Backpack Buddy: Wearable Nighttime Safety Device

Rahul Kajjam, Jeric Cuasay, Emily Grob

ECE 445 Final Paper – Spring 2023

Team #8

TA: Zicheng Ma

Abstract

This project aims to reduce the stress of walking home alone at night by providing a backpack interface that detects and alerts you to potential threats. The Backpack Buddy uses a camera to specifically identify people approaching you from far away distances and responds with haptic feedback to alert you about a potential threat. Up close, the Backpack Buddy will send an automated text message to an emergency contact upon incident detection. This report outlines the design, verification and results of our project. Apart from a few minor details, we have achieved full functionality of our project.

Contents

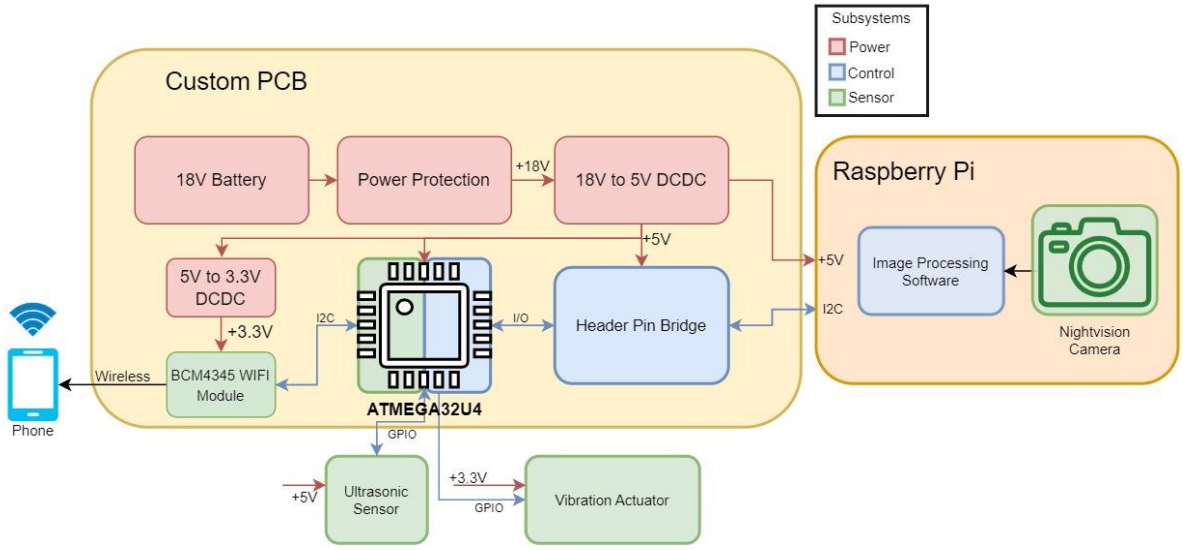
1. Introduction

1.1 Objective and Solution

Many college students walk home late at night, and even though safety resources like free transit and emergency buttons are available, walking home is often the most convenient option. However, when residential sidewalks are deserted at night, the risk of being approached or followed from behind is high. The Backpack Buddy helps reduce this risk by providing users with enough time to check their surroundings and get to a safe place if necessary. The Backpack Buddy is a discreet wearable that alerts the user to pedestrians behind them. The device should distinguish pedestrians from other moving objects in low light and alert the user if they are too close or on a collision course. The device will use a night-vision camera and image processing to detect pedestrians, and will alert the user with haptic feedback. The device also has emergency detection capabilities where a text alert is sent to a predefined contact upon incident detection.

1.2 Subsystem Overview

Figure 1.2.1: Block Diagram



The design shown in the block diagram shows two primary components: a custom PCB and a Raspberry Pi. The purpose of the Raspberry Pi will only be for image capture and processing. The data is then sent from the Pi to our PCB where the microcontroller will make future decisions about sending haptic feedback or triggering the automated phone text message system. Additionally, there are three major subsystems shown above. The power subsystem, in red, now has an 18V input rather than a 9V input. In green, are the sensors and actuators. Initially, our wifi module was an ESP8266, but in our final version of the project we used a BCM4345. To simplify our design, we ended up not needing the passive infrared sensor that we initially discussed in previous versions of our project.

1.3 Visual Aids

Figure 1.3.1: Visualization of Physical Design

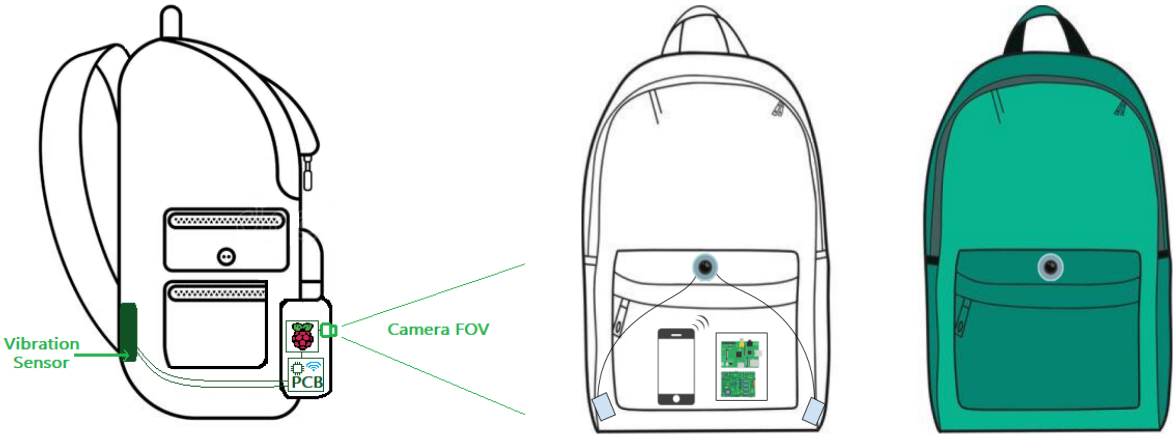


Figure 1.3.2: Physical Design

1.4 Performance Requirements

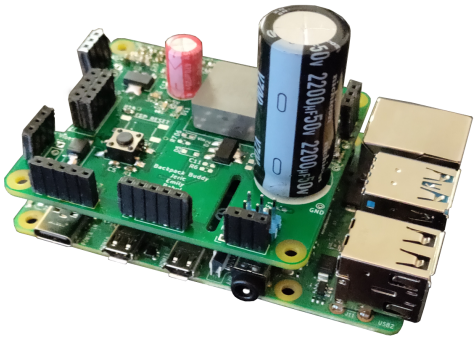
1. Distinguish pedestrians from other moving objects at a rate of 4-6 frames per second
2. Alert the user with haptic feedback if a pedestrian is less than 3 meters away
3. Send emergency alerts to given emergency contact if a pedestrian is within 30cm

2. Design

2.1 Hardware Design

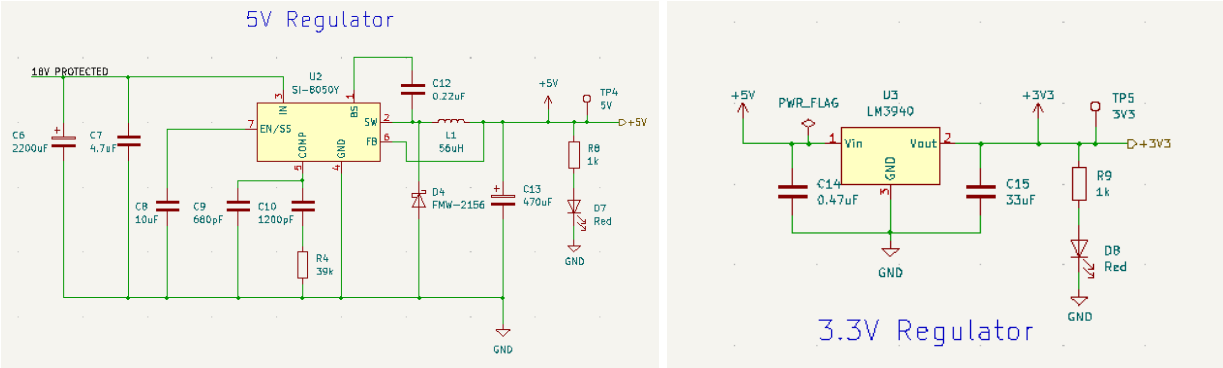
We were always planning on using a Raspberry Pi for image processing due to ease of access and cost, thus our hardware design process is focused on the intention to make our board an IO Shield or Pi-hat that could be easily plugged into the Raspberry Pi. For our crystal oscillator, we sized its necessary capacitors with the following equation: $CL = (Ca * Cb)/(Ca + Cb) + Cstray$ [2]. Knowing that the standard CL for our crystal oscillator is 18 pF and that Cstray varies around 7 pF, then, keeping symmetry, we calculate Ca and Cb to be 22 pF. Appendix C hosts the rest of our component values.

