iContact

Authors: Heather Baker, Anna Li, Edward Lucero: Electrical and Computer Engineering
Carnegie Mellon University

***Abstract*—iContact is a mobile camera system that points directly at whomever is currently speaking, whether it is a single user moving around in a room or a conference room full of people. Using both audio detection and facial recognition via computer vision, iContact can identify the speaker and maneuver the camera to face that person. Its motors can rotate the camera side-to-side, tilt up and down, and raise and lower. With iContact, video calls will be made to feel more personal.**

***Index Terms*—Haar Cascades, I2C, I2S, Jetson Nano, MIPI CSI-2, OpenCV, PWM**

I. Introduction

WITH the onset of COVID-19, video calls have become an absolutely indispensable part of everyone's daily lives, whether it is for attending lectures via Zoom, calling friends and family, or even remote internships, most people cannot go a day without a video call anymore. Even before COVID happened, people have needed video calls for keeping in touch with distant friends and family, and many companies have relied on conference calls for linking their various branches and workers around the world.

The world has seen how video calls have become increasingly crucial over recent years, but video call mechanics have not really evolved much – conference calls are all still primarily done through a laptop camera or a desktop webcam. The question we asked was: How can this project better immerse the remote viewer into a video call? The answer is iContact, an agile camera that keeps the focus on the speaker in any conversation by physically adjusting to center on the speaker's face.

There are four areas of functionality that the design requirements categorize into: conference viewing, working range, algorithm accuracy, and speed. For the viewing requirement, iContact should be compatible with any conferencing software and able to operate at 1080 pixels at 30 frames per second. For the working range, iContact will aim to have a 360-degree field of view, one foot of vertical panning, and ten feet of microphone audio pickup and person detection radius. The requirements for algorithm accuracy will be set for 90% with respect to centering, speaker identification, and cerebral command comprehension, as well as 95% motor positioning accuracy. Lastly, within the speed category, iContact should complete motor positioning adjustments, audio processing, and video processing within one second. The speeds for audio and video processing are important to have minimal lag between the conferencing video feed and iContact.

II. Design Requirements

There are different tests for the various areas of functionality. To meet the requirement to work with any conferencing software, iContact will be tested on Zoom, WebEx, and Google Hangouts. The frame rate requirement can be determined by counting the number of frames that get sent between iContact and the host computer within a certain amount of time. Regarding the working range and algorithm accuracy requirements, there are two tests to be performed: a stationary speaker test and moving speaker test. The stationary speaker test will be conducted at varying distances and heights from iContact to test the distance of the microphone pickup range, the person detection radius, the vertical panning range, and the centering accuracy of an out-of-frame speaker. The distances will be between three to fifteen feet, in and out of frame. The heights will be set such that the speaker's head is above and below the camera frame as well as above and below the center of the frame. The moving speaker test will also be conducted at varying distances and varying heights. The distance and height ranges will be determined by the working range of iContact found from the stationary test. The moving speaker test will be used to verify centering accuracy, field of view, and vertical panning range. In addition to the previous two tests, there is an additional multiple speaker test, which will gauge how accurately iContact is able to identify speakers. All previous tests mentioned will also record the time between start and speaker centering to determine how well iContact meets the speed requirements. One additional test to specifically test speed is to have multiple speakers conversing back and forth for varying speaking time durations.

III. Architecture and/or Principle of Operation

There are two main parts to the hardware design: the base and the elevator. The base has three microphones and one stepper motor that rotates the cameras around the yaw axis (see Figure 6). Another stepper motor is used to control the elevation of the elevator. On the elevator, there is a platform that holds the two cameras as well as the two micro servos. There is one micro servo for each camera to adjust the pitch of the cameras (see Figure 1).

The main computing is done through the Jetson Nano (see Figure 3). Connected to the Nano are two cameras, three microphones, and a motor HAT. Connected to the motor HAT are two micro servos and two stepper motors. The motors are on a separate power supply from the Jetson Nano.

For the overall software design of the project there are three separate components that interact with the main program. These three components are software programs that control the audio processing, the motor controller, and the computer vision software. The main program is responsible for the communications between other components by controlling when they start up and making use of the return values from each program.

The system first starts out in an idle state where the camera positioning is the most recently set angle from either initial startup or from the most recent speaker detected (see Figure 2). The first step is to wait for the audio processing component to send its results to the main program. The Jetson analyzes the three microphones' audio feeds to determine the difference in time it takes for the acoustic source's sound to reach each of the microphones. Using these time differences, it can then calculate the angle by which the camera must rotate (relative to the yaw axis) to point in the general direction of the speaker, which it sends back to the main program.

Once the main program receives its instructions from the audio component, it can then send the instructions to the motor controller. The motor controller is responsible for turning the specified motors to the desired angle. By using the Adafruit MotorKit Library, motor movement can be done by specifying the angle to turn to for the micro servos and the angle amount to turn by for the stepper motors. For each of the stepper motors, a counter tracks the current angle relative to the initial angle, to calculate the degrees the stepper motor needs to turn. Once the motors have set the angle to the correct angle, it responds back to the main program that it is finished. The angle calculated by audio detection points the camera in an approximate direction of the speaker. Computer vision face detection is then utilized to perfectly center the speaker in the camera's view, through precise adjustments. Thus, the next step is for the main program to tell the computer vision component to start looking for a speaker, as well as which of the cameras' views the speaker is in.

