Ion: An Interactive Robot Assistant

Dennis Gebken, David Jaffie, and Mohammad Rizeq

Dept. Of Electrical & Computer Engineering, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Ion is an interactive, voice recognition-based robot assistant that can help a user complete tasks in a more involved, fun manner. Through voice recognition, Ion can understand certain tasks that the user can give such as playing music or searching google. Through face recognition and a servo, Ion can interact with the user, recognizing and following their face as well as reacting to the conversation. This physical, interactive embodiment of a personal assistant sets it apart from competitors who are essentially only a smart speaker. This concept can be applied to help counter boredom and monotony when sitting at the desk. This document summarizes the design and implementation of the device.**

***Index Terms* — Face detection, face recognition, personal assistant, robotics, signal processing, voice recognition**

I. Introduction

Ion is a device that builds on smart speakers and personal assistant combinations that are taking the market by storm. Devices such as Amazon Alexa and Google Home are becoming very popular, helping to link people to the internet through voice. These devices are missing one crucial characteristic that the group built upon – physical interaction and portrayal of emotion. It's hard to interact with a device when it is essentially just a smart speaker and just responds with a voice. With Ion, there is a face that comes with the product that helps the user feel closer and more-personal with the device while also getting emotional and physical feedback through visuals on the LCD display and through motion by the servo.

The initial design for this project was influenced by Peeqo, an animated, interactive robot much like Ion. Peeqo was developed by Abishek Singh, an NYU grad student, and responds to the user both in GIFs and by physically twisting and turning. This cool, new concept for a personal assistant prompted the team to design their own.

II. Relevant Technologies

A. Face Detection and Recognition

Computer vision (CV) is one of Ion's key components. The main functionalities of computer vision are to detect faces, provide feedback to the MCU, and to recognize the user. It helps Ion to record images and video as well as helping Ion track the user's face. This capability of Ion to track faces is a key feature that separates it from other personal assistant smart speakers. The CV software for this project is implemented using OpenCV.

B. Voice Recognition

Voice recognition technologies are starting to be included in a lot more products with the emergence of hands-free devices, virtual assistants, and home automation. Voice recognition is essentially transforming analog sound waves into digital commands that the software can recognize to carry out different tasks. Ion relies on the microphone to take in the sound waves which are then translated into a digital form which can be understood by speech recognition software, Google Speech API and Google Assistant SDK in this case. A standard approach to voice recognition is to listen for a 'wake-up word' in a sentence and then guess the sentence or command with a certain amount of confidence. Higher-level systems utilize artificial intelligence and machine learning to better guess the command and to raise the confidence that it is correct. With this sentence, the software can then decide which command to start or which task to complete. Ion supports almost any command that the Google Assistant supports – this includes fetching the weather, getting directions, and playing YouTube audio.

C. Audio Signal Processing

Audio signal processing is used for the voice recognition part of the project. The audio signal is first received by the microphone, which is then converted to an analog electrical signal. After that, the microphone is filtered and amplified to acquire a much clearer signal that goes through an analog-to-digital converter so that the signal can be sent to the main CPU. Once the CPU processes the digital signal, it sends out that signal through a digital-to-analog converter so that the speaker can finally convert the analog signal to an audio signal for the user to understand the device's response. The process is further illustrated in figure 1 below.

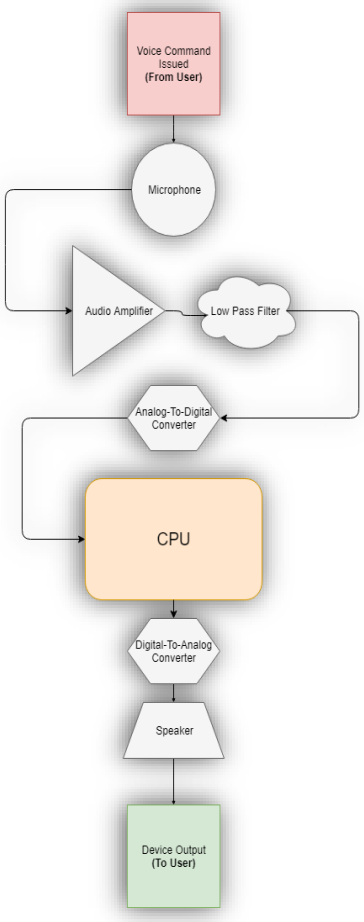


Fig. 1. The audio signal processing part of the design is shown from picking up the audio signal from the user to responding back to the user with another audio signal.