Figure 2.1.0: PCB Plugged into Raspberry Pi



2.1.1 Power Subsystem

Figure 2.1.1: Power Sheet



Initially, we planned for our design to have a 9 V battery input, due to ease of access and because our system would be operating at around 5 V, so it was important for the input voltage to be approximately double the battery voltage to account for terminal impedance. Table 1 shows the system current draw that we initially planned for. Based on this information, we wanted to make our circuit tolerant for double that amount of current, to accommodate for any additional ESR, trace losses, or any unexpected current spikes. This calls for an 8 A tolerant system, thus we choose the SI-8050Y switching converter. We used a switching converter, as linear regulators are very lossy and dissipate more heat, which introduces unnecessary thermal concerns. Not only is our power subsystem 8 A tolerant, but it can step-down

voltages from 8.6 - 40 V. However, because as batteries discharge their voltage levels will also reduce, this 8.6 V minimum threshold did become a concern for our planned 9 V input. Therefore, we opted to use two 9 V batteries in series to have an 18 V input.

For our 3 V step-down converter we used a LM3940 LDO rather than a switching regulator, because the voltage drop is low therefore efficiency is not a major concern. Also, this minimizes any noise that may be introduced by an additional switching regulator, as they have higher frequency components.

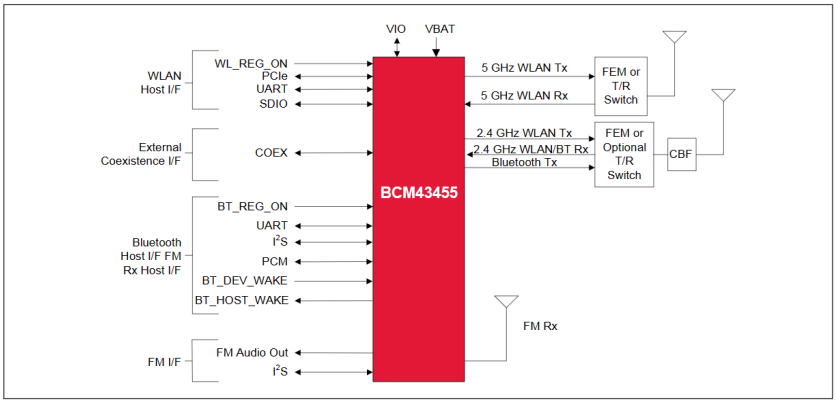
Table 1: Initial Design System Current Draw

Component	Maximum Current Draw
Raspberry Pi	3.5A
ATMega32U4	200mA
Vibrating Mini Motor Disc x2	200mA
ESP8266 Wifi Module	12mA
HC-SR04 Ultrasonic Sensor x2	30mA
HC-SR501 Passive Infrared Sensor	65mA
Total	4.0A

2.1.2 BCM4345 Wifi Module

For our wireless subsystem, we decided to use wifi over other forms of wireless communication, for its pre-existing documentation and simplicity. The ESP8266 was our original choice for wifi communication; however, the module is tedious to program and difficult to debug. More importantly, we ran into DNS resolution issues while using the ESP8266. Thus, we ultimately chose to use the BCM4345, which is more easily integrate-able with our other subsystems and is much more straightforward to test and debug.

Figure 2.1.2 BCM4345 Functional Block Diagram



2.1.3 HC-SR04 Ultrasonic Sensor

In terms of depth sensing, having a built in depth sensor with our camera is very costly. As an alternative, we wanted to have an external depth sensor that was extremely cost effective. One common architecture for motion and depth sensing is the HC-SR04 ultrasonic sensor, which is well-documented, cost-effective, and provides real time distance data from the sensor, something very useful for firmware integration.

2.1.4 Vibration Actuator

As we wanted to discreetly warn our users of any approaching pedestrians, we decided to use haptic feedback rather than flashing lights or alarms which may cause panic. Hence, we are using the Adafruit Vibrating Motor Mini Disc, which has a small form factor and low current draw. As motors can introduce noise into our power lines, we also have designed a transistor circuit to isolate the motors' current draw from the GPIO pins of our ATMega32U4 microcontroller.

2.2 Firmware and Raspberry Pi Interface

Our design utilizes a Raspberry Pi for image processing and delegates other processing and communication tasks to an ATMega32U4. The ATMega32U4 acts as a communication hub, requiring a parallel interface to handle communication between multiple sensors. Initially, we intended to use GPIO pins as the primary communication mechanism. However, after testing, we determined that an I2C interface was optimal due to its ability to handle parallel processes communicating. To ensure seamless transmission and receipt of messages, we tested and developed a stable interface using an I2C interface. We then integrated sensors into the framework to create a functional prototype of the communication system between the ATMega32U4, Raspberry Pi, and sensors.

2.2.1 Raspberry Pi Integration with Sensors

Our aim in this development was to design a system where the Raspberry Pi triggers the image processing algorithm and notifies the Arduino when a pedestrian is detected within the camera's view, along with their relative location (right, left, or middle). The Arduino then activates one or both of the vibration modules to indicate the side where the pedestrian was detected. Simultaneously, the Arduino monitors the ultrasonic sensor and sends an alert to the Raspberry Pi if an object approaches closer than a specific threshold, which then triggers the emergency protocol.

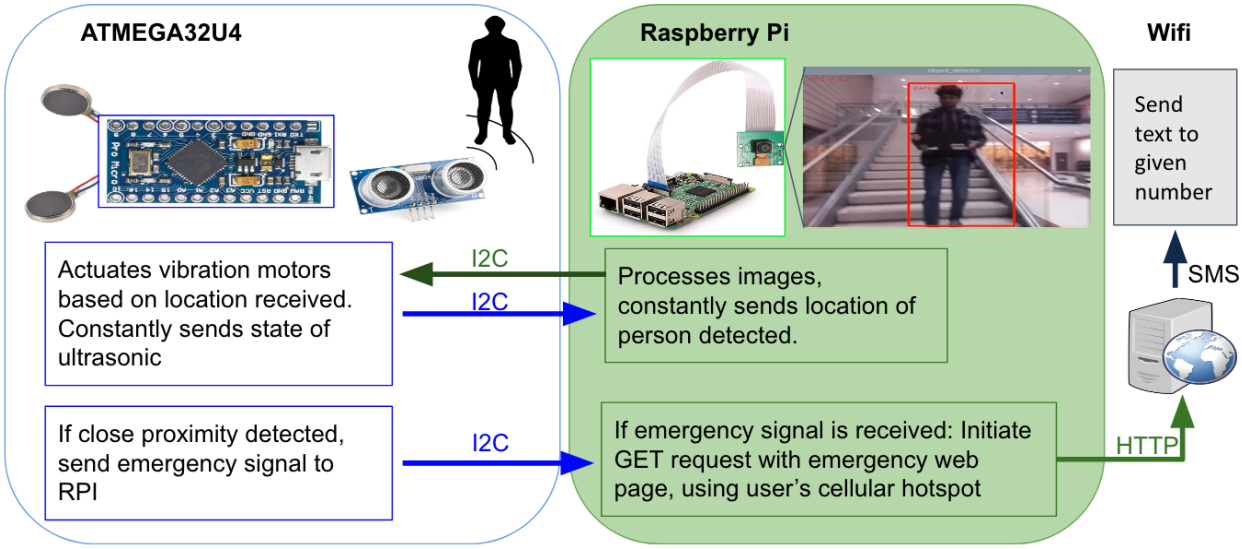


Figure 2.2.1: The communication between the Arduino and the Raspberry Pi, and the corresponding interface with the sensors and actuators.

