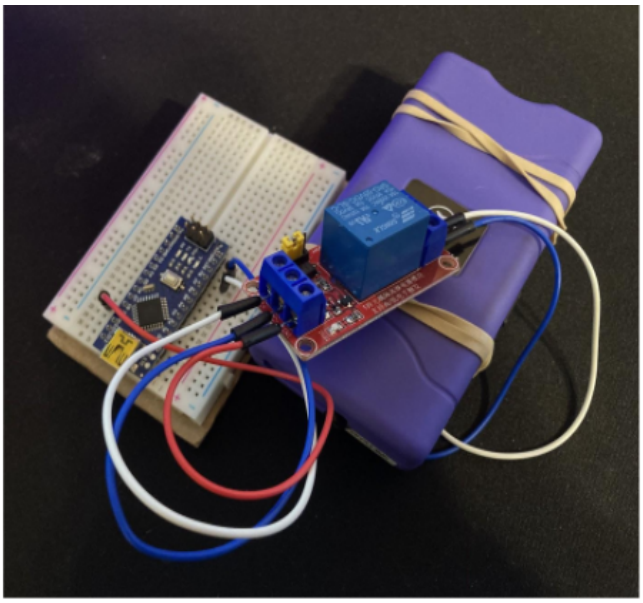
Link to Demonstration/Explanation: https://youtu.be/BU7rzuCXGSs
Source Code: https://github.com/DucksIncoming/Morse-Tapper

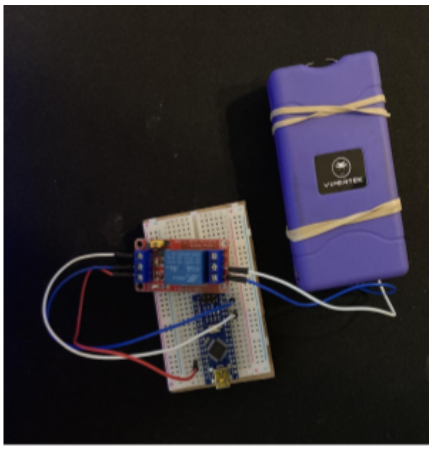
Results

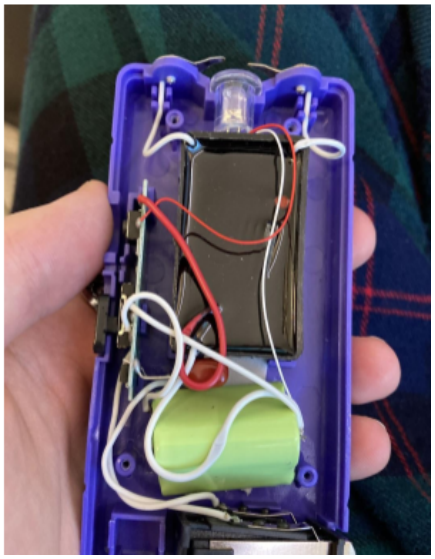
Overall, the project worked extremely well and functions exactly as intended. It's design is compact enough to be mounted to the wearer's arm and the responsiveness of the speech-to-text API makes utilizing the project reasonable. The project, however, suffers from a few core issues. The speed of the tapper makes longer sentences tedious and difficult to retain in memory when coupled with the requirement of translating the morse code back into english for the wearer. The project in itself serves as more of a learning opportunity and proof of concept, however, therefore these issues are not of extreme importance. Overall, the project was a great success and assisted in my process of learning arduino and servo functionality.

Ricky Jones
jonesr30@vcu.edu
https://www.ricky-jones.com/

Taser Chess







Objective

Construct a chess game for entertainment that shocks the user using a low-power taser when they make a poor move. The purpose of this is largely for entertainment and not meant to cause significant pain or damage to the players, but rather to serve as a fun, seemingly more risky alternative to traditional chess. The performance of the project was analyzed based on the effectiveness of the program deciding which moves were badly made and the amount of enjoyment the players expressed after playing.