After the computer vision portion has started up, it looks in the specified video feed for the speaker. There are now two paths that can be taken here. If a speaker is not in view, that implies either the speaker is below the camera or the audio estimation was not good enough. In the former case, the camera is adjusted until a face is detected or it is in the bottom most position. If iContact still has not detected a face, then we assume that it is the latter case. In the latter case, the program tells the main program that a speaker has not been found and that it should just wait for the next detection. If a speaker is detected, the final recentering for the speaker can then be computed by telling the main program the final set of instructions. This is a single complete loop of the software state machine.

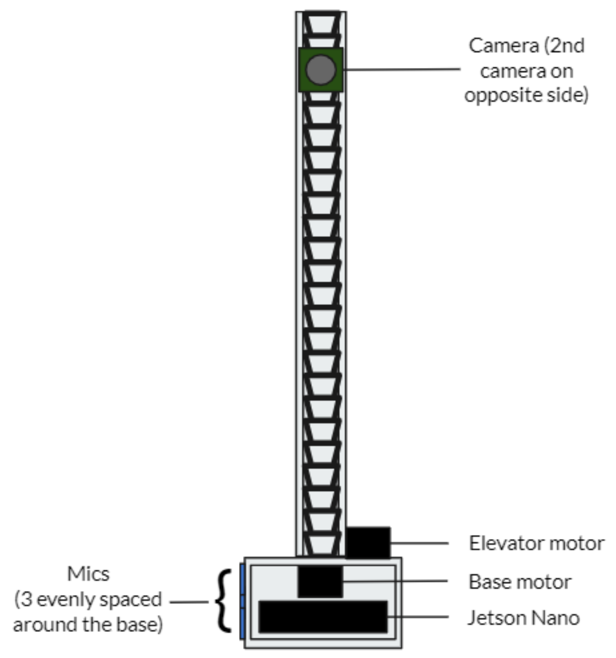

Camera (2nd camera on opposite side)
Elevator motor
Base motor
Jetson Nano
Mics (3 evenly spaced around the base)

Fig. 1. iContact mechanical design

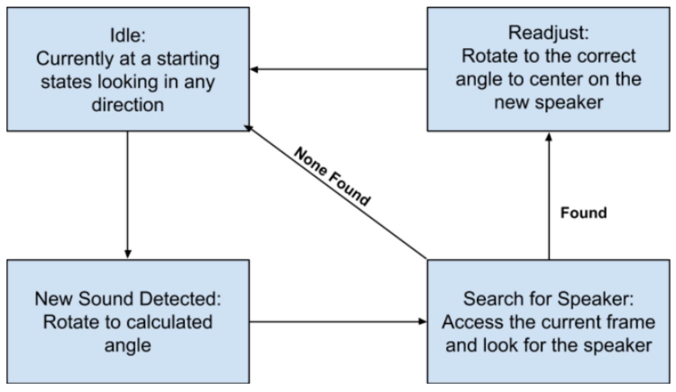

Idle: Currently at a starting states looking in any direction
Readjust: Rotate to the correct angle to center on the new speaker
None Found
Found
New Sound Detected: Rotate to calculated angle
Search for Speaker: Access the current frame and look for the speaker

Fig. 2. Software state machine

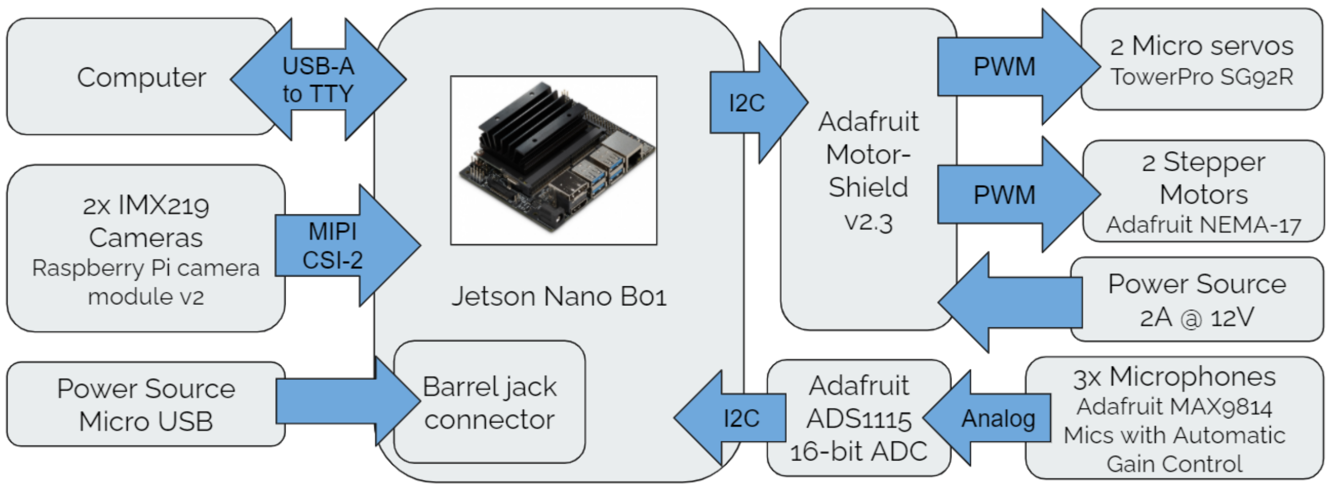


Fig. 3. Hardware system specification

IV. Design Trade Studies

The following are justifications for the component selections and any changes that were made from the project proposal.