2.2.2 Firmware Flowchart

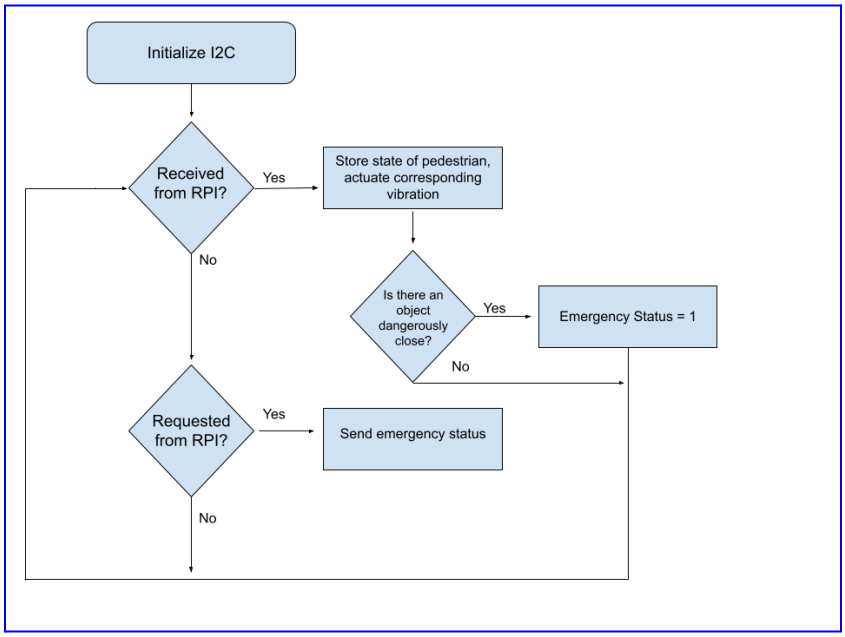


Figure 2.2.2: Arduino flowchart

Figure 2.2.2 demonstrates the flowchart for the final algorithm on the Arduino. The communication framework we developed allows the Arduino to constantly be checking if it has received a message from the Raspberry Pi while also checking if the Raspberry Pi has requested information. Similarly, the

Raspberry Pi communication framework also simultaneously checks if the Arduino has requested or sent data. The algorithm on the Raspberry Pi is shown in Figure 2.2.3.

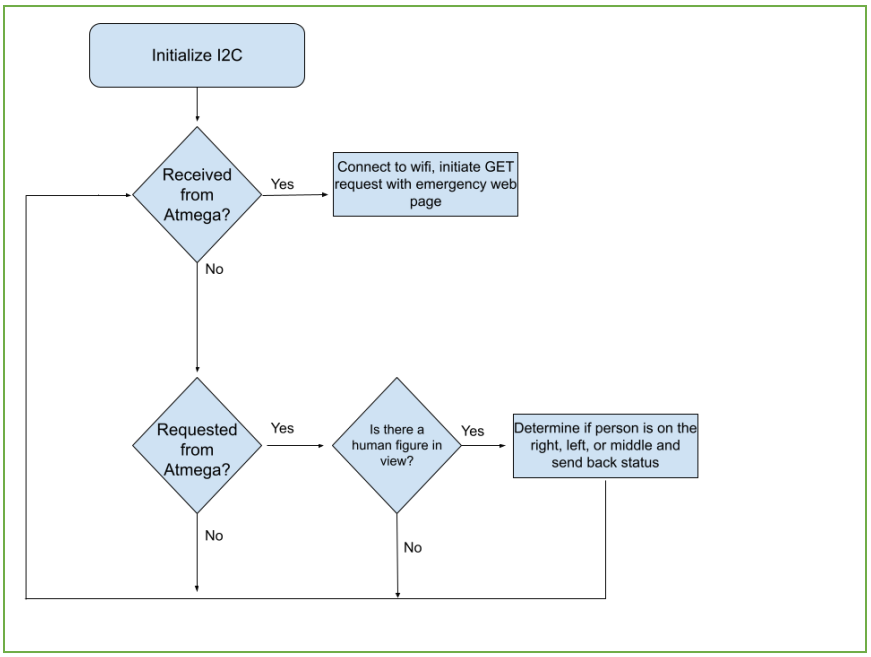


Figure 2.2.3

2.3 Image Processing Software

The night vision camera on the Raspberry Pi is responsible for capturing a continuous video behind the user when triggered. This continuous feed is then taken through a series of filters which detect and identify the location of a person in each frame. Based on this data, the PCB will more accurately send a warning to the user. Figure 2.3.0 illustrates the high level procedure of our goals.

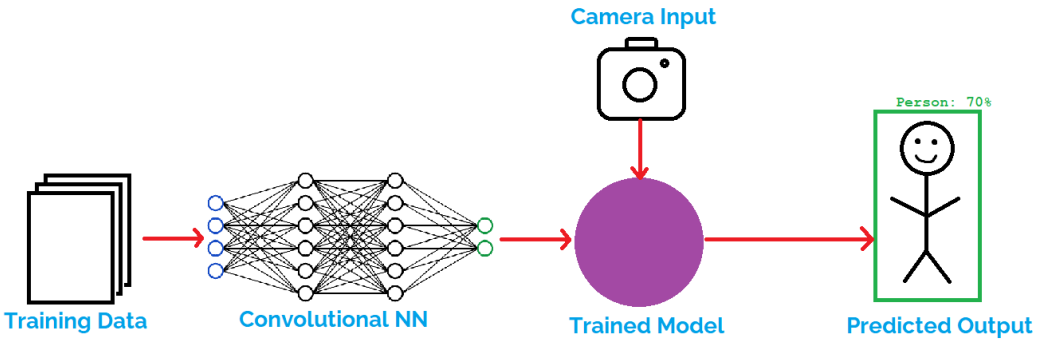


Figure 2.3.0: High level process for object detection

2.3.1 Approaching Object Detection on a Raspberry Pi

When reading into object detection algorithms, we discovered that many object detection algorithms are very computationally heavy, so the primary objective for us was to look for existing algorithms and

architectures that would be able to run on the Raspberry Pi. Initially, we looked to use the YOLO (You Only Look Once) algorithm [6] because it markets its approach as a higher reduction in latency on a trained model. However, there was still not enough computational power on the Raspberry Pi for us to use YOLO as our detection method. After looking further into object detection models, we came across the MobileNet algorithm that allows smartphones to perform object detection. This was a much better starting point because although some smartphones have higher processing power than a Raspberry Pi, the results would be relatively similar.

2.3.2 Using Object Detection with a Pretrained Model

While looking through the implementation of MobileNet, we came across a tutorial that uses MobileNet on a pre trained architecture using the COCO (Common Objects in Context) dataset. This dataset is composed of 80 object categories and is trained on 330K images [7]. We utilized the libraries of Tensorflow lite to deploy MobileNet v1 onto the Raspberry Pi. Once this was deployed, we were able to identify a variety of household objects with varying degrees of accuracy. After seeing the results of this model, we realized a big problem with this is the latency. When the camera is still, the screen can display results that average to 4 frames per second. However, when the camera is in motion, the frames drop down to 3 or even 2 FPS. This poses a problem because a lot can happen in between 2 frames. Due to this issue, we looked into optimizing the model by training it with custom data, choosing only two classes to identify, and searching for an even faster architecture.

