ArticuLight – A gesture controlled articulating lighting system

Scott Bell, Keng Chu, Benjamin Simms, and Michael Taub

Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32832, U.S.A

***Abstract* — This paper will be presenting both circuitry and mechanical design that will show people just how much more advance work benches can become. Today, current work benches lack light positioning. For example, when working on a project on a work bench, shadows casted from the non-moveable light can cause obstacles as it disorients your view of where you are trying to work on that specific portion of the project. This paper's primary focus is on the design. It was our purpose to create an adjustable light to help overcome shadows as well as positioning lighting to whatever the user prefers. Furthermore, we are hoping to make sure that the light movement is quick and accurate according to the user's gesture. While there will always be some type of shadowing from the light, we are hoping that the speed and accuracy are prioritized the most in our design. The data presented throughout this paper will show our focus and reasoning for the design we chose.**

***Index Terms* — Camera Serial Interface (CSI), Inter-integrated Circuit (I2C), OpenCV, Serial Peripheral Interface (SPI), and Universal asynchronous receiver-transmitter (UART).**

I. Introduction of ArticuLight

The objective of the ArticuLight device is to implement a gesture controlled articulating light fixture through the use of computer vision. A camera connected to an embedded computer will analyze the gestures of the user and perform the relative action associated with the gesture. The planned gesture implementations are switching the light on or off, resetting the arm position, moving the light to a desired location through the articulating arm, setting a home position for the arm, moving the arm to the stored home position, setting the dimness of the light, and powering off. ArticuLight's technical specifications are as follows: the system will be powered by 120V AC at 60 Hz. This will power everything in the system including the embedded computer and light bulb. The arm will articulate with an accuracy of less than or equal to one cm in distance from the gestured location that is instructed to the system. The system will have a software to hardware interface of less than three seconds per gesture and the computation of issuing an instruction from a gesture will be two seconds. The entire system will weigh in at less than or equal to seven pounds and will contain an arm that is twenty-four inches in length. Motors connected to a motor controller will be the driving force that implements the articulation of the arm, and the arm will articulate with up to 90 degrees of movement in both the yaw and pitch orientations. The physical footprint of the system will be designed so that it will be mountable while also being capable of sitting on a stand.

II. Overview of PCB Design

For our PCB board we wanted it to be able to power it up from the wall outlet. It made sense since our light was going to be powered by 120VAC. We needed to design our circuit where we could utilize both AC and DC voltage. All the electronics we will be using will be running on DC voltage except for the light. There were two main components that we needed to implement into our circuit design. The first is a step-down transformer to reduce the full 120VAC to about 12VAC. Once we got our voltage reduced, we needed a way to turn the AC voltage to DC. The easiest way to turn AC signal to DC signal was with a full bridge rectifier. Using 4 diodes connected into a box shape going one direction we built a simple ac to dc converter. Since the output dc voltage is not a pure dc voltage as it is still pulsating and not a continuous signal, we added capacitors to smooth the DC voltage. One of the requirements was that our circuit board needed a microcontroller. We opted for an ATmega328P. The jetson nano and Atmega328P both required 5V input voltage. The ATmega328P TX pin outputs 5V, whereas the JETSON could only receive 3.3V otherwise we would run the risk of burning out the pin. So, a voltage divider was implemented on the PCB for the serial TX pin.

During our initial design we went from using 4 diodes to a single-phase rectifier, reducing the number of components in the circuit board. Due to the significant difference in current required for the Jetson and motors and the ATmega328P, we split the power into two voltage regulators. One for the Jetson and motors, and one for the ATmega328P. The voltage regulator could only handle an output 5v with a maximum current of 1.5A, so a PNP power transistor was added to enable the amperage to be increased to accommodate a current draw from the Jetson and servo motors.