Ricky Jones
jonesr30@vcu.edu
https://www.ricky-jones.com/

Technical Explanation

This project was made using Python, pygame and Visual Studio Code. At its core, this project consists of a chess position evaluator, a move scoring system, and a taser that can be activated through code. First, the chess position evaluator was developed. Using a subpackage of Python's chess library, the popular Stockfish chess engine can be implemented into the project. Because Stockfish focuses on automatically generating moves, however, it cannot be simply referenced in its basic form. By digging through the Stockfish source code, however, the evaluation function can be extracted and isolated. By forcing the position of the visible chess board to the position Stockfish keeps in memory, we can achieve an accurate evaluation of the board at its current state. After the player makes a move, the board is reevaluated using the same function. The difference between the two scores provides an excellent approximation of the overall board value lost through the move. Because Stockfish sees multiple moves in advance by calculating the best possible move using min-max optimization, it is incredibly unlikely that the player will make a move that is then evaluated as better than the move selected by Stockfish. Because of this, we can simply evaluate the difference between the two scores to determine the severity of the move. If the difference is past a certain threshold, the taser is activated for the player who made the move for a short period of time. The taser is activated using a signal sent through one of the digital pins on an arduino nano to a relay that is connected to two wires that were soldered to the taser's button pins, thus activating the taser. Another digital pin is used for the taser for the other player to minimize cost. Once the taser is set up and the code is complete, the program can be run and played.

Link to Demonstration/Explanation: [ADD HERE]
Source Code: https://github.com/DucksIncoming/Taser-Chess

Results

While this project has very few genuine real-world applications, its real purpose was to further my understanding of microcontrollers, electronics, and programming using pyfirmata. Additionally, the technical challenge of analyzing chess game states and parsing the criteria to classify moves as well or poorly made improved my programming skills greatly and introduced me to new coding concepts that can be applied to future projects. Overall, the project worked as intended and greatly improved and reinforced my knowledge of electrical engineering and programming.

Ricky Jones
jonesr30@vcu.edu
https://www.ricky-jones.com/

Weeping Angels

Objective

Create a statue inspired by the concept of weeping angels that detects when it is not being looked at and turns to face a person in this room. While the project has little real world application on its own, aside from entertainment and decoration, the use of facial detection technology and acting based on that data opens the doors to a variety of possibilities. The success of this project was evaluated based on the reliability of the weeping angel and the success of the effect being achieved.

Ricky Jones
jonesr30@vcu.edu
https://www.ricky-jones.com/

Technical Explanation

This project was made using Python, blender, and Visual Studio Code. For this project, there are a few clear aspects that must be considered and built. Those are the facial recognition module, the face position logger, the loss of face detection, and the head rotator. First, there needs to be a camera feed to find a face in. Using a small, 2.3cm camera, a feed of the angel's view can be obtained in a discrete way. Multiple cameras can be used to increase the angel's effective range, however using only one functions well enough that it can be used with just one. By using Google Cloud's facial recognition API on the camera feed, the angel can find faces in its environment. For each frame the facial recognition is run, the variable lastFace can be set to the position of any face that is found. When no face is found, the variable does not change. Once there is no face detected, the loss of face detection function can send the contents of lastFace to a function that moves a servo motor one degree at a time. If the facial detection picks up another face, it immediately tells the servo function to stop, so as to not spoil the illusion. Once all of these pieces are put together a weeping angel to hold the electronics can be constructed either out of household materials or 3D printed. In this case, 3D printing seemed a better option. Once everything is put together and the angel is turned on, it is ready to be used.

Link to Demonstration/Explanation: N/a
Source Code: https://github.com/DucksIncoming/weeping-angels

Results

Overall, the project worked nearly perfectly and achieved the desired effect. The experience of turning away from the angel and turning back to see it staring at you works perfectly and is an effective illusion. Through this project, my understanding of facial recognition and arduino was built up and reinforced effectively. With these newly developed techniques, I hope to further pursue projects utilizing facial recognition in the future.