2.3.3 Training a Model with a Custom Dataset

After realizing the inconveniences of using the pretrained MobileNet architecture, we looked into using EfficientDet as an alternative. Like YOLO and MobileNet, EfficientDet is also a single shot detector (SSD) [8]. SSD works by dividing the image into a grid and having each grid accountable for detecting an object in that location. If nothing is detected, the grid is considered to be a background region. The advantage of EfficientDet is that it is 4x - 9x smaller and uses 13x - 42x fewer FLOPS than previous detectors such as YOLO [9]. It was difficult for me to find a comparison between MobileNet and EfficientDet, so we decided to try both and see which had lower latency.

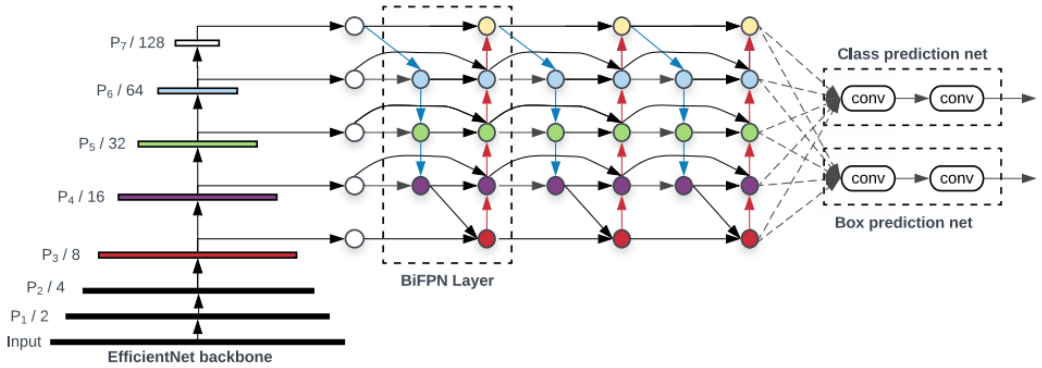


Figure 2.3.3: EfficientDet Neural Network architecture

To start, we began gathering training images of people from the internet. Instead of manually downloading images, we were able to use Google's Open Images dataset to filter any image containing a person and use a script to download as many images as we desired. Initially, we only wanted to use a few training images to save time annotating each and reduce model training time. Therefore, we initially started by annotating 29 training images and 10 test images. In each image, we identified the objects in 2 classes; either a person or a group of people. After collecting these annotated images, we used the EfficientDet0 architecture to train the model. Figure 2.3.4 shows the various differences between each version of EfficientDet. Our main problem is the latency, so we decided to use the model with the lowest latency possible.

Model architecture	Size(MB)*	Latency(ms)**	Average Precision***
EfficientDet-Lite0	4.4	146	25.69%
EfficientDet-Lite1	5.8	259	30.55%
EfficientDet-Lite2	7.2	396	33.97%
EfficientDet-Lite3	11.4	716	37.70%
EfficientDet-Lite4	19.9	1886	41.96%

Figure 2.3.4: EfficientDet architecture differences. Latency is measured on the Raspberry Pi 4 and average precision is the mean average precision on the COCO validation set.

In the paper describing the details of the algorithm, we discovered that each model is trained using a standard gradient descent optimization that uses momentum 0.9 and weight decay 4e-5. A step learner is used for the learning rate which initially increases from 0 to 0.16 in the first epoch but then slows down using the cosine decay rule [9].

With the smaller training set, we used 29 training images, 20 epochs, and a batch size of 4 to obtain a loss of 1.15. After training this model, it was deployed on the Raspberry Pi to yield 6 FPS while the camera was

stationary. After determining that this model worked, we spent time annotating 500 training images to increase the model precision. With 500 images, we used 50 epochs, and a batch size of 32 to yield a loss of 0.355 – a much lower value.

Figure 2.3.4: Object detection model using 500 training images on EfficientDet

3. Verification

3.1 Integrated System Verification

Overall, we were able to meet all three high level requirements. Using the camera, we are able to detect anyone following the user at a framerate of 4-6 frames per second at a distance of over 3 m. The system worked even while testing at 11:30 pm in the middle of the night. With any person detected by the camera, the system successfully warns the user of such presence using the vibrational modules. If at close proximity, that is, at 30 cm or less, the ultrasonic sensor will detect any such presence and trigger an emergency signal which allows the wifi module to successfully trigger the SMS response.

Appendix B shows all verification tables and includes the final results of each requirement. Some important things to note are the failure of the PIR sensor requirements due to high sensitivity to movement, which made integration into our backpack-based design very difficult. Thus as a contingency, we decide to eliminate the use of the PIR sensor as it has no significant impact on meeting the high level requirements. Additionally, while testing with the ESP8266 module, we struggled to make an HTTP connection despite having wifi connection. As a contingency, we used an alternative wifi module which had greater serial monitor and debug capabilities in order to meet our desired functionality. Lastly, after testing in complete darkness and losing person detection capabilities, we introduced an IR flashlight into our system that would act as a discreet solution for situations lacking ambient lighting such as street lights.

3.2 Hardware Verification

The integration process saw many difficulties. During initial testing phases, our circuit was not capable of outputting 5 V from a 10 V power supply input but would rather float at 200 mV. Our switching power supply would also get hot. The issue lied in using an incorrect footprint for our part, thus traces were connected in an incorrect way. After debugging this issue for the 5 V step down, we saw a similar problem for the 3.3 V step down, and made new board revisions accordingly. There were also many struggles in programming the microcontroller, as we planned on using micro USB but were not able to get it working. Our final hardware revision includes an ISP programmer header and corrected footprints.

3.3 Firmware Verification

To simplify the debugging process, we developed the firmware incrementally and verified each stage as it was created. After achieving a fully functional communication framework where the Arduino and Raspberry Pi could communicate and receive all required information simultaneously, we shifted our focus towards user-centric functionality. The most important aspect of this was timing. The

communication needed to be fast enough to provide quick feedback in response to the environment, while also being carefully timed to prevent I2C timing issues.

3.3.1 Accuracy and Functionality

To ensure accurate transmission and reception of information, we included a 100ms delay on both the Arduino and Raspberry Pi after each reading and transmission of data before restarting the read/transmit process. Additionally, we implemented counters for each transmitted datapoint to ensure accuracy. These counters counted the number of messages received indicating consistent information. Only if a device received a threshold number of messages with consistent information, would it actuate a vibration module. This helped improve the accuracy of our image processing algorithm when detecting a person. For instance, a person had to be detected in 10 subsequent readings of the same object in the same relative position to actuate a vibration module. Similarly, in an emergency scenario, the Raspberry Pi wouldn't initiate the emergency text sequence until it had received 10 subsequent emergency messages from the Arduino. If a person is continuously in the frame, we ensured that the backpack would not vibrate constantly. We made sure that the vibration modules would vibrate in 10 second increments if someone is constantly detected. We considered this timing fast enough to be considered a real-time alert.

The firmware uses the camera's perspective to determine the relative location of a pedestrian. To verify this would align with the user's intuition, we tested the device on a person and had them specify what direction they were feeling as we moved behind them. In this testing process, we determined that as long as the camera was positioned correctly in the middle of the backpack, the positional vibration would align with a pedestrian's relative position with the user.

3.3.2 Emergency Procedure and Wifi

To verify our emergency procedure was working, we manually triggered the emergency event by triggering the ultrasonic sensors and timed how long it took for the emergency contact to receive a text. If a user's phone has a cellular hotspot, we assume that the wifi module will automatically join the hotspot. This assumption is valid for most modern cell phones, including the iPhone XS which we tested on. The emergency text was consistently sent within 1-2 minutes.