III. Strategic Components

A. Microcontroller

The Raspberry Pi Compute Module 3 Lite is a version of the Raspberry Pi that is intended for more industrialized applications. The module does not include any peripherals found on the regular SBC (Wi-fi, USB, HDMI, etc.). Instead, it is a barebones component that can be connected to a custom PCB via a SODIMM socket. The Lite version was chosen over the regular version because of its use of SD cards as main storage as opposed to flash. Another main argument for choosing the Raspberry Pi was its quad core processor which is needed for the image processing algorithms explained above.

B. Microphone

The microphone used for the device's voice recognition is connected through a USB port. The microphone chosen for the project, manufactured by Sienoc, is simple to use, and does not take up too much physical space.

Fig. 2. Sienoc USB microphone used as a device input for voice recognition. Image credits to www.amazon.com.

C. Speakers

The speakers are mainly responsible for device playback and response. The speakers used are also connected through a USB port. The USHONK USB mini speakers were found the most compatible for the project based on price, sound quality, and ease of use.

Fig. 3. USHONK USB mini speakers used for device response and media playback. Image credits to www.amazon.com.

D. Camera

Because the resolution of the camera is set to 320x240, an inexpensive camera module was chosen. It had to be compatible with the Raspberry Pi in order to make use of the Camera Serial Interface. The camera chosen for the device was the Keyestudio camera module Rev 1.3.

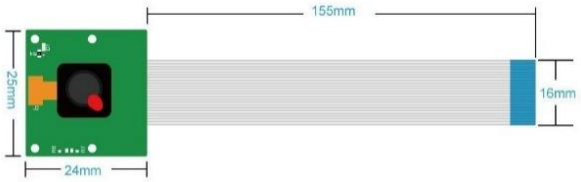


Fig. 4. Keyestudio camera module Rev 1.3 which is compatible with the Raspberry Pi Compute Module 3, which can also be used for prototyping with the Raspberry Pi 3 microcontroller. Image credits to www.amazon.com.

E. Wi-Fi

The ESP 12-E Wi-Fi chip is used to provide the device with wireless internet access. Internet is mainly used for the device to stream media and search open libraries for device response as an assistant.

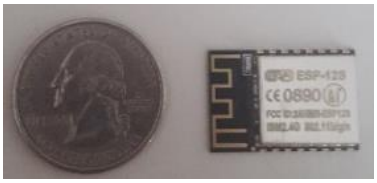


Fig. 5. ESP 12-E WiFi chip which is connected to the main CPU for internet access.

F. LCD

The Pi Foundation model is a 7" LCD touch screen with a resolution of 800x480. The screen has capacitive touch, allowing for multi-touch interaction with the device - up to 10-finger touch capability in fact. It has an operational voltage of 10V, making it compatible with the power supply needed for the other modules. The main functionalities of the LCD are to display the GUI, display YouTube videos if necessary, and to also display GIFs.

Fig. 6. The Pi Foundation LCD screen. Image credits to Adafruit.

G. MCU and Motors

An Atmega328-au was chosen to be the motor control unit (MCU) because of its simplicity and price. It only needs to receive signals via I2C and generate PWM signals for the servo and an RGB LED. An Attiny was also considered but did not have enough GPIO pins. The au form factor was chosen to reduce the footprint on the PCB.

