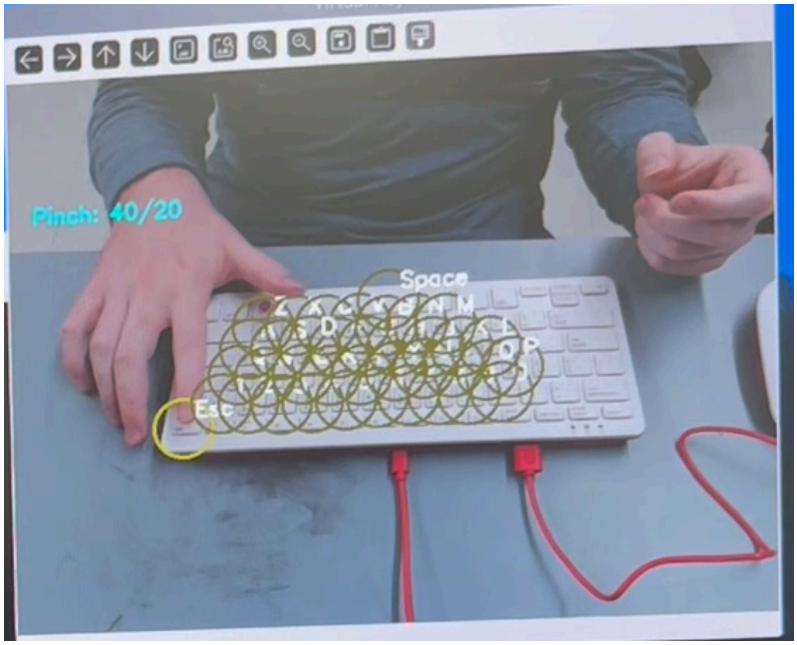
EC535 Final Project Report: Virtual Keyboard

Peter West, Kevin Connell

https://github.com/Kconn7/EC535_Final_Project.git

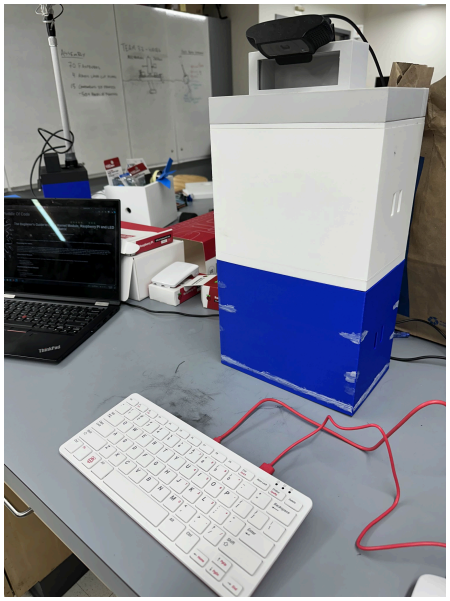
Abstract:

Traditional physical and touchscreen keyboards present limitations as the world heads toward a technologically integrated society, where seamless and intuitive human-computer interaction is increasingly essential. In augmented and virtual reality environments, conventional input methods can be cumbersome, impractical, or entirely unusable. This project aims to create a virtual keyboard system that enables gesture-based typing on any flat surface using a single webcam. A Raspberry Pi 4 microcontroller processes camera footage using computer vision libraries such as mediapipe to dynamically track a user's hands and fingers and openCV to detect the user's desired key.



Introduction:

The virtual keyboard is an embedded system device that allows users to type on any flat surface. By utilizing computer vision techniques, the system tracks hand and finger movements to interpret user input without the need for a physical keyboard. Our approach transforms any surface into an interactive keyboard, offering a portable and versatile typing solution. The system employs a Raspberry Pi 4 and mediapipe to track and process hand gestures to output a user's keystroke. This system can be implemented for a variety of real-world applications, such as augmented reality. The virtual keyboard can be used in place of traditional keyboards to allow for easy typing in any location. This project may also have potential applications for assisting individuals with limited dexterity or hand functionality.

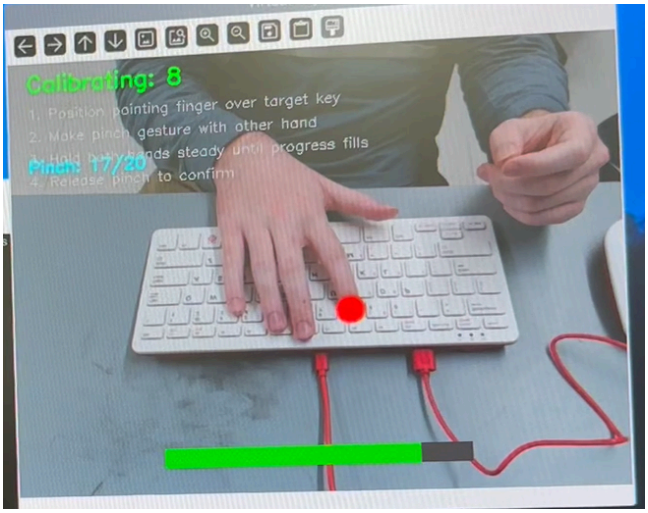


Methods:

The system's hardware consists of a Raspberry Pi 4, a Logitech C920 Webcam, and a monitor. The Raspberry Pi 4 serves as our embedded system and utilizes the X11 backend. The

Pi processes the captured webcam video feed of a user's hands to then be displayed to a monitor with the system's output. A reference keyboard is used for calibration, but is not needed when using the virtual keyboard.