3.3.3 User Experience

Timing is also added throughout the firmware to improve user experience. When a person is detected behind the user, the corresponding vibration module will vibrate for 300 milliseconds and will wait 10 seconds to vibrate again. Similarly, to avoid emergency texts being sent excessively during a detected incident, we have timing constraints in place to only allow texts being sent once per minute.

3.4 Software Verification

3.4.1 Custom Trained Model Dataset Results

As mentioned in Figure 2.3.4, the custom training set had been able to somewhat identify a person correctly in an image. To quantify the accuracy of these tests, we used 10 test images to evaluate the results of the model once on the 29 images, and another time on the model trained with 500 images and tested with 100 images. The evaluation metrics can be confined to average precision and average recall [10] [11].

$$Average\ Precision\ =\ \frac{True\ positive}{True\ positive + False\ Positive}$$

$$Average\ Recall\ =\ \frac{True\ positive}{True\ positive + False\ Negative}$$

AP - The average precision of the intersection of bounding boxes. This value is the average of ten thresholds that range from 0.5 to 0.95 in increments of 0.05.
AP50 - The average precision of predicted bounding boxes that intersect over 50% of the annotated box.
AP75 - The average precision of predicted bounding boxes that intersect over 75% of the annotated box.
APs - The average precision across scales of predicted bounding boxes for small boxes: area < 32^2
APm - The average precision across scales of predicted bounding boxes for med boxes: 32^2 < area < 96^2
APl - The average precision across scales of predicted bounding boxes for large boxes: area > 96^2
ARmax1 - Average recall given one detection per image
ARmax10 - Average recall given ten detections per image
ARmax100 - Average recall given 100 detections per image
ARs - Average recall across scales of predicted bounding boxes for small boxes: area < 32^2
ARm - Average recall across scales of predicted bounding boxes for med boxes: 32^2 < area < 96^2
ARl - Average recall across scales of predicted bounding boxes for large boxes: area > 96^2
AP_/person - Same as AP, but only looking at a the class “person”
AP_/person_group - Same as AP, but only looking at a the class “person_group”

```
{'AP': 0.16598538,
 'AP50': 0.39855877,
 'AP75': 0.12706271,
 'APs': -1.0,
 'APm': 0.028104251,
 'APl': 0.23360266,
 'ARmax1': 0.05,
 'ARmax10': 0.234375,
 'ARmax100': 0.309375,
 'ARs': -1.0,
 'ARm': 0.18181819,
 'ARl': 0.37619048,
 'AP_/person': 0.16598538}
```

Figure 3.4.1: Metrics used for evaluating the small training dataset (29 images)

```
{'AP': 0.27443424,
 'AP50': 0.7327121,
 'AP75': 0.14436093,
 'APs': -1.0,
 'APm': 0.2934093,
 'APl': 0.29503432,
 'ARmax1': 0.075,
 'ARmax10': 0.375,
 'ARmax100': 0.434375,
 'ARs': -1.0,
 'ARm': 0.37272727,
 'ARl': 0.46666667,
 'AP_/person': 0.27443424,
 'AP_/person_group': -1.0}
```


Figure 3.4.2: Metrics used for evaluating the large training dataset (500 images)

When evaluating the models on the same training data, it is clear that the dataset that used 500 images to train had a much higher average precision. In this particular case, there were no test images annotated with person_group, so the precision for this value is irrelevant. Therefore, AP and AP_/person are the same value. For 500 images, the average precision resulted in 27.44%. This is above our expected precision value of 25.69% from Figure 2.3.4.

3.4.2 Low Light Testing

The last component of verifying the object detection model was to test how our system worked at night. Initially, this was done in the ECE 445 Lab by slowly dimming the lights in the room. Once we tested this, we soon discovered that our camera is not sensitive enough to low light conditions. To improve upon this, we added IR light to illuminate the FOV of the camera. With the initial setup, we saw drastic improvement while testing in the senior design lab. The next step of this was testing in an outdoor setting. Our verification table declared that our object detection should be able to work at least 30 minutes before sunrise and 30 minutes after sunset. We tested this by going outside with the IR light at 11:30PM and ensuring our object detection model worked. We found that it worked very well when the IR light illuminated the person for a distance over 3 meters.

4. Cost and Schedule

4.1 Cost

Table 2: Cost Analysis

Part	**MPN**	**Manufacturer**	**Description/Notes**	**Quantity**	**Cost**
Ultrasonic Sensor	HC-SR04	Sparkfun	ultrasonic low range	1	4.50
PIR Sensor	HC-SR501	Xiaohunike	passive infared, longer range	1	8
Night Vision Camera	OV5647	MakerFocus	5MP 1080P camera	1	17.99
3.3V Regulator	LM3940	Texas Instruments	5V to 3.3V, 1A max output	1	2.84
5V regulator	SI-8050Y	Sanken	9-5V, 8A max output	1	2.26
9V Battery	–	EBL	5400mWh Li-Ion	4	23
Microcontroller	ATMega32U4	Microchip Tech	5V operating voltage	1	5.68
Vibration Sensor	Adafruit Motor Disc	Adafruit	11000 RPM at 5V (x2)	2	17.70
Wifi Module	ESP8266	Sparkfun	3v3 operating voltage	1	7.50
Backpack	–	Jansport	Standard Lightweight Schoolbag	1	20.00
IR Light	-	WAYLLShine	For night time detection	1	20.00
Total Cost (Materials)	**–**	**–**	**–**	**–**	**$129.47**
Labor	–	–	$38.60/hr for UIUC EE from [7]	144 hrs/person	16,675.20
Total Cost (with Labor)	**–**	**–**	**–**	**–**	**$16,804.67**

4.2 Schedule

Table 3: Schedule

Week	Task	Assignee
2/26	Finalize schematic	Jeric
	Refine library for pedestrians	Rahul, Emily
3/5	After receiving components, begin verification	Everyone
	Test algorithm in low-light and various moving conditions. Begin firmware	Rahul, Emily
	Begin testing ultrasonic and PIR in low light conditions	Jeric
3/12	Complete software to distinguish human from other moving object, send corresponding signal to ATmega	Emily
3/19	Begin integrating subsystems	Everyone
	Finalize datasets for image processing algorithm	Rahul
3/26	Second Round PCB Orders Due	Jeric, Rahul
	Finalize PCB design and component selection	Everyone
	Individual Progress Reports	Everyone
4/9	Finalize software and firmware components	Emily
	Team Contracts	Everyone
4/16	Mock Demonstration and debug	Everyone
4/23	Final Demonstration	Everyone
4/30	Final Presentation and Final Paper	Everyone

5. Conclusion

5.1 Accomplishments

Throughout the semester, we were able to meet all major project goals and fit them all onto one small PCB. We successfully developed an image processing algorithm that can successfully detect human beings, and use that information as the main input to a night time safety wearable capable of not only discreetly letting users know they are being followed from behind, but also capable of sending an SMS message to an emergency contact if approached too closely. The level of cooperation and work necessary to engineer this scope of project has instilled in us invaluable hands-on experience that we will take with us to our future endeavors.

5.2 Ethical Considerations

Ethical issues lie in the nature of the image processing algorithm. Because our priority is to classify and localize an object in an image quickly, it will be prone to numerous errors. This can result in a higher number of false positives and negatives when recognizing pedestrians. It is important to ensure that privacy is maintained while video may capture someone unknowingly [3]. To prevent this, we will make sure that the processed image data will not be stored after its use; however, in the case that a figure does approach the user, a divergence can be made where an image can be saved in order to assist law enforcement in the case of emergency.

5.3 Broader Impact

In a larger context, our project hopes to address the issues of night time safety for pedestrians. Ultimately, the desired societal impact of our project would be to improve night time safety. If our product becomes common enough and well known to the general public, hopefully it would deter situations where someone may be approached unexpectedly or followed at night.

