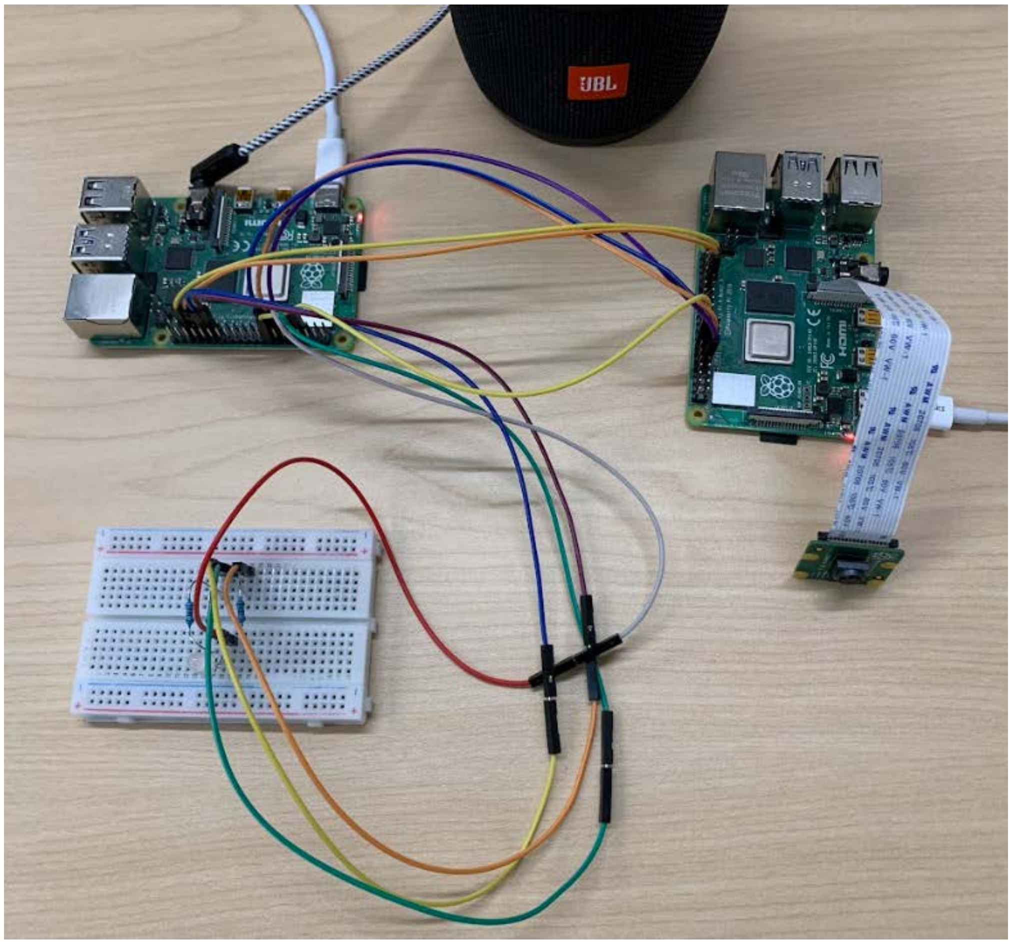
# Gesture-Controlled Speaker

Isabel Holtan, Dale Kercorian, Renee Li, Kevin Pere, Eli Smith, Teddy Withey



# Table of Contents

## Background

Our team designed and built a standalone speaker system capable of being controlled via simple hand gestures. There are a variety of benefits gesture control can provide including being touchless and not requiring physical control mechanisms. In a post Covid world, cleanliness and controlling the spread of germs is a very important aspect of society. Our system only relies on a camera viewing you and the gestures you make which means no physical contact of any sort, as well as the possibility of social distancing among those using this device. We also hope that this technology could potentially be used for accessibility with people with accents and speech impediments where traditional voice-controlled speakers are difficult or even impossible to use.