FIGURE 1
PCB BOARD

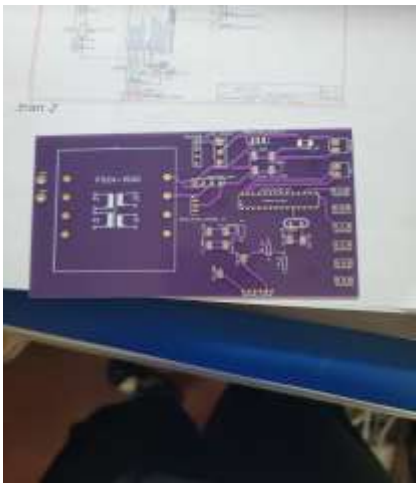
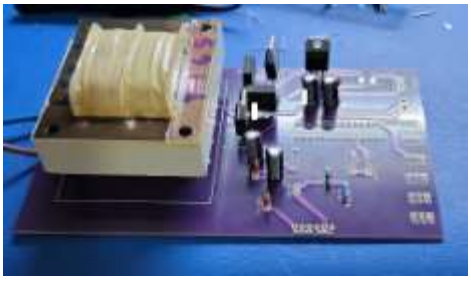


FIGURE 2
PCB BOARD



Our circuit board design was created on a software program called easyEDA. Design was sent out and we received our board in about a week. We started to assemble our board and quickly ran into our first issue. We were not getting a signal to from the ATMEGA to a single servo motor. We redesigned our circuit board on the breadboard and started to test things again.

One of the issues we encountered was the need to decouple the Vcc, AVcc, and ARef pins which was done by adding decoupling capacitors from these pins to ground. Once the capacitors were added, the issue of PWM output was still present. We re-examined our circuit design and found some flaws in the design.

The AMEGA AVcc was not connected to power and traces for the ground pins were not in the PCB layout that was sent to JLCPBC for printing. We ran two wires for power and ground and retested the board, but still had the same issue. After more circuit analysis, we concluded the pull up resistor on the reset pin may be the cause. We removed the 10k pull-up resistor from the circuit board and we finally had a PCB that functioned as intended.

III. MECHANICAL DESIGN

The mechanical design of the ArticuLight can be broken down into 5 major components. The segments/wire guides, the camera mount, the servo mount and stands, pulley design, and the case or housing.

A. Segments/Wire guide

The arm is a continuum arm design. This uses a spring, or some equivalent tubing that would return to a center resting state. The center spring has multiple segments with the wires fastened at the top, by the camera. This shifts the pivot point from the base of the arm to the top. Each segment is essentially the base of the interlocking connectors which have 4 near the edges to act as the wire guides, with a small plastic spring added to prevent the wire from chewing through the plastic allows the wire to flex around the hole better and a center hole the diameter of the spring which provides a very tight fit.

B. Camera Mount

The camera mount was designed to accommodate the leg stands of a raspberry pi camera stand. A segment was used as the template with two legs extended from the top. The center hole was widened to accommodate the light socket.

C. Servo Mount and Stands

The servo mounts were designed as an L bracket with bolt holes that lined up with the servo mounting holes. The original design was not strong enough to account for the amount of torque placed on them by the actuation of the arm, so we made the mounts nearly 3 times thicker. The stand was made from L brackets for corner joints and sheet copper. The bracket provides stability for the servo during actuation, not allowing it to flex and the copper extends behind the motor to both give it more rigidity and act as a heat sync.

D. Pulley Design and Cable Selection

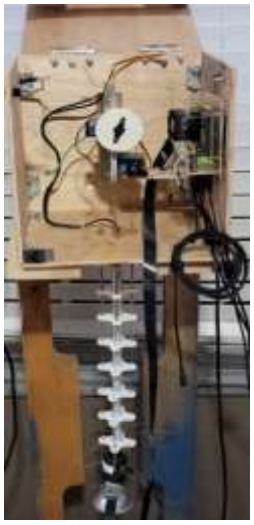
The pulley design was made to allow the center gear through the center to allow us to connect the 2 leg servo actuation arm. A channel was built into the print so that the servo actuation arm would fit inside of the pulley. Along with the mounting holes for bolts to secure it in place, the inset actuation arm gives allows us to distribute the force

across the pulley instead of the mounting bolts. A set screw hoes was added to fix the wire to pulley. Deep sea fishing line was selected for the guide wire because it is a coated steel wire that is very strong and very flexible. The coating increases the longevity of the wire guides.