References

1. IEEE.org, "IEEE Code of Ethics", IEEE Policies. https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: Feb 6th, 2023].
2. IQDFrequencyProducts.com "Myths around load capacitance - how to choose the right capacitors", Blogs & Press. https://www.iqdfrequencyproducts.com/blog/2020/08/03/myths-around-load-capacitance-how-to-choose-the-right-capacitors/ [Accessed: May 2nd, 2023].
3. Alldatasheet.com, "BCM43455 Datasheet (PDF) - Cypress Semiconductor", Electronic Components Datasheet Search. https://pdf1.alldatasheet.com/datasheet-pdf/view/828943/CYPRESS/BCM43455.html [Accessed: May 2nd, 2023]
4. Sparkfun “ESP8266 Wifi Module” https://cdn.sparkfun.com/assets/f/e/5/6/f/ESP8266ModuleV2.pdf [Accessed: Mar 23rd, 2023]
5. Google “Open Images Dataset V7 and Extensions” https://storage.googleapis.com/openimages/web/index.html [July 2020]
6. Baeldung.com “What is YOLO Algorithm?” https://www.baeldung.com/cs/yolo-algorithm [24 November 2022]
7. Digikey “How to Perform Object Detection with TensorFlow Lite on Raspberry Pi” https://www.digikey.com/en/maker/projects/how-to-perform-object-detection-with-tensorflow-lite-on-raspberry-pi/b929e1519c7c43d5b2c6f89984883588#:~:text=For%20this%20tutorial%2C%20you%20can,faster%20processor%20and%20more%20memory. [19 October 2022]
8. ArcGIS Developers “How single-shot detector (SSD) works?” https://developers.arcgis.com/python/guide/how-ssd-works/#:~:text=SSD%20has%20two%20components%3A%20a,classification%20layer%20has%20been%20removed. [Date Unknown]
9. “EfficientDet: Scalable and Efficient Object Detection” https://arxiv.org/pdf/1911.09070.pdf [27 July 2020]
10. COCO “Detection Evaluation” https://cocodataset.org/#detection-eval [Date Unknown]
11. ArcGIS Developers “How Compute Accuracy For Object Detection works” https://pro.arcgis.com/en/pro-app/latest/tool-reference/image-analyst/how-compute-accuracy-for-object-detection-works.htm#:~:text=Recall%E2%80%94Recall%20is%20the%20ratio,the%20recall%20is%2075%20percent.&text=F1%20score%E2%80%94The%20F1%20score,of%20the%20precision%20and%20recall. [15 October 2020]

Appendix A: Abbreviations

Unit or Term	Abbreviation
General Purpose Input/Ouput	GPIO
Printed Circuit Board	PCB
You Only Look Once	YOLO
Common Objects in Context	COCO
Frames Per Second	FPS
Floating Point Operations per second	FLOPS
Single Shot Detector	SSD
Average Precision	AP
Average Recall	AR
Infrared	IR
Field of View	FOV
Short Message/Messaging Service	SMS

Unit or Term	Abbreviation
Microcontroller	MCU
Light Emitting Diode	LED
Passive Infrared	PIR
Volts	V
Amperes	A
Watts	W
Seconds	s
Milliseconds	ms
Domain Name System	DNS
Low Drop Out	LDO
In-System Programming	ISP
Hypertext Transfer Protocol	HTTP

Appendix B: Verification Tables

Table 4: Power Subsystem Requirements and Verification

Requirement	Verification Procedure
~~9V~~ 18 V +/-10% Power Protection circuit ensures battery does not experience sudden surges in voltage in current leading to an undervolt situation	• Plug battery configuration into the PCB. With a DMM, we will able to probe the voltage coming out of the protection circuit • Connect an LED from the output of the converter to GND for quick visual verification • Passes with changes: 18 V used rather than 9 V, but circuit does not experience sudden voltage drops and unexpected undervolt situations.
SI-8050Y buck switching regulator for 5V +/- 10% conversion and up to 8A output	• Probe circuit voltage and current at testing points • Connect an LED from the output of the converter to GND for quick visual verification • Passes: Switching regulator outputs approximately 5 V while being able to supply the system's peak current draw.
LM3940 LDO for 3.3V +/- 10% output and at up to 1A output	• Probe circuit voltage and current at testing points • Connect an LED from the output of the converter to GND for quick visual verification • Passes: 3.3 V is successfully probed in-system and successfully powers its respective sensors.

Table 5: Sensor Subsystem Requirements and Verification

Requirement	Verification Procedure
~~HC SR501 Passive Infrared Sensor makes initial detection from 5m +/- 0.5m away~~	~~● Pedestrian walking towards PIR turns on indicator LED~~ ~~● Modify sensitivity of PIR and repeat tests~~ ● Failed: We did not end up using the PIR sensor due to sensitivity of user movement. Its removal had no impact to meeting high level requirements
HC-SR04 Ultrasonic Sensor measures depth with person 30cm +/- 10cm away	● Use the ATmega32U4 Arduino Dev Kit to read sensor data ● Verify sensor detects correct depth within tolerance ● Passes: Resulting range of 3 cm to 130 cm
MakerFocus Raspberry Pi 4 Camera Night Vision Camera works in Low Light Conditions (in between last light and first light)	● Connect camera to external monitor for live video output ● User is able to distinguish most objects with live camera feed ● Test between sunset and sunrise (+/- 30 minutes) ● Passes with contingency: Was not able to work in low light conditions unless there were street lights. Issue fixed by adding infrared flashlight
Vibrating Mini Motor Disc for vibration generation that produces noticeable haptic feedback through a standard canvas backpack	● Connect motor to microcontroller ● When an object is detected within 3m away the microcontroller will supply the motor with 5V ● Verify user is able to feel vibrations through backpack ● Passes: User is able to directional feel vibrations
~~ESP8266~~ BCM4345 Wifi Module can send emergency message to Phone within 5 minutes	● Independently Power using breadboard setup ● Determine programmability to ping a webserver to send a text message to any preset phone number ● Test functionality with AtmegaA32U4 microcontroller ● Test integration with entire system ● Passes with Changes: As long as there is signal, emergency contact receives SMS message. We used a different wifi module.

Table 6: Control Subsystem Requirements and Verification

Requirement	Verification Procedure
Image Processing Algorithm identifies people with the similar accuracy as the ~~YOLO~~ EfficientDet0 algorithm +/- 5% at a rate of 4-6 frames per second	• With a stationary camera, perform object detection on objects stationary objects • Algorithm should be able to determine which objects are people and which objects are non-people (not relevant whether it can properly identify what non-people objects are) • Camera is stationary, objects are moving at average walking speed (2.5 - 4 mph) • Algorithm should be able to determine which objects are people • Camera is on a moving platform to simulate walking, objects are moving toward the camera at average walking speed • Passes with Design Changes: We ended up using the EfficientDet0 rather than YOLO; otherwise, specification is met.
Microcontroller sends correct I/O signals to each subsystem within 1 second	• Flash the MCU with a program that drives pins to high and measure output with a DMM as well as have LED indicators • Verify PWM signal pins with an oscilloscope • Verify that each sensor is able to work with the same functionality on the dev kit and custom PCB • Passes with Design Changes: We opted for I2C communication instead for its parallel capabilities; otherwise, there is minimal delay, on the order of microseconds.