We believe that the most interesting element of this project is the adaptability of gesture control and its ability to work as a control mechanism for products in the quickly growing IOT space. Our system focuses on controlling a speaker, but the same technology could be applied to door systems, lights, kitchen equipment, or anything else where touchless control is a priority.

Currently, there are no standalone gesture-controlled speaker systems on the market. However, gesture control is beginning to gain traction as a control mechanism as some cars, like BMWs, incorporate gesture control as a means of accepting calls and setting navigation destinations.

## System Architecture

Our project is composed of three main phases: data acquisition, data processing, and decision making & actuation. Figure 1 illustrates the flow between these three phases.

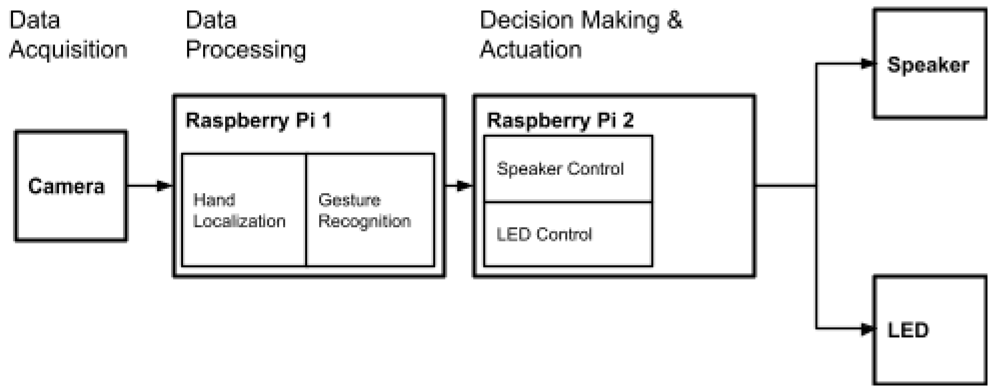


Figure 1: Three Phase Speaker System

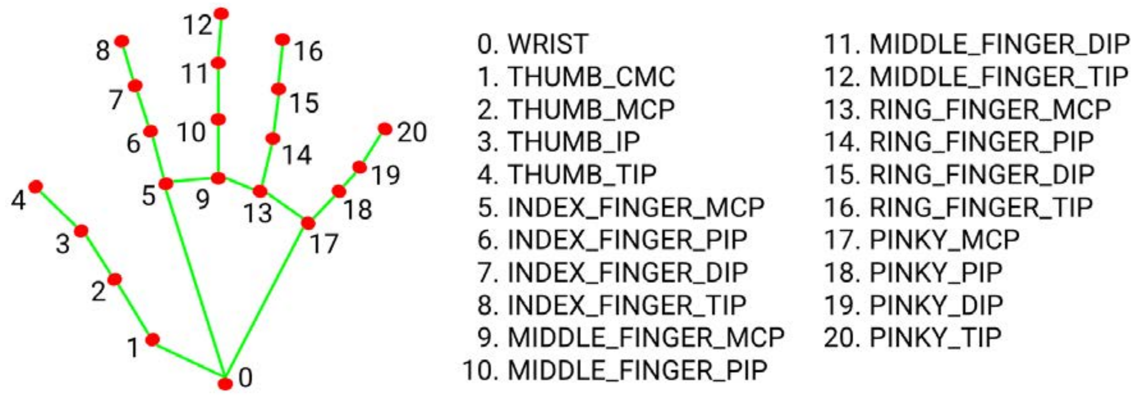
### Data Acquisition

In the data acquisition phase, a Raspberry Pi (RPi) camera continuously captures video from the environment. The camera captures frames at a maximum rate of 30 fps. These frames are transmitted directly to the microprocessor (RPi 1) via a ribbon connector.

## Data Processing

Data processing is done on the same microprocessor (RPi 1) as data acquisition. Using a still image from the camera, the system first identifies where the hand is in the image (hand localization) and then determines what gesture, if any, that hand is making (gesture classification). In the case of multiple hands, the system focuses on the hand that is closest to the camera.