E. Case/Housing

The case was originally built out of plexiglass with 3D printed corner connectors. The connectors allowed us to assemble the case without the use of hardware such as bolts and screws. Upon testing, we noticed that the plexiglass would flex when the arm was moving. This was skewing our testing results since the amount of flexing was not constant. This also would likely cause the case to break over time. The case was redesigned to be sturdier, using wood. This added the rigidity and strength needed to prevent excess movement from the servos.

FIGURE 3
FRAME/ARM DESIGN



IV. Software (Nano Focus)

To implement computer vision and gesture detection, python scripts on the NVIDIA Jetson Nano developer kit were written. The articulight software on the Jetson Nano is based on the software for trt_pose_hand. This (trt_pose_hand) library uses a pretrained model based on Resnet-18 that intakes a 224x224 prepared image and returns locations in the image where it detects the locations on the hand that map to parts of the hand, i.e., three digits for each finger and certain locations of the palm. With the returned orientations of the hand, we train a support vector machine using svm.py for multi-class classification on the gestures we want to train. This script will save the SVM as a .sav model which is loaded in articulight.py for classification. To train the model, we run image_collection.py, which saves images and their labels (which gesture it corresponds to) to train the SVM model for gesture recognition.

A. Image_Collection.py

Written for gathering a dataset for the program svm.py, it loads a GUI using the PySimpleGUI library top navigate saving images to the Jetson Nano and which gesture the image corresponds to. When the script starts, it prompts the user for the name of the dataset and tells the user the path that it is going to save the dataset to. Once the name of the dataset is inputted, the number of gestures is then prompted. It loops through the number of gestures and gathers the names of the gestures from the user. Then, the GUI is opened. The GUI has buttons for the following: Save Image, Save 600 Images, Save 100 Images, Next Gesture, Previous Gesture, Toggle Train/Test, and Exit. When any of the save image buttons are pressed, an image is saved to the dataset directory in their own directories dependent on the gesture it represents. The next gesture and previous gesture buttons are for selecting the gestures the user wishes to save images to. The toggle train/test button allows the user to save images to a training and testing dataset. Once the exit button is pressed, the program cleans the directories up, renames them to iterative names (1, 2, 3, etc) and creates a json file for the labels of the respective gesture for each image. Then, this directory is used in svm.py to train a support vector machine.

B. Svm.Py

A support vector machine is used in this project to classify the data that the hand model returns when a hand is passed into it. Svm.py takes the dataset images and trains the returned values of the hand model supplied by NVIDIA. After the svm model is trained, a json file is created with the respective labels of the hand gestures, as an svm outputs a number representing the classification. The svm model is saved to the project directory to be loaded in another script later through pickle, a file loading python library. Once the svm model is trained, it is loaded and the camera is booted up to test the performance of the model. An issue that we faced in this software is that the svm model seemed to randomly assign gesture classifications and the programming of the json file would not result in expected behavior. To overcome this, we named the gestures that the json file

mapped to the svm outputs as ‘one’, ‘two’, ‘three’, etc and mapped them manually through trial and error. If more time was available, this would be investigated further.

C. ArticuLight.py

ArticuLight.py uses the trained SVM model from svm.py to achieve real-time gesture detection. It passes each frame into the hand model and the outputs of the hand model into the SVM model, and performs actions based on these gestures. The gestures and their actions are outlined in the gesture section. An issue that we came across developing ArticuLight.py was that the SVM model would identify a gesture that was not intentionally supplied by the user. To overcome this, we added a gesture buffer that keeps track of the previous gestures. For a gesture’s behavior to process, it must be held for one second. Because the program process frames at a constant frequency, in this case 8 frames per second, we can know that if the buffer has the same gesture 8 times in a row, the gesture was held for one second in real time. Therefore, instead of taking the gesture and performing an action based on the gesture, we check if every gesture in the buffer is the same and compare that gesture with the expected behavior.