A. Jetson Nano vs. Raspberry Pi

We opted to use the Jetson Nano for iContact. As shown in Table 1, the Jetson Nano provides more power for video decoding and better support for peripherals than the Raspberry Pi does. A big advantage of the Jetson Nano is the increase in MIPI CSI-2 lanes as this increases the number of cameras we could use, which reduces the latency for speaker detection (see Section G). The Jetson Nano also has I2S for audio, which is a better choice since it is digital, resulting in less noise compared to analog and thus better audio quality. While this was our original plan, it was later changed as detailed in Section C.

Originally, the increased support of peripherals provided by the Jetson Nano allows for less additional boards that would be needed for a Raspberry Pi; however, as the project evolves, we are realizing that this may no longer be the case and will be an analysis point in the final report. For now, the Nano's better peripheral support is a benefit because it enables additional cameras, which may be needed for the final product to meet the timing requirements.

B. Microphone Algorithm Analysis

We decided to use acoustic location in our design to facilitate the process of locating the speaker without having to entirely rely on the cameras. For this, we used the physics of the traveling of sound waves to determine the direction of the speaker based on the difference in time of arrival of the speaker's sound to each microphone. From the measurements noted in Figure 4 and the speed of sound, c, we can get the time difference between one pair of microphones as:

$$\Delta t = (d/c) \sin(a + \pi/2 - x + \theta) \tag{1}$$

From this, we can solve for x, the direction of the source.

TABLE I. Jetson Nano Vs. Raspberry Pi Spec Comparison

	Specs and Cost	
	Jetson Nano B01	***RaspberryPi Model 4***
Price	\$99	\$55
RAM	4 GB	4 GB
Video	2 MIPI CSI-2 DPHY lanes	1 MIPI CSI-2 DPHY lanes
USB	4 USB 3.0	2 USB 3.0, 2 USB 2.0
GPIO	40 pin	40 pin
Video Decoder	H.264 up to 1080p240	H.264 up to 1080p60
Audio	2 I2S	No I2S
CPU	ARM A57	ARM A72
Motor	4 I2C, 1 PWM	6 I2C, 2 PWM

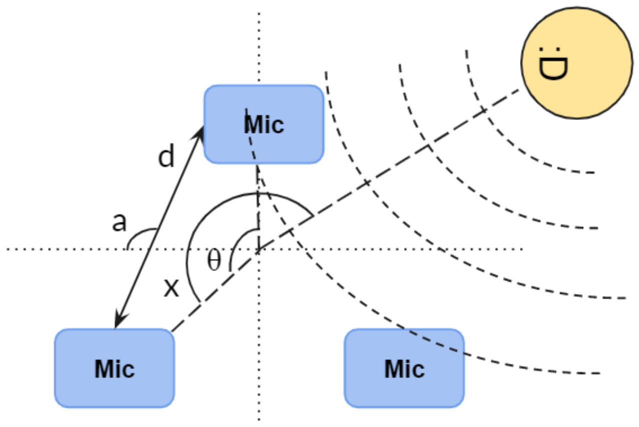


Fig. 4. Two microphone setup

C. *Microphone Two vs. Three vs. Four Comparison*

Our initial plan for the audio was to use four microphones, two connected in stereo to each of the Jetson's I2S pins. However, we soon realized that we would need each individual microphone's audio feed in order to determine the difference in arrival time of the speaker's sound at each of the microphones, and by connecting two microphones in stereo on one I2S input, we would not be able to process their audio feeds separately. We briefly decided simply to switch from using four microphones to just two -- one on each I2S pin. We did not stay with this idea for long, as we realized that we would never know from which side of the microphone array the source would be. The difference in arrival time would be the same for the two speaker locations, as illustrated in Figure 5, since both are equidistant from each microphone; the two-microphone array would not be able to distinguish between them.
From here, we resolved to abandon our I2S microphones and make use of the three remaining I2C pins on the Jetson (the Jetson has four in total, and one is being used for the motors). This unfortunately meant we had to switch to analog microphones, as opposed to our I2S microphones with digital output. In the search for new analog microphones, we had to make some more tradeoff analyses. We narrowed our options down to Adafruit's MAX4466, which has adjustable gain, and MAX9814, which has automatic gain control. The automatic gain control made the 9814 the more appealing choice, but it was also more expensive; we initially figured that we could deal with managing the gain ourselves on the 4466 if it meant minimizing the cost of the iContact. However, upon reading customer reviews of the 4466, we found that the production quality seemed to vary widely, and that it picked up a great deal of noise. We ended up choosing the 9814, which was pricier but also seemed much more reputable and higher quality, based on its reviews. Plus, of course, the automatic gain control was a solid bonus.

D. *Panning Range Comparison*

The vertical panning range was changed from the original plan of three feet to one foot because it became too much of a mechanical challenge. The focus for this project is on the electrical and algorithmic challenges presented.

E. *Motor Selection*

The motor selection was made based on the choice to use the Adafruit motor HAT v2.3. The use of the motor HAT simplifies motor control because the Jetson can easily send controls to the motor HAT via I2C and using the Adafruit MotorKit Library. In addition, Jetson Nano does not have support for PWM (pulse-width modulation), which was a must to get the fine angle selection for the micro servos controlling the camera angle. The motor HAT allows the use of PWM communication with the micro servos and stepper motors. The motors then selected were ones known to be compatible with the motor HAT, as suggested by Adafruit.