Hand localization is done using Google's MediaPipe library, which is an open source suite of software offering "cross-platform, customizable ML solutions for live and streaming media." Of the many options offered, including face and body detection, segmentation, and motion tracking, we utilize the "Hands" solution. This program first detects human hands in a frame and then applies a hand landmark model to identify 21 key points on the hand. The landmark model key points can be seen on the hand shown below in Figure 2.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 2: Hand Landmark Model

In the next step, the 21 landmark points identified by MediaPipe are fed through our custom gesture classifier, which is trained to recognize the five gestures shown in Figure 3. Our classification model is a multi-class support vector machine (SVM), with c=1.0, gamma=31.6. The only preprocessing step taken is subtracting the wrist coordinates from the rest of the key points to ensure that gesture classification is position invariant.

Once a gesture is detected and identified, it writes LOW to a particular GPIO pin corresponding to that gesture and command. This pin is connected to the same GPIO pin on the second Raspberry Pi (RPi 2), which is set to trigger on pin lows to begin the decision making and actuation phase.

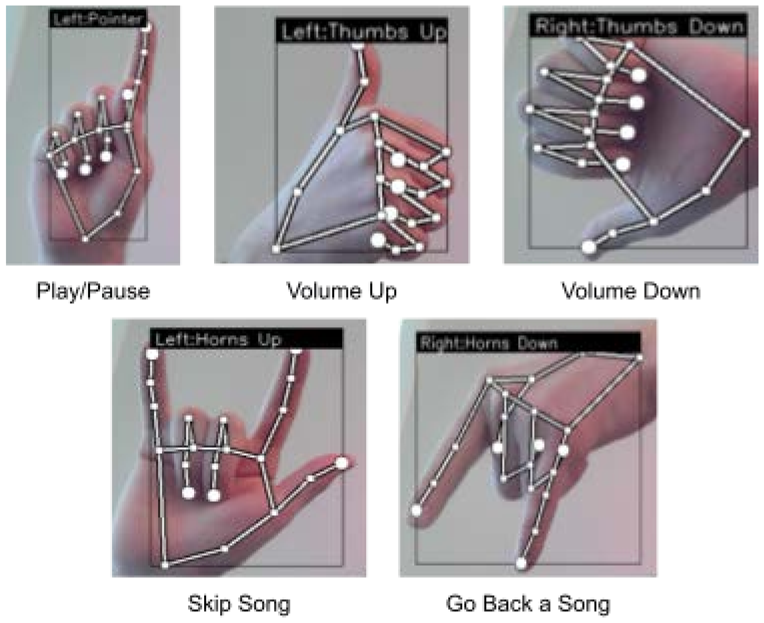


Figure 3: Detectable Gestures

## Decision Making & Actuation

The second Raspberry Pi (RPi 2) runs a script that includes interrupts for each possible gesture. If a pointer figure is detected then the music will turn on or off depending on its current state. A thumbs up increases the volume, and a thumbs down lowers it. Finally a horns up gesture will skip to the next song and a horns down gesture will go back a song. The speaker is controlled using VLC and is connected to the microprocessor via an aux cord. There is also an LED that changes color (blue to play/pause, green to raise volume, red to lower volume, teal for next song, purple for previous song) depending on the gesture detected. This is to give feedback to the user so that they know that their gesture has been received.

We can see the five main parts of this system labeled in Figure 4 below.