V. Software (ATmega328P Focus)

The ATmega328P was selected for its ability to communicate over both I2C and Serial with the Jetson Nano and for the 6 PWM digital pins. This microcontroller utilizes the Arduino IDE and libraries which made programming the chip straightforward. The program initializes which pins will be connected to the servo motors. In the setup, the servo motors are set to their resting position and start serial communication with a 9600 baud rate. In the main loop, once data is available in the serial buffer, the program saves the first byte which is used to determine what actions the servos will take. If the first byte is an ‘x’, then the program parses and captures the next integer for both x and y which is then compared to the center of the frame to determine how many steps will be needed for each servo to center the pointer landmark to a defined window in the center of the frame. We added logic to ensure that neither servo angle variable can go below 0 or above 180 degrees. We attempted to implement this on the Jetson Nano and have the ATMEGA strictly move the motors, however, this caused more of a delay in computation which was reflected in the servo actuation being slower. This had the further effect of flooding the Serial buffer, which would be parsed by the loop causing the arm to continue moving even after the hand itself has stopped. If the first byte is a ‘w’, then the program parses the integers for the pixel coordinates of the thumb landmark and using those values, calls the map function to map the frame size to a set range centered on 90 degrees. This allowed us to tune the size of the angle step based on the frame size which in turn moves the servo to that specified position which is also saved as the x and y angle if the follow me function runs after this. Lastly, if the first byte is an ‘r’, then the servos are reset to 90 degrees with those angle values stored as well. Once these characters are checked, the Serial buffer flushed, which has the ATMEGA run through anything else on the Serial buffer before continuing with the loop. This has the side effect of slowing down the program slightly but that was not noticeable.

VI. GESTURE CONTROL

To alter the states of the system, gesture control is implemented through computer vision. The Jetson Nano will recognize a gesture and issue commands to the ATmega328P microcontroller on a gesture basis. The following sections outline the implemented gestures and their respective functions to altering the state of the system. To prevent the gestures from issuing commands accidentally, each gesture must be held for one second before activating their respective behavior.

A. Moving Arm to Desired Position

To move the arm and light to a desired position one time, a thumbs-up gesture is supplied to the camera. The top portion of the thumb in this gesture supplies the reference to which the system will align to. After the gesture is recognized, the Jetson Nano will submit the location of the thumb point in the frame, to which the ATmega328P will map this position to a servo angle that represents its location. After the commands of the gesture are complete, the arm will have the camera oriented so that the camera has centered the thumb and therefore moved the light to that location.

B. Setting the arm to follow a hand

A gesture is implemented that activates the following logic of the system. To activate the system to follow a hand, the gesture ‘Rock On’ must be supplied for one second. This gesture is one which the pinky, index finger, and thumb are extended, and the other digits are closed in the palm. Once the gesture is supplied and the system is following a hand, the gesture should be broken. To prevent the system from activating and immediately deactivating, the ’Rock On’ gesture must be broken before it can be deactivated. Once activated, no gesture must be present for

the system to follow the hand, only a hand must be present. The software grabs the location of the index finger on the hand and uses it as a reference to supply information to the ATmega328P to center the index finger to the center of the camera, resulting in a following behavior.

D. Resetting the arm's position

The last gesture that was implemented is the reset gesture. After the gesture activated servos move the arm while the device is in use, the reset gesture will instruct the servos to set their positions to neutral. This corresponds to a servo angle of 90 degrees. To reset the servos, a gesture known as the 'peace sign' must be held for one second and the Jetson Nano will instruct the ATmega328P to set the positions of the servos to 90 degrees.