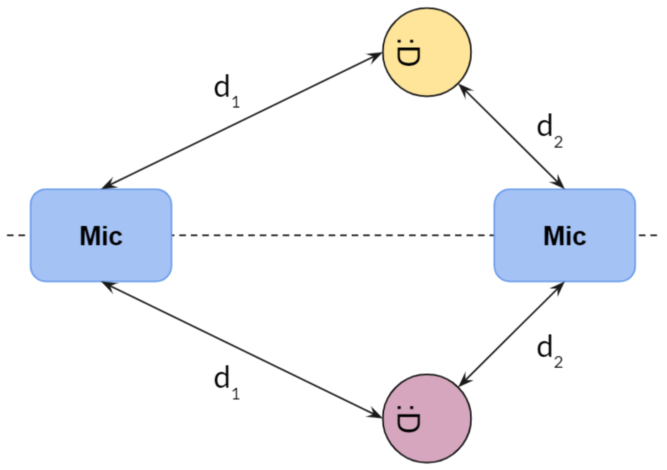


Fig. 5. Two microphone setup

F. *Webcam Setup Selection*

Originally, the Jetson Nano was planned to send video data from one of the cameras to the computer. The plan for this was to edit the Linux kernel of the Jetson Nano before flashing it. Linux has Linux Gadget Drivers, such that when a USB probe from the user's computer arrives, the Linux computer, AKA the Jetson Nano, would present itself as a USB Video Class gadget. For the scope of this class, this seems like it could be a risk point because of the time it would take to try and get this component working. To free up time to focus on speaker detection, iContact will be using a regular webcam for the minimum viable project. This camera will be placed directly next to one of the Raspberry Pi cameras and will be used as the video feed.

G. *Camera Comparison*

For our cameras, we had the choice between USB cameras and MIPI camera modules. USB cameras are very easy to use and are readily available but the biggest concern for them was the greater latency, since the USB protocol includes routing through the CPU before it can be used, whereas a MIPI connection is sent straight to the memory. In addition to this, MIPI cameras were cheaper than a webcam. The final product that we landed on was the Raspberry Pi Camera Module V2 since it was highly rated and known to being compatible with the Jetson Nano and other Jetson environments. Our second design choice for the cameras was the number of cameras that we wanted to use. Part of our design requirements is to minimize the amount of time for speaker transitions and part of that will come from minimizing the time it takes for the motors to adjust the camera. Given that each of the chosen cameras has a 62-degree horizontal field of view, we could have full coverage at all times, but we felt that this was unnecessary, as we would still need to include rotation for the proper recentering. We wanted to choose the number of cameras that would reduce the maximum amount of yaw-axis rotation to a reasonable degree (see Figure 6). For example, in a single camera setup, the max we would need to rotate is 180 degrees to get to the furthest point away from our current field of view. We initially wanted to work with 3 cameras since that would reduce it to 60 degrees as our maximum rotation, but then we ran into the limitations of the Jetson Nano. The Nano could potentially accommodate additional cameras, but that would require us to purchase additional hardware to mux on the MIPI lanes. For this reason, we felt that we would still have a reasonable amount of rotation using only two cameras, and it would not require additional hardware, thus minimizing the total project cost.

V. System Description

Our hardware system is made up of a Jetson Nano B01, two Adafruit Nema-17 stepper motors, two TowerPro SG92R micro servos, one Adafruit v2.3 motor HAT, and two Raspberry Pi V2-8 camera modules.

The Jetson is the brains operating our entire device. It is powered from its microUSB port connected to a USB on the host laptop; this same connection is also used for video data transfer. The Jetson is what controls the motors, which are connected by the motor HAT to one of the Jetson's I2C pins and powered by a 12-volt, 2-amp power supply. The remaining three I2C pins on the Jetson are connected to an Adafruit ADS1115 analog-to-digital converter, which receives its analog input from three Adafruit MAX9814 auto-gain control microphones and passes along the converted digital signal to the Jetson.

Upon booting the Jetson Nano, a script to start the iContact software will begin [7]. For the overall structure of the software, there are three distinct parts that all need to communicate with each other. These three components are the motor controller, the audio processor, and the speaker tracker. These components are all connected to a main program that will keep track of the current state of the software cycle. It will expect new input from the audio processor, give motor instructions to the motor controller and signal to the speaker tracker when to start and expect a result from it as well. These are all going to be concurrently running so part of the challenge will be to ensure we do not have race conditions.