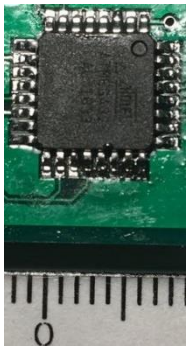


Fig. 7. The Atmega328-au chip that is already mounted on the PCB along with a ruler below it for size measurement. It is responsible for controlling the servo motor and an RGB LED

H. Storage

For storage, a microSD card socket was included to store the device's operating system and software.

Fig. 8. The microSD card socket used to store the device's operating system.

I. Troubleshooting

A USB-A socket is added to the device mainly for connecting the microphone and speakers that are connected through a USB hub, but it could also be used for troubleshooting. A keyboard or other peripherals can be used through this USB port to troubleshoot the device in case of any malfunctioning.

Fig. 9. The USB type A socket to be used for connecting the microphone and speakers along with troubleshooting if required.

J. Power Source

To power the device's PCB, a micro-USB port was used that can take an input of 5V and then voltage regulators do the rest of the work. A micro-USB port was used since micro-USB cables are very common amongst consumers.

Fig. 10. The microUSB port that is used to accept an input power source. It is responsible for powering the PCB.

K. Voltage Regulators

Since the device's main CPU requires multiple input voltages, three linear dropout voltage regulators were used to regulate voltages of 3.3V, 2.5V, and 1.8V. No voltage regulator was required for the 5V input since it was already regulated.

Fig. 11. The LM317MBSTT3 linear dropout voltage regulator used to regulate certain voltages required by the device's components.

IV. Software Design

A. GUI

The graphical user interface (GUI) functions as a secondary means of communication between the user and the device, behind voice control. It is the main point of integration between all that Ion can do including voice recognition, face detection and recognition, user data and profiling, applications such as Spotify, YouTube, and internet searches. One last major functionality of the GUI is to display Ion's emotional states as GIFs including smiling, yawning, etc. This provides Ion with a fun, interactive way to visually interact with the user.

The GUI is designed with Kivy, an open-source python library. Kivy is a library that is often used for development of natural user interfaces (NUIs) that involve multi-touch interfaces and widgets. It includes a library of touch-aware widgets and hardware accelerated OpenGL drawing which is perfect for GUI development. Python is a great language to develop a GUI in because it is easy to learn, is event-driven which is ideal for a GUI.

The GUI layout includes a front page which displays a welcoming GIF or face that is displayed on boot-up. From there, the user can tap the screen to navigate to the menu, where the user can see a clock with the current time as well as launch YouTube or the web browser. The user can also navigate to the 'Commands' page, where they can see the available voice commands that Ion recognizes.

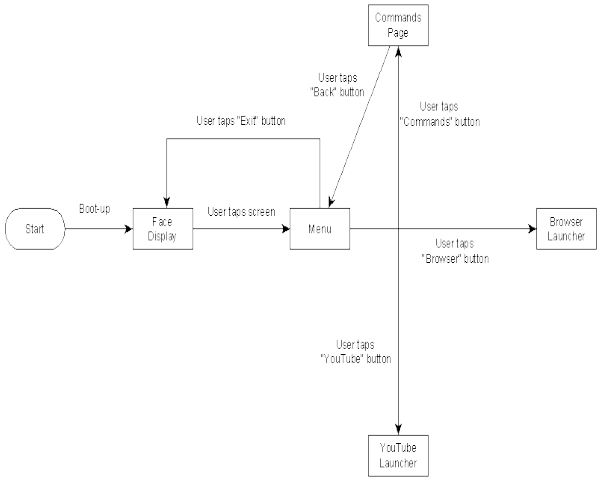


Fig. 12. The Graphical User Interface program flowchart.