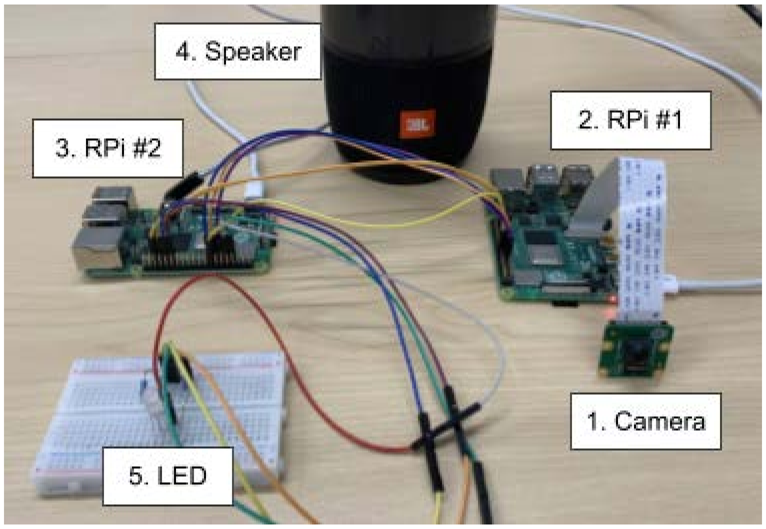


Figure 4: Labeled System Architecture

## Technical Issues

We originally planned to use an IR sensor to detect if someone was within range of the camera. If someone was in range, then the system would begin capturing frames and looking for gestures. However, a few problems began to emerge when putting the subsystems together. On their own, each of the IR sensors we tested were finicky and not reliable at long ranges. We were only able to somewhat accurately detect a hand within 20 to 50 cm. When a hand was detected within this range, a serial output would be sent to the Raspberry Pi to turn on the camera and begin the gesture detection portion of the code. This startup process began to cause problems. Without the IR sensor, the camera would run at about 10 fps, but with the IR sensor, the camera only ran at around 2 fps. We ultimately decided to remove the IR sensor from the system because the drop in fps was too detrimental to the system to justify.

Another problem we ran into during initial system integration was that the singular Raspberry Pi was not properly playing music when most of its processing power was dedicated to running our gesture detection code. To solve this, we added a separate Raspberry Pi to control the speaker and LED. This way, the first Raspberry Pi using all of its computing resources to detect gestures would not affect the playing of music and control of the LED.

Some more technical issues we faced included tracking hands that were far away from the camera when there were closer hands available to track. We solved this issue by raising the confidence required for MediaPipe to consider something a hand. A last technical issue we ran into was forgetting to ground the GPIO pin connections. This made the GPIO connection between the Raspberry Pis very noisy.

# Testing

## Software

### Hand Localization

We tested the MediaPipe hand localization module in a variety of different rooms with different lighting conditions so we could be confident that our system would work in the EECS atrium for the MDE Design Expo. We also tested the system with different values for the confidence parameter to see what the best balance of detecting things that might be a hand versus might not be a hand was.

### Gesture Classification

We developed our gesture classification model from scratch by testing a variety of models and choosing the best performer, optimizing for macro average accuracy while considering computational limitations of the Raspberry Pi. As shown in Figure 5, the best performing methods were nearest neighbors and SVM with a RBF kernel; we chose SVM for the final model since nearest neighbors is a non-parametric method that requires uploading the dataset and performing many calculations for new predictions. We also tested various methods of preprocessing (normalization, standardization) and dimension reduction (PCA, random projection, feature agglomeration), but they did not significantly improve performance.

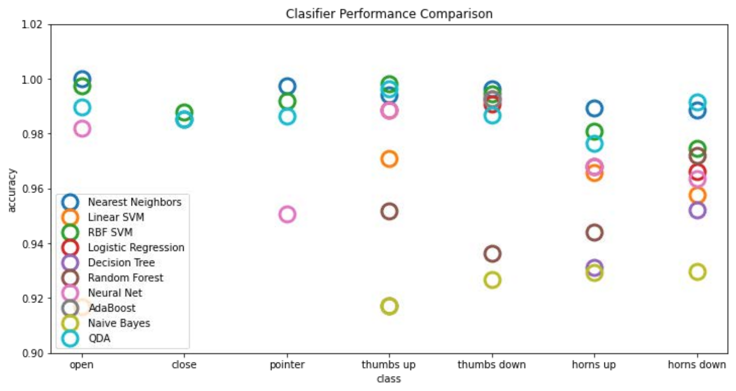


Figure 5: Per-class accuracy results of different classifier methods

## Hardware

We tested several different IR sensors that were designed to be used in conjunction with our camera in order to save processing power. Most of the sensors didn't have a significant enough range to be effective with the rest of our system, and the sensors that did have adequate range (time-of-flight) were not reliable enough to be used in our final design. The measurements of the