Appendix C: Circuit Schematics

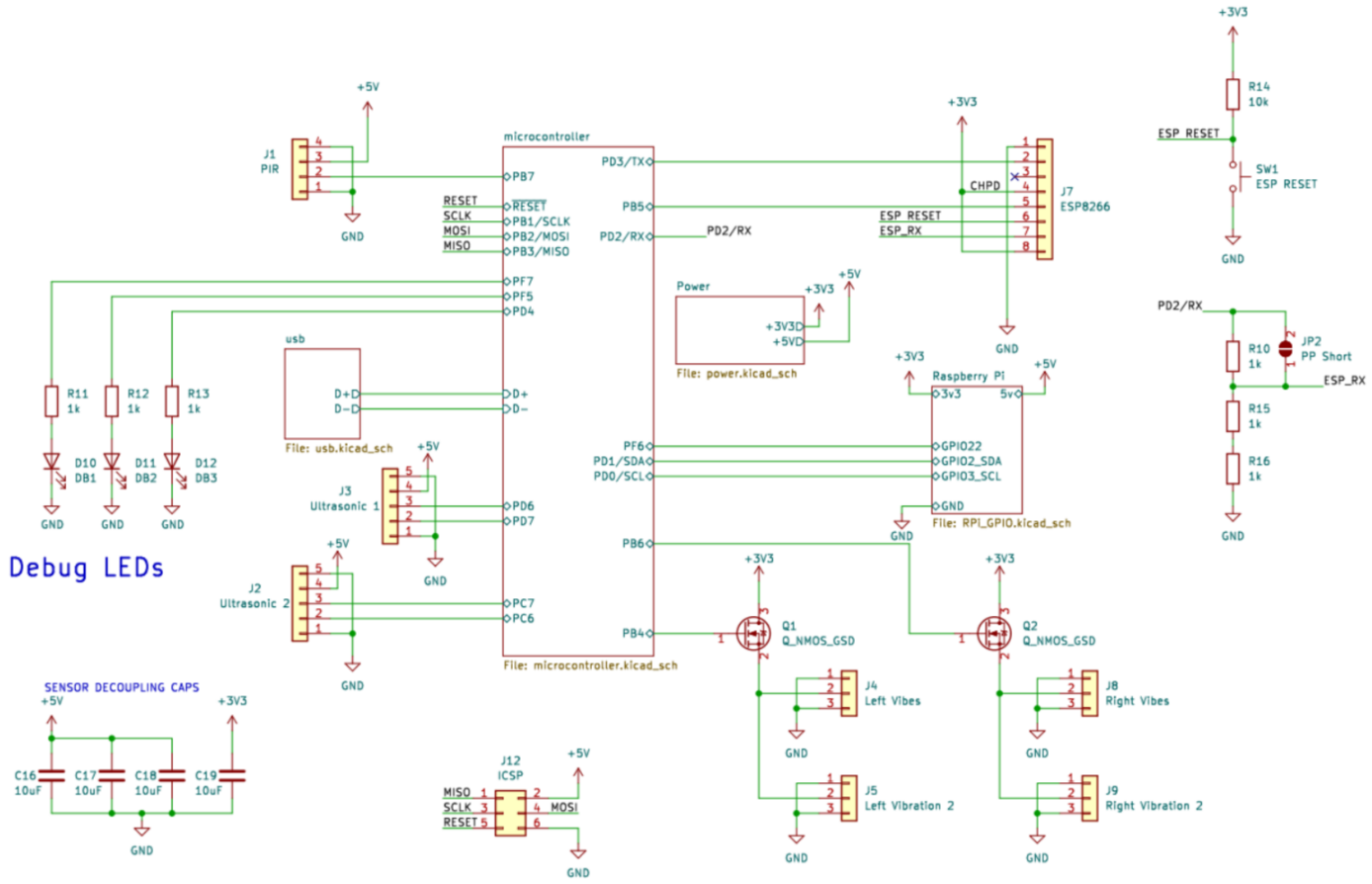


Figure 6.1 Main Schematic Capture

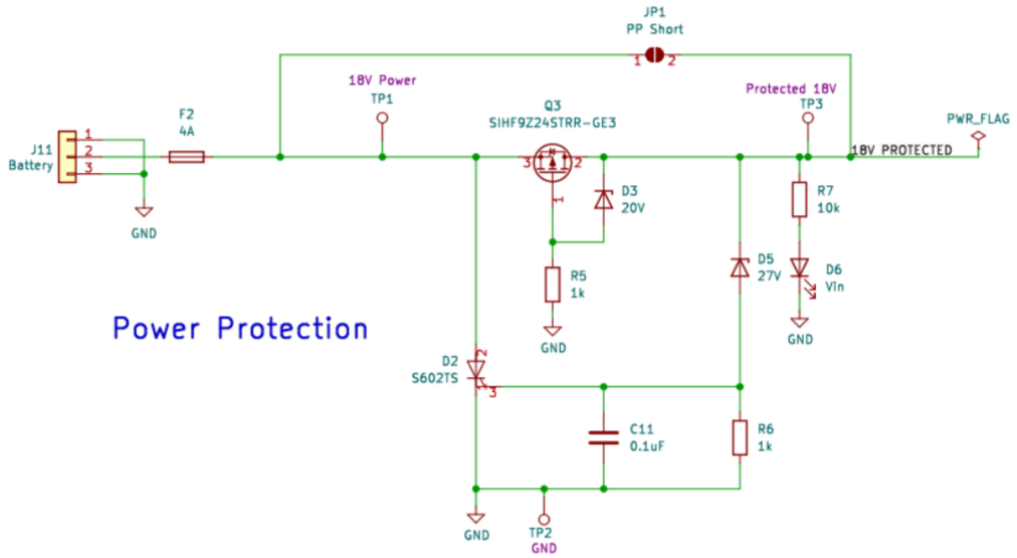


Figure 6.2: Power Protection Circuit

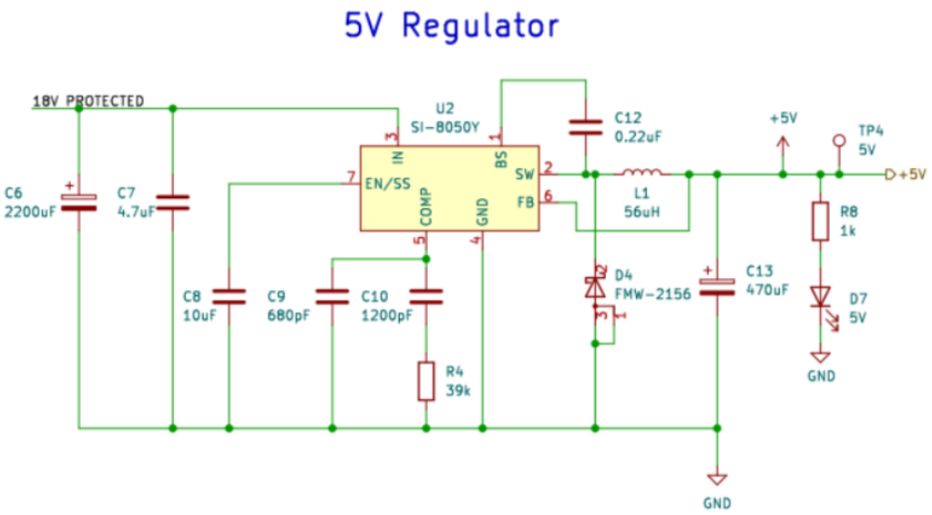


Figure 6.3: 5V Switching Regulator Schematic

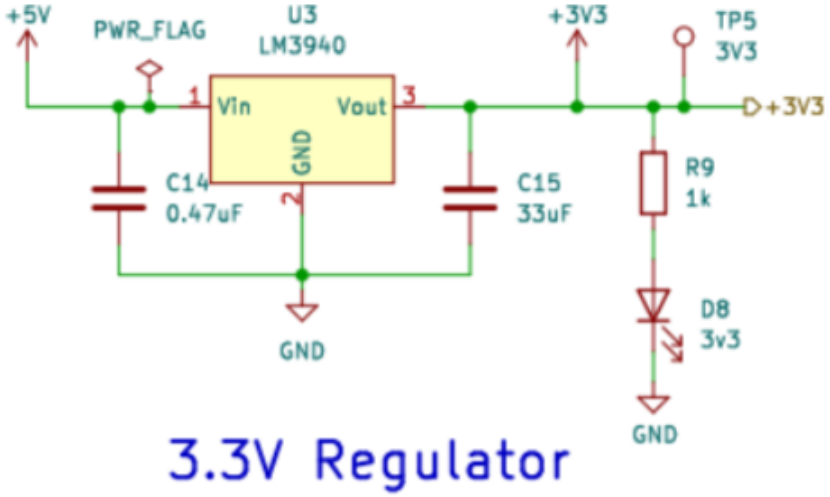


Figure 6.4: 3.3V Linear Regulator