B. Voice Recognition

The voice recognition is implemented using Google Speech API and utilizes the Google Assistant SDK which allows the target device to essentially function as a Google Assistant. With the Google Assistant, python scripts are used to create a request with the Google Speech API in the form of a JSON file. This JSON file contains the recorded audio that was picked up by the mic. From there, the device sends the JSON to the API which is processed into text and returns it back to the device in the form of a JSON file in a format that Ion can now recognize with the Google Assistant SDK. The voice recognition is further illustrated in a flowchart shown next page in figure 13.

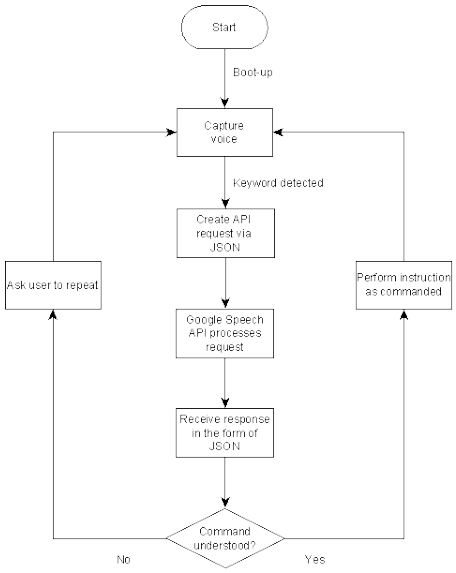


Fig. 13. The voice recognition flowchart.

C. Computer Vision

Face detection and face recognition are the components that make up computer vision (CV) for Ion. They are used to identify the user, detect the user's face location within the camera's frame, and to provide feedback to the 'face-the-user' system that allows Ion to follow the user's movement.

To identify the user, a face needs to be detected by the face detection algorithm and matched to a saved face in the user database. If a match is found, a prompt will confirm with the user whether the identification was correct or not. Once this has been done, only the face tracking feature will run until the face has been out of sight for a specified amount of time. The face detection feature determines the position of the face within the camera frame and either returns its location as a 2-D coordinate marking the face's center, or the image capture and the bonding rectangle of the face. The 2-D coordinates are used by the face-the-user system, the image while the image and bounding rectangle are used by the facial recognition algorithm.

The facial detection software's purpose is to detect faces in frames captures by the camera. The facial detection software was written in python and makes use of the OpenCV library. Python was chosen because of its readability and portability. It is pre-installed in the Linux operating system, specifically Raspbian which we are using, and has a broad spectrum of useful libraries that were used for Ion. Most programming was done in the Atom text editor and prototyped on a PC with the help of a webcam. Final adjustments for compatibility were made on a Raspberry Pi and its native python IDE.

The main hardware constraint for facial detection is processing power. Since the facial detection software is a key component of the servo feedback system, lag had to be reduced as much as possible. Therefore, input from the camera is set to a frame size of 320x240 pixels. In order to free up processing power for other programs, facial detection does not run continuously. The interval in which facial detection runs depends on the amount and priority of other simultaneous processes. Video playback and all other user experience centric processes are of higher priority.

Two facial detection methods were considered: Haar classifiers and linear binary patterns (LBP) classifier. Both of these methods have key advantages and disadvantages. Haar tends to be more accurate and have a low false positive rate. This reduces the chance for Ion to mistake an object for a person. However, Haar is slower and has problems in difficult lighting conditions. LBP, on the other hand, is fast and has less problems with changes in lighting conditions. The drawback is a loss in accuracy and more false positives.

The two most important factors are speed and accuracy, so it was difficult to select one method over the other. At the end, an implementation using the Haar method was running fast enough because the image size was reduced to 320x240 pixels and only a grayscale color palette is used. The algorithm does struggle a bit in lower lighting but fewer false positives do not cause Ion to turn away from the user as does happen with LBP classifiers. figure 14 shows the high-level flowchart of the face detection algorithm.

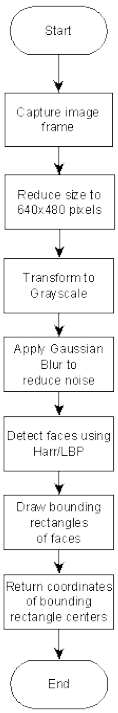


Fig. 14. The face detection algorithm.