VII. Table of Computation Time Tests

Test #	Computation Time (in seconds)
1	.75
2	.81
3	.89
4	.73
5	.89
6	.81
7	1.01
8	.85
9	.82
10	.84

From the ten different tests that were done, we got an average of 0.84 seconds. This is well below our 2.0 second goal that we wanted to achieve for the computation time.

VIII. Serial Communication

As the ArticuLight implements both an embedded computer and a microcontroller, communication between these devices is imperative. The Jetson Nano can implement the GPIO, I2C, I2S, SPI, and UART communication protocols. We decided that the UART protocol using the Serial libraries on each respective device would be the best way for the Jetson Nano to communicate with the ATmega328P microcontroller. This is because the libraries for establishing connections are easy to use and allows testing of the communication to be straight-forward.

IX. Motor Selection and Control

A. Servo Selection

The motors we had to select from were, DC brushed, DC brushless, Stepper, and Servo motors. From our research, we determined that best choice would be use a servo motor because they provide higher accuracy and precision in an angular or linear position since the sensor inside the servo provides position feedback. They have a constant torque across its speed range due to the gears inside the servo motor and are quicker than a stepper motor. Most servo motors have a limited rotation anywhere between 180 to 270 degrees. One major consideration is that programming a servo motor is more complex than that of a stepper motor because of our articulation time specification. They also require tuning and positional feedback and has a higher cost than that of the other motors. When it comes to repeatability, accuracy and speed, servo motors have the advantage and are the best choice for this project. Due to the length and weight of the arm, servo motors with high enough torque were needed.

B. Servo Control

To control the servo motors in our design, requires the use of a pulse width modulation (PWM) signal which is used to determine the output position of the motor. The Jetson Nano was used for the computer vision and machine learning algorithms and an ATmega328P was selected to act as a servo controller. The ATmega328P was selected because it uses the Arduino bootloader which contains libraries to control servo motors. Servo control will be accomplished with the Jetson sends the X and Y

coordinates over UART to the ATmega328P which will then output the required PWM signal to the motors. The Jetson nano by-itself has the capability to produce clean PWM signals that are needed to accurately control the servo motors for speed and accuracy, as those were one of our key engineering specifications. The parameters of the pulses that are sent to the motor define the position of the rotation, which will vary depending on the constraints of the motor itself. For are servo control design we created a code that programs the servo motor to have a degree radius of 180 degrees before it stops. So, for the loop code, all it is doing is writing the position to the servo from 0 to 180 degrees with a 10-millisecond delay. Another factor we have to take in effect with servo control is stability. Stability is important because servo motors operate with continuous current to reach a specified position, velocity, or torque.

X. Table of Articulation time tests

Test #	X – Axis Movement	Y – Axis Movement	Instantaneous Movement
1	1.731	1.823	0.033
2	1.845	1.815	0.04
3	1.735	1.808	0.04
4	1.844	1.817	0.033
5	1.773	1.793	0.033
6	1.809	1.842	0.037
7	1.744	1.779	0.031
8	1.826	1.812	0.038
9	1.772	1.808	0.041
10	1.812	1.806	0.32
Std.Dev	0.04385	0.01681	0.08980

XI. Electrical System

Looking at the system overall, the hardware will have a power regulator that will be designed to take 120 volts AC input from an outlet and output 12 volts DC, although when tested gave out 17 volts DC after flowing through the three-phase transformer as outlined above in section II. Due to the Jetson Nano using a 5 volt DC input, a node will be made to have 12 volts going to the motor controller relay and another to a DC to DC converter to step-down the voltage for the Jetson. Fuses will be used ag various stages to prevent over voltage in order to protect the circuit and the components from power surges, short circuits, or accidental overdraw of power. The Jetson will connect to a motor controller and microcontroller to send an enable signal to the component as needed. The signal sent to the motor controller will include the PWM that will activate the motors, gears, and pulleys to move the light to the appropriate position. A signal to the light system will be connected to a potentiometer circuit to regulate the amount of AC voltage going to the light, enabling us to dim or brighten the light as desired.