IR were accurate, but when it came to transmitting the information to a RPi ,we found that the methods to transmit this data weren't consistent enough to use in our system. We tested both serial and GPIO communication but found that neither were satisfactory.

To test the speaker before integrating the whole system, we passed in dummy gesture numbers to make sure that each function worked as they should. With a small playlist of a few songs, we would start the music and perform the same command every five seconds to make sure that the speaker could pause/play, raise/lower volume, and skip and go back to previous songs.

## Integration

When integrating the entire system, we used our LED system to ensure that the system was picking up the correct gesture. In our gesture recognition program, we would also print out comments saying which gesture was being recognized. To communicate between the two RPis we were using in our system we decided to use a direct GPIO connection. In order to test this communication system, we read what the input GPIO pins were being sent and then confirmed whether this was the result we desired.

## Demonstration

At the MDE Design Expo we had the full system set up and allowed users to control the speaker using the 5 hand gestures we had trained the system on. A monitor was connected to the Raspberry Pi to allow users to see the 21 points that MediaPipe mapped on their hand as they made gestures. One thing we noticed at the design expo was that people often held their hands too close to the camera for it to properly detect their gestures - the gesture detection “sweet spot” was between 0.5 and 2.5 meters away from the camera. We also noticed confusion related to the timing of the system; we added a 2 second delay between commands to ensure a gesture is not accidental or repeated, but some users attempted shorter or longer timing. This delay could be tuned to be more appropriate for all or specific users. Figure 6 shows what our setup looked like.

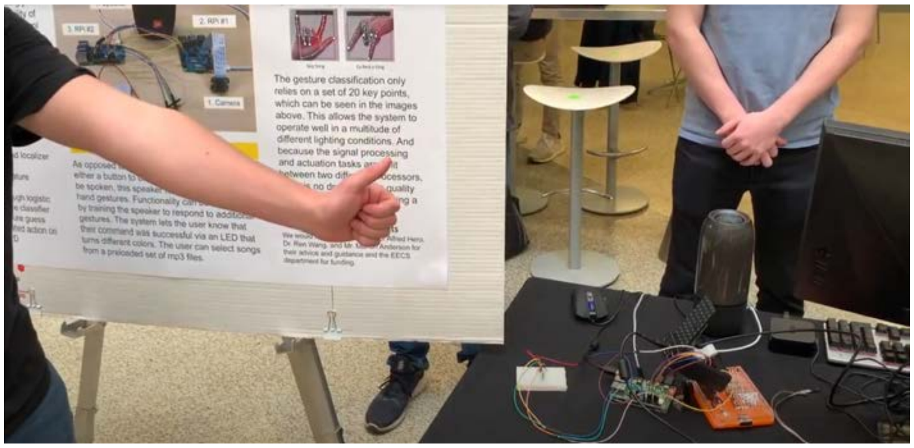


Figure 6: System set up at MDE Design Expo

## Project Milestones

### Milestone 1

For milestone 1 our goal was to make progress on each individual subsystem.

For the hand localization and gesture classification system, our goal was to figure out what gesture an image of a hand was making. We tried many different approaches, starting with using transfer learning to train our own neural network. This proved to be difficult as we had to download a large number of images, preprocess them, and train for a long time. The results were mediocre because of the lack of training images and the system was slow, so we switched to using Google's MediaPipe library for hand localization.