A. Motor Software

The motor implementation is done with an Adafruit motor HAT to control the micro servos and the stepper motors. The motor HAT uses PWM to communicate with the micro servos and stepper motors. The Jetson Nano controls the motors through the motor HAT using I2C. The Adafruit MotorKit Library in Python is used to code the controls. The micro servos are easily controlled by specifying angle to turn to. The stepper motors can only specify the amount of degrees to turn, so iContact will have a predetermined starting position for the stepper motors, and at shutdown will return to this position. This is especially important for the base stepper motor because the software algorithm needs to know where the stepper motor is in relation to the microphones to be able to adjust the base motor so the cameras point at a suspected speaker location. When a new speaker has been detected as talking, and there is currently no

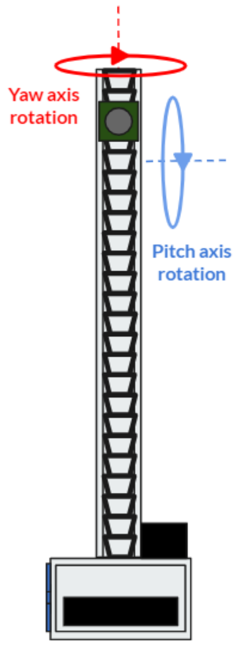


Fig. 6. Camera rotation

speaker in frame, the algorithm in Figure 7 details how the motor control for face centering is done. Upon shutdown of the Jetson, motors are returned to their predetermined starting position.

Motor angle adjustment differs for the micro servos and the stepper motors. The micro servos can be adjusted by specifying an angle within a 180-degree range. The stepper motors are adjusted by 1.8-degrees at a time. A counter is used to keep track of the current angle of the stepper motor, relative to its initial position.

B. Audio Software

Our audio implementation plan is to sample 16 bits at a time from each microphone at 860 samples per second (the sample size and rate are both limited by the ADCs). For each microphone, we partition the sample set, find the peak in amplitude of each partition (which we assume refers to the same sound within the same partition across the microphones), then calculate the time difference between peaks of the same partition across each pair of microphones. Next, we calculate the average time difference across all the microphones, which we use along with the relative distance and angles between each pair of microphones, to determine the angle of the acoustic source relative to the arbitrary axes defined by our microphone arrangement.

C. Computer Vision Software

When called upon, the computer vision software will track for the face of a person. These cameras are connected to the Jetson Nano via two MIPI lanes. The first step is to identify all the people in the current frame. This is done with the use of Haar cascades. Haar cascades are pretrained models for openCV and can detect a pattern within the image it is scanning through. We plan on using a mix of both facial and full body detection to search for figures within each frame. If there is only

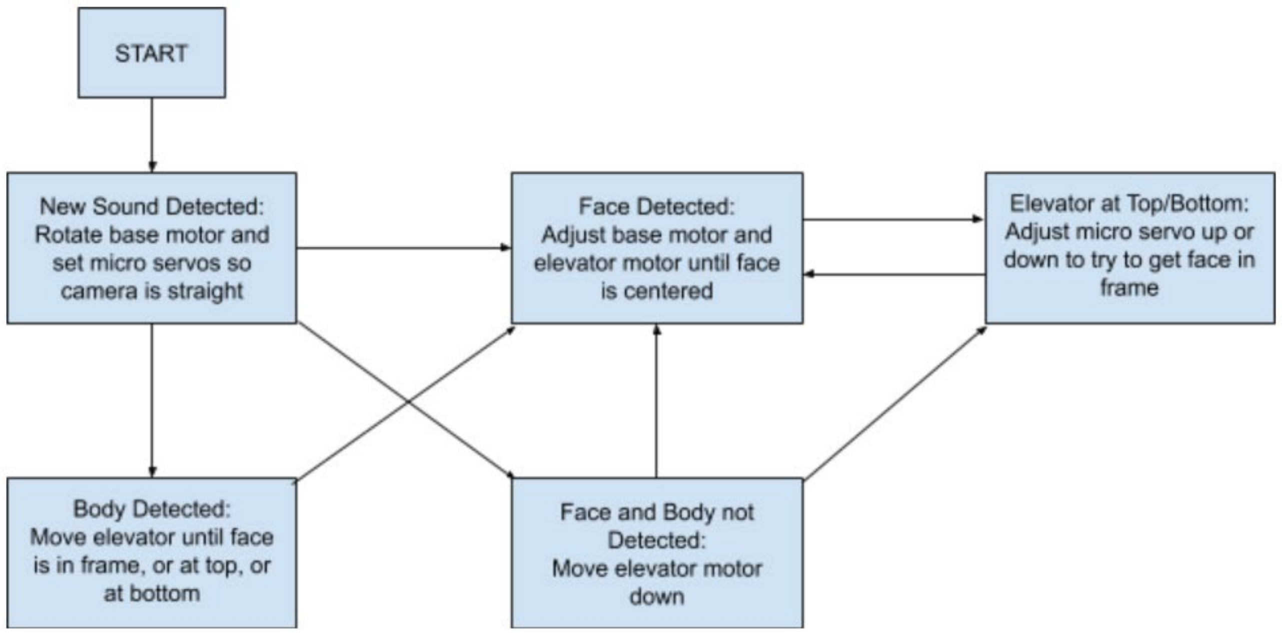

START
New Sound Detected: Rotate base motor and set micro servos so camera is straight
Face Detected: Adjust base motor and elevator motor until face is centered
Elevator at Top/Bottom: Adjust micro servo up or down to try to get face in frame
Body Detected: Move elevator until face is in frame, or at top, or at bottom
Face and Body not Detected: Move elevator motor down

Fig. 7. Motor movement algorithm

one speaker in the frame, we will assume that this is the speaker. We will also verify that this is the true speaker by detecting mouth movement over multiple frames will help us determine if the detected person is speaking. This same algorithm will be even more important for when multiple speakers are detected within a single frame. Here we must apply the algorithm across all the different detected people. One issue with this is that we might not successfully detect certain facial features on each of the frames which can lead to error. When we hit situations like these, we will first try to use the little information we have and if that is still not enough, we will simply report back to the main function that we could not find any speakers in the frame. If we can successfully identify the speaker, the final step is to calculate the degree of movement necessary to center the speaker in the camera frame. Whether or not we are successful in detecting the speaker, we still send the message back to the main program to interpret any changes in the rotation that is necessary.