The facial recognition software takes the output from the from the facial detection software (bounding rectangle of the detected face and the image in which it was detected) and tries to match the detected face to a known user from Ion's internal database. The framework use for the facial recognition algorithm is the same as for the facial detection algorithm described above.

The facial recognition software was implemented with Eigenfaces. Eigenfaces (Principal Component Analysis) uses a training set of pictures taken in the same lighting conditions. Eigenvectors and eigenvalues will then be computed from a covariance matrix that represents the set of training pictures. These so-called eigenfaces can then be used to compute the difference of a face to these eigenfaces. Comparing these deltas of various pictures will generate a match (with determined range) if the face in the pictures is the same. Because of the reliance in even lighting conditions, this method requires a controlled environment. All compared pictures must also be of the same size and be normalized.

The advantage of using eigenfaces is that it is very fast and does not need a lot of storage space and is very accurate when the lighting conditions are ideal and unchanged. It also does not need a very high-resolution image which speeds up processing time and reduces storage needs even further. The drawbacks are that it handles varying lighting conditions, change in facial expression, varying background, and faces of different sizes poorly. The recognition depends on lighting rather than facial features. [1]

Because of its drawbacks, the images get cropped around the face-binding rectangle produced by the facial detection algorithm. This reduces the effect the background has on the method. The cropped image is then resized to 100x200 pixels, which should normalize it (face centered in the image and of a uniform size). To reduce the errors when the user's face is tilted or rotated, several different sets of eigenfaces should be generated by asking the user to take several pictures from varying positions when calibrating the system. figure 15 on the right shows the flowchart of the facial recognition algorithm.

Facial detection for Ion is not advertised as a security feature because of accuracy concerns but rather as an interactive feature helping to humanize Ion.

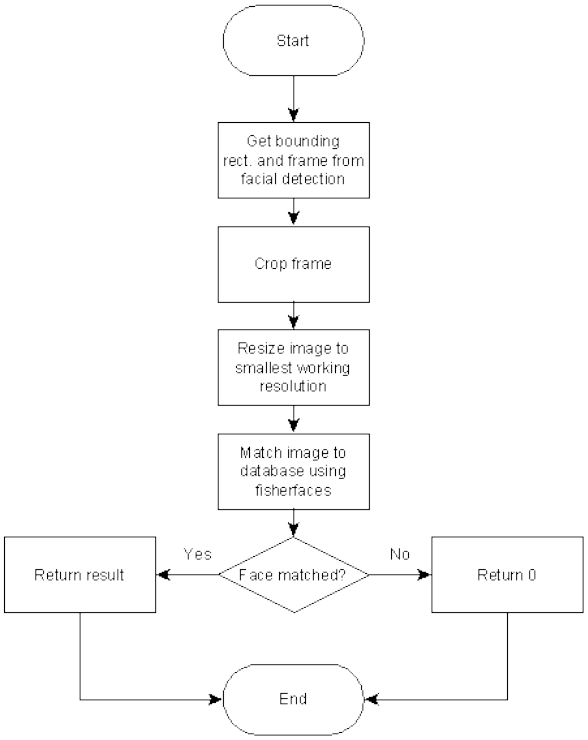


Fig. 15. The face recognition algorithm.

V. Project Design

A. Power Consumption

To approximate the device power consumption, although not completely accurate since power in the device is not constant, a table, table 1 shown below, was constructed illustrating all the nominal input voltages, current draws, and power consumption by each of the components that require power from the supply. It was then concluded that the device easily matches its power consumption requirement of less than 50W.