Once we made the switch to using MediaPipe, the gesture classification task also became much easier because instead of processing an entire image of a hand, we only had to process 21 keypoints. This opened the door to many types of multiclass classification algorithms, and we set up our software so that any model could be used in the system. We ended up with an image processing pipeline that could take an image of a person making a hand gesture and figure out what gesture they were making. This pipeline worked for most different lighting conditions and gesture orientations.

We also chose to use a speaker with an aux connection (one that could be bought at Best Buy or Walmart) to easily connect it to the RPi. To control the speaker, we decided to use VLC Media Player because it has an importable Python library with functions that allow you to play a song, pause a song, and set a volume number. Once the library was imported, we were able to play different songs on the speaker using dummy values.

For the LED feedback system, we planned to have each gesture assigned to a different number, and each of those numbers assigned to a different color LED. We used an Arduino to power the LED and were able to control the light with different input numbers as planned. We later changed the LED to be controlled by an RPi GPIO connection. Additionally, there was a second LED that would turn blue when an object was detected from 20 to 50 centimeters away from the original IR sensor for more than two seconds. Ultimately, we decided to remove this feature in the final system.

### Milestone 2

The goal for milestone 2 was to have a minimally viable system. The system would be running on a Raspberry Pi and would have a speaker, an LED, and an IR sensor. We were able to achieve a fully integrated system that included two Raspberry Pi's, the speaker, and an LED. An IR sensor was not included because it reduced the reliability of the system. It cut the fps of the camera in half and made the system less reliable in detecting gestures. The goal of the IR sensor was to reduce the effect of motion in the background and reduce power consumption. We were able to achieve this goal, and maintain the system reliability, by increasing the confidence threshold of the logistic regression gesture classifier. Additionally we did not have problems with power consumption so the camera could be run continuously. Another goal for milestone 2 was to have four gestures mapped to four commands. By milestone 2 we were able to have five gestures mapped to the following five commands: raise volume, lower volume, pause/play, skip song, and go back a song.

### Stretch Goals

Our first stretch goal was to allow the user to connect to the speaker using Spotify or Apple music. We did not implement this functionality because it confused the purpose of the speaker. If the user had spotify open on their phone, then they could control the speaker at a distance via bluetooth, similar to how gesture detection lets them control the speaker at a distance. Instead, music was downloaded to the Raspberry Pi as mp3 files. Our second stretch goal was to have the system work in dynamic environments. For example the system would be robust to different types of lighting and background movement. The Google MediaPipe hand localizer was very good at detecting hands in these different conditions and producing the 21 hand key points. Because we were only using 21 points and not a full image, the linear regression gesture classifier was also robust to varying conditions. Our final stretch goal was to have a protective box that would encase the speaker, sensors, and light system. We found that there were several Raspberry Pi cases with camera stands available online. As we were no longer using an IR sensor and the speaker was too large to put in any sort of additional casing we decided to order one of the online cases. This served the purpose of protecting the Raspberry Pi and angling the camera correctly.

## Future Work

To improve our system, we would like to have integrated a second sensor that is more robust than the IR sensor. This sensor would allow us to run gesture detection only when an object is in range for a certain period of time so that our RPi does not need to be running constantly. This would save power and prevent the speaker from responding to background movement. We would need a better quality sensor than the IR sensor previously used so that it can detect objects farther away and in a more focused region. We would also want to add an LED to give feedback if the user's hand was in range. Often a hand would be too close or off to the side, so the LED would tell the user if their hand was visible to the system. Another change would be to have the code run on startup on the RPi. This would make the system easier to use instead of needing two monitors and keyboards to control the two RPi’s. Finally we would want to create casing that encompasses the entire system. This would make the system more portable and robust.

## References

Google Mediapipe Library

https://google.github.io/mediapipe/solutions/hands.html

Hand Gesture Recognition using Mediapipe Github Repository

https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe/blob/main/README_EN.md

## Appendix

GitHub repository with all of the team's deliverables, code, videos, etc

https://github.com/googolplex8/2022W-Team-7-repo