XIII. Administrative

In this section, you will be reading about the progression of the ArticuLight over the course of the two-semesters that we were tasked to complete this project. You will see three subsections. The first one will be a figure of our project milestones, which will show our development of our project in both senior design I and senior design II. It will also show the date when that specific task was started and the deadline that we gave ourselves to complete it. The second section will show the current progression of where our project is currently at and the third section will show our budget that we spent on creating and producing our design.

A. Milestones

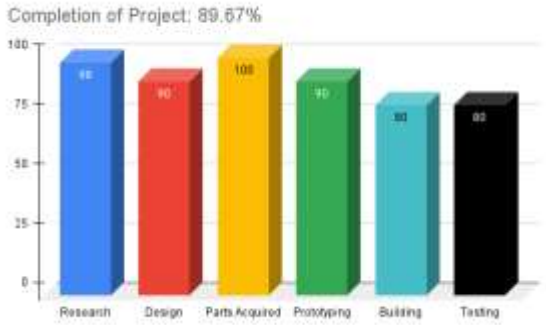
The milestone start and end dates were set based on the corresponding tasks needed and with accordance with each team member's schedule and responsibilities. Most of the milestones that are in the first half of the implementation are based on research and familiarization with system components, while functionality will be the focus of the second half during senior design II. Moreover, these milestones were achieved by multiple group calls, meetings in person, and scheduled help sessions with the professors.

Table 3
Milestone of Project Progression

#	Task	Start date	End date	Status	Responsible
Senior Design I					
1	**Ideas**	1/11/2022	1/14/2022	Completed	Group 9
2	**Project Selection & Role Assignments**	1/11/2022	2/4/2022	Completed	Group 9
	Project Report				
3	Initial Document - Divide and Conquer	1/18/2022	2/4/2022	Completed	Group 9
4	Table of Contents	2/4/2022	2/18/2022	Completed	Group 9
5	First draft	2/4/2022	3/29/2022	Completed	Group 9
6	Final Document	2/4/2022	4/26/2022	Completed	Group 9
	Research, Documentation, and Design				
7	Power Supply	2/4/2022	2/11/2022	Completed	Keng/Michae
8	Robot Vision	2/4/2022	2/18/2022	Completed	Ben/Scott
9	Schematic	2/11/2022	2/25/2022	Completed	Keng/Michae
10	Microcontroller	2/21/2022	2/25/2022	Completed	Ben/Scott
11	Reset Button	2/25/2022	3/4/2022	Completed	Group 9
12	PCB Layout	3/4/2022	3/11/2022	Completed	Keng/Michae
13	**Order & Test Parts**	3/15/2022	6/1/2022	Completed	Group 9
Senior Design II					
14	Build Prototype	6/6/2022	6/20/2022		Group 9
15	Test & Redesign	6/21/2022	7/15/2022		Group 9
16	Finalize Prototype	7/22/2022	7/24/2022		Group 9
17	Group presentation to friends/family	7/25/2022	7/25/2022		Group 9
18	Final Report	7/20/2022	7/27/2022		Group 9
19	Presentation to judges	7/27/2022	7/27/2022		Group 9

B. Project Progression

TABLE 4
PROJECT PROGRESSION



C. Budget

The total estimated expected budget is outlined in Table #. This project is self-financed by the group members. After taking time to research the components needed for our design, we decided to go with the products that offered the benefits we needed while keeping the overall project cost had a fair price.

TABLE 5
COST OF PROJECT

Part	**Quantity**	**Cost**	**Producer**
Jetson Nano	1	$99	NVIDIA
CanaKit Raspberry Pi 3 B+ Power Supply	1	$9.95	Amazon
PNY128GB Micro SD Card	1	$13.99	Amazon
Raspbery Pi Cam v2	1	$26.57	Amazon
D956WP Servo Motor	2	$109.04 each	Amazon
Step down transformer	1	$20	
Metal Wire Loom	2 feet	$19.95	Amazon
E-26 Light Socket	1	$8.99 for 5-pack	Amazon
LED bulb	1	$2.50	Amazon
Frame/case material	1	$15	N/A
Total		$280.95	

XIV. CONCLUSION

In summary, this two-semester long project is a valuable experience for the group by teaching us the general skills needed in the field, such as, how to work in group, how to conduct professional meetings/presentations and how to professionally write a technical report.