D. Jetson Nano

For the computation power, iContact utilizes a Jetson Nano B01. It is powered by a micro USB cable connected to the host laptop. This connection will also be used for the video data transfer.

E. Motors

The Jetson controls the motors through an Adafruit motor HAT v2.3. The motors being used are two Adafruit Nema-17 Stepper Motors and two TowerPro SG92R Micro Servos. A 12-volt, 2-amp power supply is used to power the motors. (See Figure 3)

F. Microphones

For the audio aspect of our project, we ultimately decided on using three Adafruit MAX9814 microphones with automatic gain control, all of which would be connected to the Jetson via an Adafruit ADS1115 16-bit analog-to-digital converter to minimize the noise on each signal (see Figure 8).

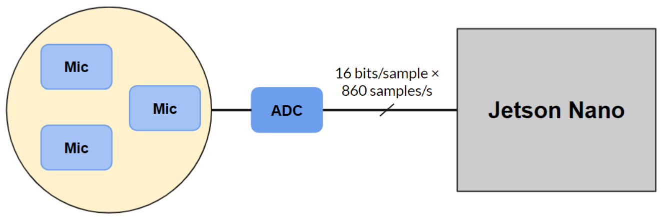


Fig. 8. Microphone hardware

G. Webcam

A Microsoft Lifecam will be attached directly next to one of the RaspberryPi Module v2 cameras. This webcam will connect to the user's computer.

VI. Project Management

A. Schedule

Our project schedule has changed little up to this point (Week 7), as we have managed for the most part to stay on track. The only major change has been the removal of the section of our plan where we intended to implement verbal command functionality, which we realized early on would be too great of a challenge on top of our existing design.

From this point on, our schedule will be significantly ramping up in involvement as we enter the second half of the semester. By the end of next week, we plan to have our individual components finished, such that by Week 9, we can commence integration, which will be a sizable hurdle as we bring our various pieces of the project together while our team is scattered across the country. Directly following integration will be rigorous testing, after which our project will end with the final presentation and report.

B. Team Member Responsibilities

We have divided the work for our project such that Heather's primary task is motor control, Edward's CV, and Anna's audio. As for our secondary tasks, Heather and Edward are working jointly on hardware communication for the video feed (i.e. camera to Jetson to computer), while Anna and Heather are working together on hardware communication for the audio feed (i.e. microphone to Jetson to computer).

C. Budget

Our bill of materials may be referred to in Figure 9. We had to purchase multiple (sets of) items, such as the Jetson Nano, cameras, and microphones, so that each of us could have our own partial implementation of the project, since we are all working together remotely. The blue rows are the items that we planned to purchase from the start; the yellow rows are the items that we did not initially plan to purchase but ended up needing; and the red rows are the items that we purchased but did not use.

D. Risk Management

From the beginning of our project, we did a great deal of planning to minimize the many risks -- particularly with respect to integration -- that would inevitably come with working on a mechanical design while the three of us were all in different locations. For example, knowing integration would be perhaps the most challenging part of our project, we made sure that none of the integration tasks were solo tasks.

Moreover, we spent a great deal of time coming up with what components to use in our design and the potential risks that came with each decision. One of the biggest decisions we made was whether to use a Jetson Nano or a Raspberry Pi, which would be the brains of our entire device. The primary features we were examining while deciding between the two was the peripheral support. Our overarching ideology was "the more peripherals, the better" -- it is, of course, always easier to add components or change protocols with an excess of ports than a lack thereof. For instance, we knew our project would require the use of either one or two cameras, but at this early stage, we

Item	Quantity	For Whom	Final Price (includes shipping and tax)
Jetson Nano	1	Heather	$107.91
Jetson Nano	1	Anna	$104.94
Jetson TX2	1	Edward	Rented
MicroSD Card	1	Heather	Already owned
MicroSD Card	1	Anna	Already owned
Microsoft Lifecam HD-3000 Webcam	1	Heather	Already owned
Adafruit MotorHat v2.3	1	Heather	Already owned
Adafruit TowerPro SG92R	2	Heather	Already owned
Adafruit Nema-17 Stepper Motor	2	Heather	Already owned
Raspberry Pi Camera Module V2-8	2	Heather	$45.40
Raspberry Pi Camera Module V2-8	2	Edward	$45.40
Adafruit I2S MEMS Microphone	1	Heather	$11.00
Adafruit I2S MEMS Microphone	4	Anna	$41.12
USB TTL	1	Heather	$16.62
Adafruit ADS1115 16-bit ADC	1	Anna	$15.59
Adafruit MAX9814 Microphone	3	Anna	$27.30
Capacitor Assortment	1	Anna	4.72
Total Spent			**$420.00**
Budget			**$600.00**
Amount Leftover			**$180.00**

Fig. 9. Bill of Materials

were still unsure of which quantity we would ultimately agree upon. In that respect, the Jetson, having two video lanes, was the safer bet compared to the Raspberry Pi, with its singular video lane. When it came to choosing microphones, we opted for I2S, in part because it is designed to be digital (and would thus have less noise than analog), and because it would not use up any of the USB ports, which we wanted to keep open in case we needed to add any other components to our project down the road. Lastly, in general, we did our best to minimize our spending to leave us enough room in the budget for unexpected purchases to be made in the future.