Table I
Power Consumption

Component	Nominal Input Voltage (V)	Nominal Current Draw (mA)	Power Consumed (mW)
Processor	5	16	80
LCD	5	~500	~2,500
Servo Motors	5	500	2,500
Motor Control Unit	5	0.5	2.5
WiFi Chip	3.3	80	264
Camera	5	250	1,250
LEDs	5	~20	~100
microSD Card	3.3	100	330
USB Peripheral	5	~500	~2,500
Total:		~2.0A	~9.5W

B. Power Connections

After the power consumption table was established, it helped with building a power connections diagram, shown below in figure 16, that gives a better idea on how to properly power the components on the PCB and where to place them. The final decision for the component placement on the PCB was done after the data connections were considered as well.

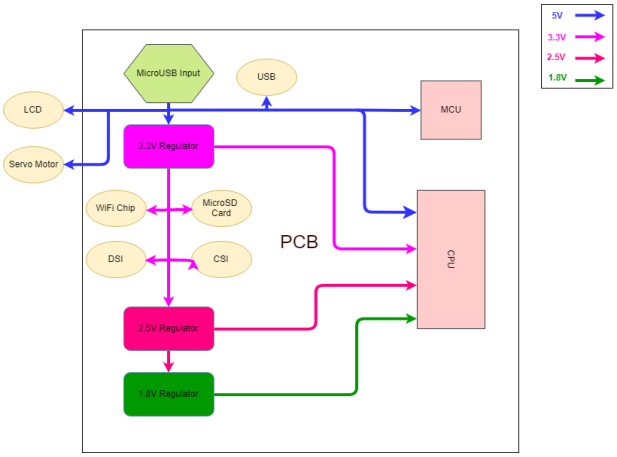


Fig. 16. PCB power connections diagrams, whatever is inside the square is mounted on the PCB while the outside is connected externally.

C. PCB Schematic

Once all the power connections were planned out, it was simple to design the PCB schematic. The schematic was divided into multiple parts and sorted into categories for better organization. Shown below in figure 17 is one of the parts of the PCB; the Motor Control Unit (MCU). The MCU is responsible for handling the PWM for the servo motor that will rotate the device and for blinking RGB LEDs.

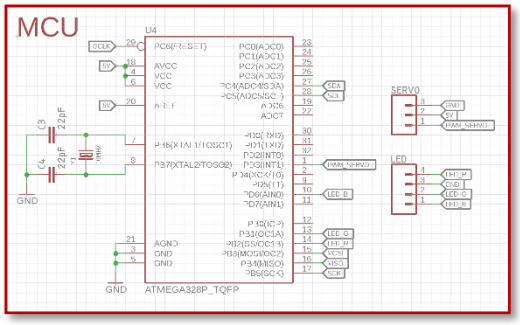


Fig. 17. The MCU chip schematic drawn on EAGLE along with its connections.

Next, shown below in figure 18, is the group's CPU, the Raspberry Pi Compute Module 3. It is responsible for all the rest of the main functions and processes done by the device. The 100kΩ resistors shown in the schematic are pull-up resistors to prevent GPIO floating.

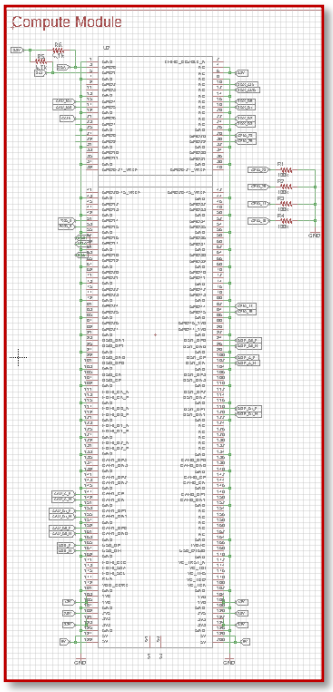


Fig. 18. The CPU chip schematic drawn on EAGLE along with its connections.

After that, shown below in figure 19, are the device peripherals. In that section, there is a three-pin male connector for the LCD 5V input and ground (One pin was reserved for future functionality), a USB-A port for the microphone and speakers/troubleshooting, the ESP-12E Wi-Fi chip, the camera and display serial interfaces, and finally the microSD card socket along with all their connections.