In the group meetings that were hosted, contributing ideas to the group, discussing various constraints with engineers, and going to the lab to investigate has given us the necessary experience that can never be obtained from our school education. We also observed that the main concepts or "little pieces" we learned from our engineering classes have become a big picture in reality. For example, in the electronics study, we learned about rectifiers, which is important as it allowed us to control the current flow in our design because most of our components used only allowed a max current flow of 5 Amps or less.

Furthermore, while progressing through our project design, we learned why it is important for engineers of different disciplines to work together. For example, for the gesture recognition it was super useful having a computer engineer on our team, which allowed software implementation to be more successful.

ACKNOWLEDGEMENT

During this project, the group gratefully appreciates the help of the following mentors and consultants for spending their free time with the group, guiding along, and providing valuable information and feedback to help the group succeed in the creation of the project. We kindly thank for the help of the people: Samuel Richie and Lei Wei. We would also like to give a special shoutout to our group member Scott Bell, for allowing us to use his at home lab to build, test, and prototype multiple components at his at home lab.

REFERENCES

[1] NVIDIA. (2022). Hand Pose Estimation And Classification. [online] Available at: https://github.com/NVIDIA-AI-IOT/trt_pose_hand [Accessed 26 Jul. 2022].

[2] Dethe, H. (2019). Face Tracking OpenCV, Python, & Arduino. [online] Learn Robotics. Available at: https://www.learnrobotics.org/blog/face-tracking-opencv/#:~:text=First%2C%20open%20CMD%20and%20type%20the%20following%20codes%3A.

[3] Arduino Project Hub. (n.d.). Face Tracker Using OpenCV and Arduino. [online] Available at: https://create.arduino.cc/projecthub/shubhamsantosh99/face-tracker-using-opencv-and-arduino-55412e.

Scott Bell is currently a Technical Sergent in Air National Guard with 18 years of military experience. He has served 12 years on Active Duty as a Bioenvironmental Engineering Craftsman, performing industrial hygiene and environmental compliance evaluations, radiological surveys, and chemical, biological, radiological, and explosive (CBRNE) emergency response and 6 years as a Client Systems Craftsman establishing tactical communications kit assembly and operation, network connection, network access, encryption, maintenance, and disassembly for Joint operations and hurricane response for the state of Florida. He will be receiving his BSEE from UCF, as well as a minor in Robotics and Intelligent Systems. His current interest lies in Hardware Description Languages (HDL) programming for Field Programmable Gate Arrays (FPGA), Robotics, or Control Systems Engineering. He has recently accepted a position with Aeornix Inc. as a FPGA Hardware Engineer with the aspiration to expand his knowledge to utilize FPGA technology for machine learning and to design digital control systems for robotic systems.

Keng Chu, is currently pursuing a Bachelor of Science degree in Electrical Engineering at the University of Central Florida. He recently accepted and started working at Lockheed Martin as a System Integration and Test Engineer at Kennedy Space Center. He is currently on the Artimis Orion program supporting the launch abort system (LAS) on the Orion crew module.

Benjamin Simms, a senior student of the computer engineering department at University of Central Florida. He will be receiving his BSCpE from the University of Central Florida. After graduation he is planning on joining the workforce in the field of software engineering.

Michael Taub is receiving his BSEE from University of Central Florida. He is presently working as a Customer Service Team Leader at Publix Supermarkets. He will be starting classes towards his PMBA (Professional Masters in Business Administration) in the fall of 2022. His current research interests are in business engineering, economics, and financing as he hopes he can be a manager at an engineering firm to help budget and manage projects more efficiently.