With regards to audio, when we realized that our original four-microphone plan would not work and we were left with two options, we resolved to go with one of the options (a two-microphone I2S setup) but ordered the components necessary and took time to consider the design of the second option (a three-microphone I2C setup) to prepare it as a fallback for the first option. We did end up needing to abandon the first option and resort to the second, so our prior risk mitigation certainly paid off.

VII. Related Work

Aside from iContact, there are several other similar products that seek to make video calls more personal using smart cameras that dynamically focus on whomever is commanding attention. One such existing solution is the Meeting Owl, a smart video conferencing camera that captures 360-degree video and audio. Its single omnidirectional camera takes in a static, panoramic view of the room from the center of the table, then displays the section of that view where the current speaker is located. One of last semester's CMU ECE Capstone projects, COMOVO, was also a smart video conferencing camera to be placed at the center of the table. Instead of having a panoramic view of the room like the Meeting Owl, this unidirectional camera was motorized and capable of turning to whoever is speaking either automatically or by interpreting physical gestures. Other devices, like Google Meet's video conferencing hardware, Polycom's Poly Studio, and Facebook's portal feature a stationary, unidirectional camera to be placed at the front of a conference room such that it has a full view of everyone present and can zoom in on the current speaker. What sets iContact apart from all these products, which seem limited to one plane of view, are its abilities to raise and lower and to tilt up and down, giving it a vertical range of view that existing solutions cannot achieve.

VIII. Summary

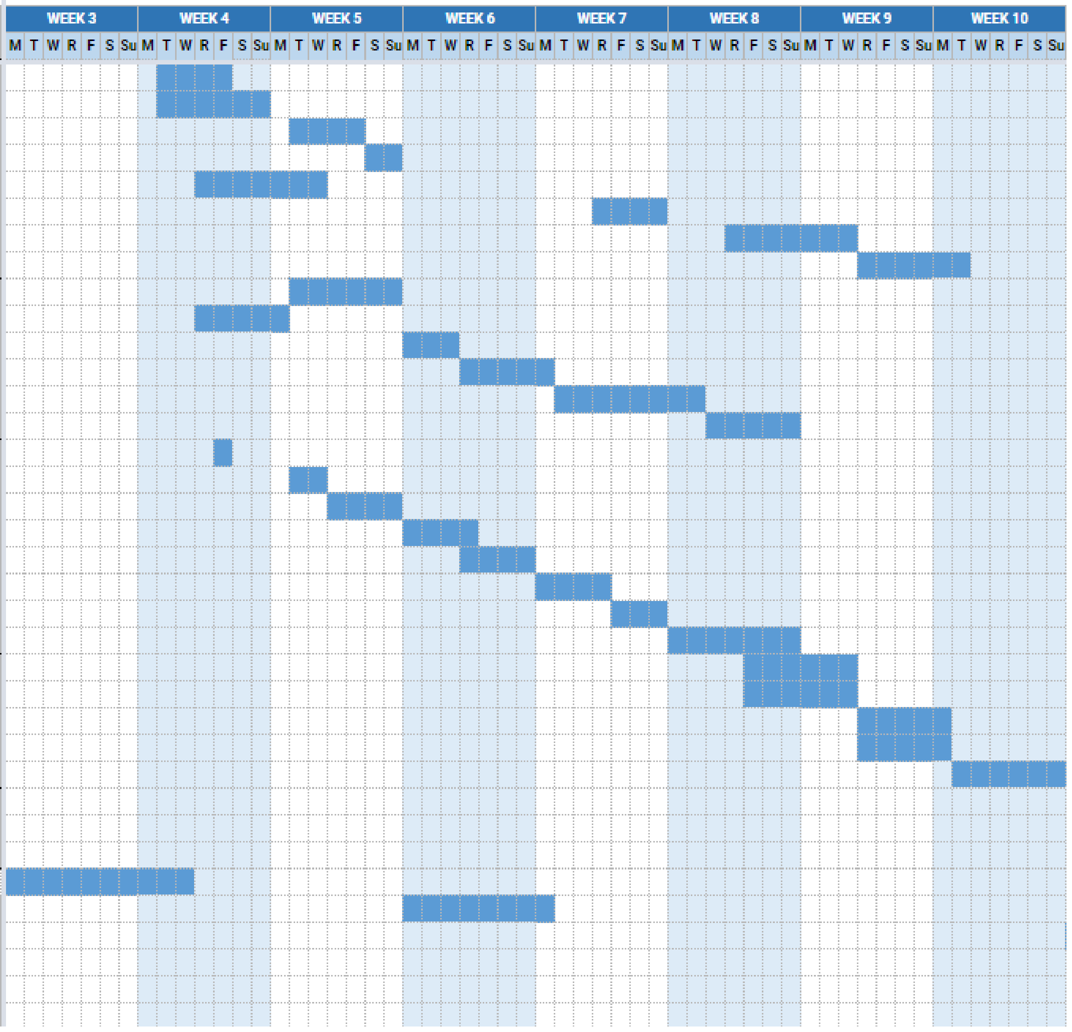
So far, we have learned a lot about teamwork and setting and meeting our own deadlines. We have been following our schedule well, so we are optimistic for the project's continuation throughout the rest of the semester.