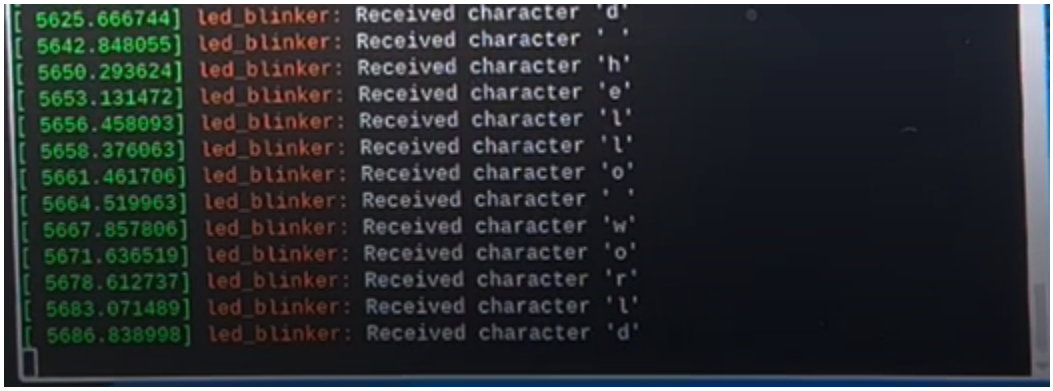
The software side of the system consists of a Python script to process each frame with mediapipe to track the user's hands. The user's primary hand is used to hover over a key while their non-dominant hand confirms the keystroke via clicking their pointer finger and thumb together. Upon initial use, the user is guided through a calibration phase to set up detection regions for each key. After calibration, the system enters the typing phase and requires no further calibration. OpenCV will then display to the user a feed from the webcam and the detection regions for each key.



An older version of this system previously used a BeagleBone to stream the webcam data over a UDP socket to an external laptop. UDP was used to maintain fast processing speeds, as TCP introduced delays that made the device difficult to use. The laptop then processed the image data through the Python script in much the same way the current system does now. The switch to the Raspberry Pi 4 was done to better optimize the system's performance.

Results:

The setup for this system involves connection the webcam and monitor to the USB and mini HDMI ports of the Raspberry Pi 4. The webcam is then mounted 17 inches above a reference keyboard used during the calibration phase. The user can choose to configure the height of the webcam to their liking however, this height was found to be ideal for hand detection with mediapipe. The Raspberry Pi is capable of running and displaying the processed camera feed at 10 fps to 15 fps and a measured latency of one second. When the user enters a keystroke through the virtual keyboard, the script writes the key pressed to the writable character device as a kernel module. The kernel module then reads the written character and then prints the character to dmesg. This could be expanded into a keyboard controller for future iterations.



Limitations:

Our biggest limitation was that of our original microcontroller, the BeagleBone. The Beaglebone lacks the needed processing power to handle the webcam data, process each frame, and displaying the camera feed to the user. While we had an original implementation that functioned entirely off the Beaglebone, its slow performance of 1-2 fps and latency of around 5 seconds made this implementation impractical. We came up with a temporary solution to this problem by streaming the camera feed from the Beaglebone to a laptop through a UDP socket connection. The laptop then handled the post-processing with mediapipe and detected keystrokes

through a Python script. This resulted in a working version of our keyboard, but still with some delay. A TCP connection was also considered, but the transmission delay introduced by TCP was over 12 seconds on top of preexisting processing delays.

Our final iteration of our virtual keyboard utilized a Raspberry Pi 4 to significantly increase the processing power of our system. We no longer needed to stream the camera feed and instead relied entirely on the Pi. Libraries such as mediapipe needed to be configured to be run on the ARM processor, and additional libraries were installed to get the system running. Implementing our kernel module also took some work, as we had issues with the Pi recognizing GPIO and not allowing us to activate the LED through a keystroke. We pivoted around this issue and instead chose to have our kernel module read the character written by the keyboard and display the character through dmesg. For future iterations, we would like to have the system boot on start to streamline the keyboard, add optimizations to minimize latency, and look towards adapting our kernel module to operate a GPIO LED, physically acknowledging our key presses. We would also like to see if we can implement our system, like pyautogui, to perform keyboard typing functions in the Pi or any device that will use the Pi as a keyboard device.