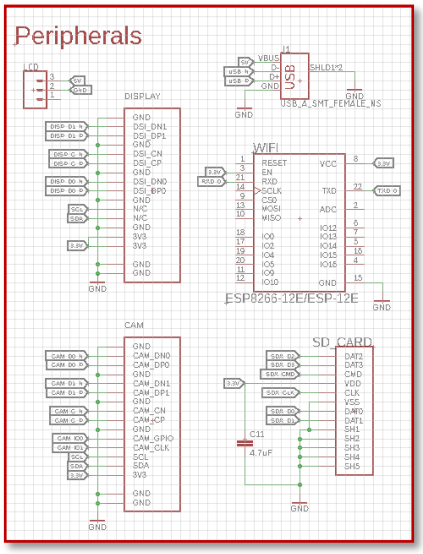


Fig. 19. The PCB peripherals drawn on EAGLE along with their connections.

Finally, shown below in figure 20, is the power section. It shows the microUSB power input, along with the three voltage regulators that are connected in a certain sequence. As the CPU requires, the sequence is that it should receive its input voltages from highest to lowest. So, the output of every higher voltage regulator is connected to the input of the lower voltage regulator. That was also to have less power dissipated in every regulator; less power wasted and less heat in each.

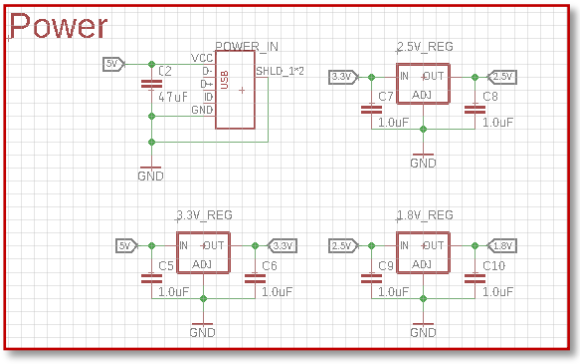
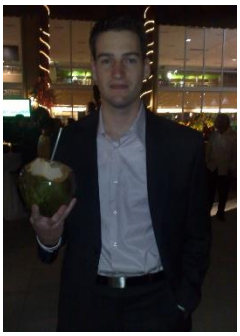


Fig. 20. The PCB power section drawn on EAGLE that shows the input power going through the micro-USB port and the three voltage regulators along with their respective connections.

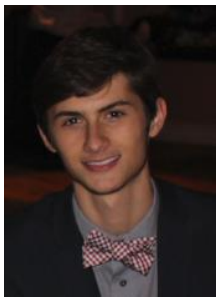
ACKNOWLEDGEMENTS

The members would like to thank Dr. Lei Wei and Dr. Samuel Richie for their dedication to helping students throughout the semesters.

BIOGRAPHIES



Dennis Gebken is a senior CpE student at UCF, graduating in August 2018. He opened a family restaurant, completed two internships, and worked full time as an engineering coordinator for Universal Creative's engineering division. He will be joining Universal Creative as an associate engineer upon graduation.



David Jaffie is a senior CpE student at UCF, graduating in August 2018. He completed two internships throughout the course of his undergraduate degree, both in military and DoD simulation. He will be joining Veraxx Engineering Corporation as a Software Engineer upon graduation.

Mohammad Rizeq is a senior EE student at UCF, graduating in August 2018. He worked part time with the Student Academic Resources Department as a Supplemental Instruction Leader for two semesters for the Electrical Networks course. At the time of completion of this project, he is undecided where he wants to work full-time.

REFERENCES

[1] Philipp Wagner, "How to implement the Eigenfaces and Fisherfaces algorithm in GNU Octave/MATLAB and Python", https://github.com/bytefish/bytefish.de/blob/master/blog/fisherfaces.md, August 2017