References

[1] OpenCV, https://docs.opencv.org/4.1.1/
[2] ScienceDirect, https://www.sciencedirect.com/topics/engineering/interaural-time-difference
[3] Wikipedia, https://en.wikipedia.org/wiki/Acoustic_location
[4] RidgeRun, https://developer.ridgerun.com/wiki/index.php/How_to_use_the_audio_gadget_driver
[5] Microsoft, https://docs.microsoft.com/en-us/windows-hardware/drivers/stream/usb-video-class-driver-overview
[6] RidgeRun, https://www.ridgerun.com/usb-video-class-gadget
[7] Nvidia Developer Forum, https://forums.developer.nvidia.com/t/how-to-auto-run-shell-script-made-by-me-when-tx2-system-is-booted/57991/4

iContact Schedule

Phase	Task	Assigned to	Start Date	End Date	Number of Days
Physical Product	Purchase Hardware Components	All	9/22/20	9/25/20	4
	Review Datasheets for Components	All	9/22/20	9/27/20	6
	Verify multiple motor control	Heather	9/29/20	10/2/20	4
	Test timing for rotations	Heather	10/3/20	10/4/20	2
	Design model for product	All	9/24/20	9/30/20	7
	Assemble first draft of physical model	All	10/15/20	10/18/20	4
	Make alterations to original design after tests	All	10/22/20	10/28/20	7
	SLACK	All	10/29/20	11/3/20	5
Audio	Verify stable multi mic connection	Anna	9/29/20	10/2/20	4
	Research algo to detect speaker's general location	Anna	9/24/20	9/28/20	5
	Build microphone array	Anna	10/3/20	10/6/20	4
	Write/test acoustic location algorithm for 2-mic array	Anna	10/7/20	10/12/20	6
	Assemble/test 4-mic array	Anna	10/13/20	10/20/20	8
	SLACK	All	10/21/20	10/25/20	5
Computer Vision	Meet with CV Professor	Edward	9/25/20	9/25/20	1
	Basic setup and installation for Jetson	All	9/29/20	9/30/20	2
	Create a simple videofeed for a computer	Heather	10/2/20	10/4/20	3
	Verify multi camera communication	Heather	10/5/20	10/8/20	4
	Extract single frames	Edward	10/8/20	10/11/20	4
	Facial and Body detection	Edward	10/12/20	10/15/20	4
	Zooming in on a portion of the frame	Edward	10/16/20	10/18/20	3
	SLACK	All	10/19/20	10/25/20	7
Integration	Integrate the video passthrough	Edward	10/23/20	10/28/20	6
	Integrate audio into the passthrough	Anna	10/23/20	10/28/20	6
	Integrate motor movement with presets	A/H	10/29/20	11/2/20	4
	Integrate motor movement with CV component	E/H	10/29/20	11/2/20	4
	SLACK	All	11/3/20	11/9/20	7
Testing	Test Latency of the system	All	11/12/20	11/17/20	6
	Optimize Latency	All	11/17/20	11/22/20	6
	SLACK	All	11/23/20	11/29/20	7
Course Logistics	Project Proposal	All	9/14/20	9/21/20	8
	Design Presentation	All	10/12/20	10/14/20	3
	Demo 1	All	11/9/20	11/11/20	3
	Demo 2	All	11/30/20	12/2/20	3
	Final Presentation	All	12/3/20	12/9/20	7
	Final Report	All	12/8/20	12/13/20	6



Phase	Task	Assigned to	Start Date	End Date	Number of Days	WEEK 11							WEEK 12							WEEK 13							WEEK 14							WEEK 15						
						M	T	W	R	F	S	Su	M	T	W	R	F	S	Su	M	T	W	R	F	S	Su	M	T	W	R	F	S	Su	M	T	W	R	F	S	Su
Integration	Integrate the video passthrough	Edward	10/23/20	10/28/20	6																																			
	Integrate audio into the passthrough	Anna	10/23/20	10/28/20	6																																			
	Integrate motor movement with presets	A/H	10/29/20	11/2/20	4																																			
	Integrate motor movement with CV component	E/H	10/29/20	11/2/20	4																																			
	SLACK	All	11/3/20	11/9/20	7																																			
Testing	Test Latency of the system	All	11/12/20	11/17/20	6				🟦	🟦	🟦	🟦	🟦	🟦																										
	Optimize Latency	All	11/17/20	11/22/20	6									🟦	🟦	🟦	🟦	🟦	🟦																					
	SLACK	All	11/23/20	11/29/20	7															🟦	🟦	🟦	🟦	🟦	🟦	🟦														
Course Logistics	Project Proposal	All	9/14/20	9/21/20	8																																			
	Design Presentation	All	10/12/20	10/14/20	3																																			
	Demo 1	All	11/9/20	11/11/20	3	🟦	🟦	🟦																																
	Demo 2	All	11/30/20	12/2/20	3																						🟦	🟦	🟦											
	Final Presentation	All	12/3/20	12/9/20	7																									🟦	🟦	🟦	🟦	🟦	🟦	🟦				
	Final Report	All	12/8/20	12/13/20	6																														🟦	🟦	🟦	🟦	🟦	🟦