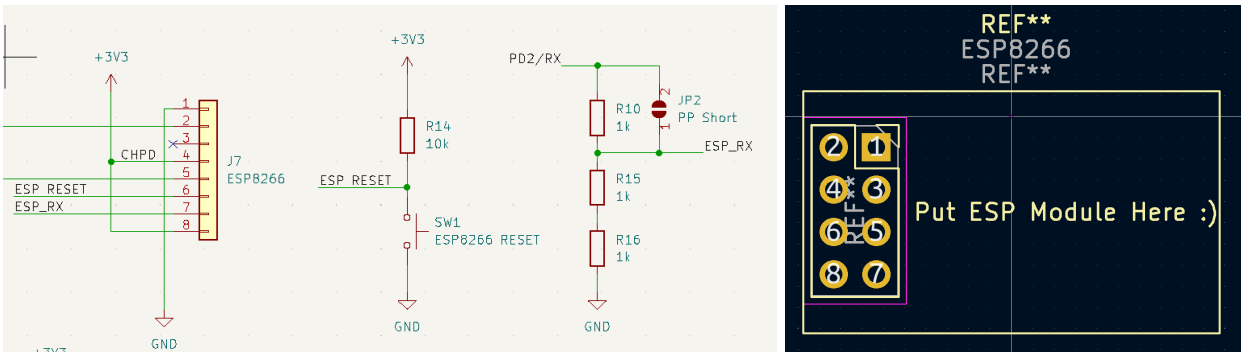


Figure 6.5: ESP8266 Schematic and Footprint

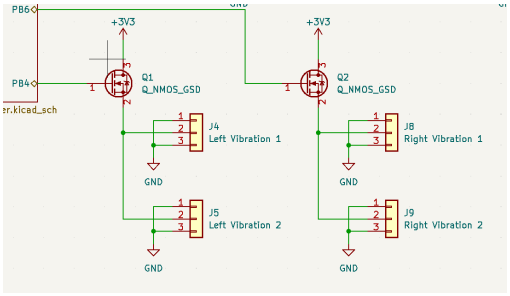


Figure 6.6: Vibration Actuator Array

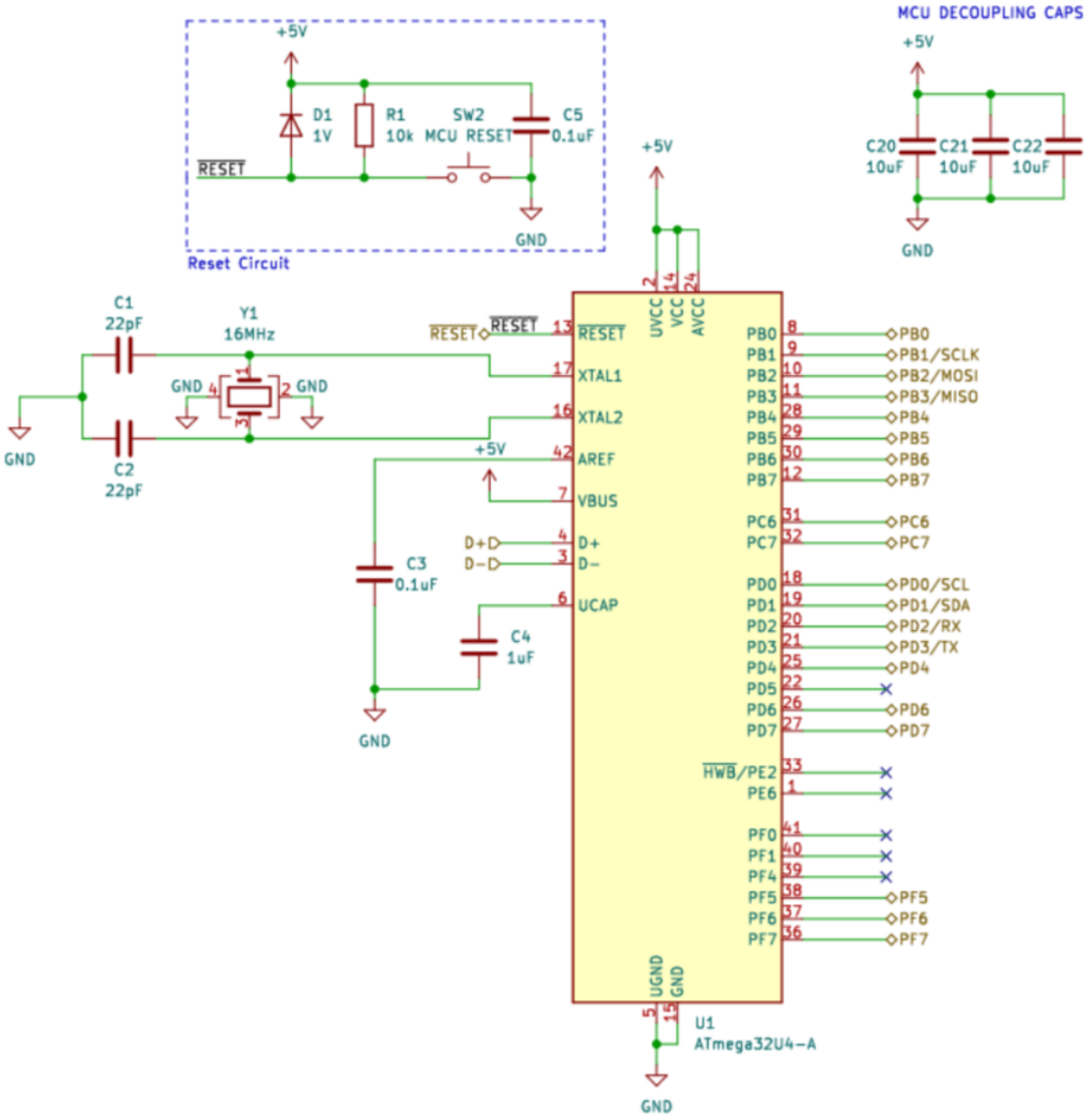


Figure 6.7: Microcontroller Schematic

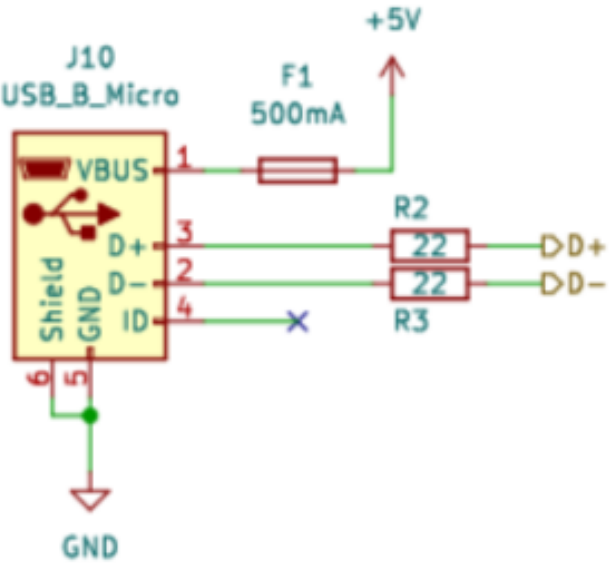


Figure 6.8 Micro Usb Programmer Schematic

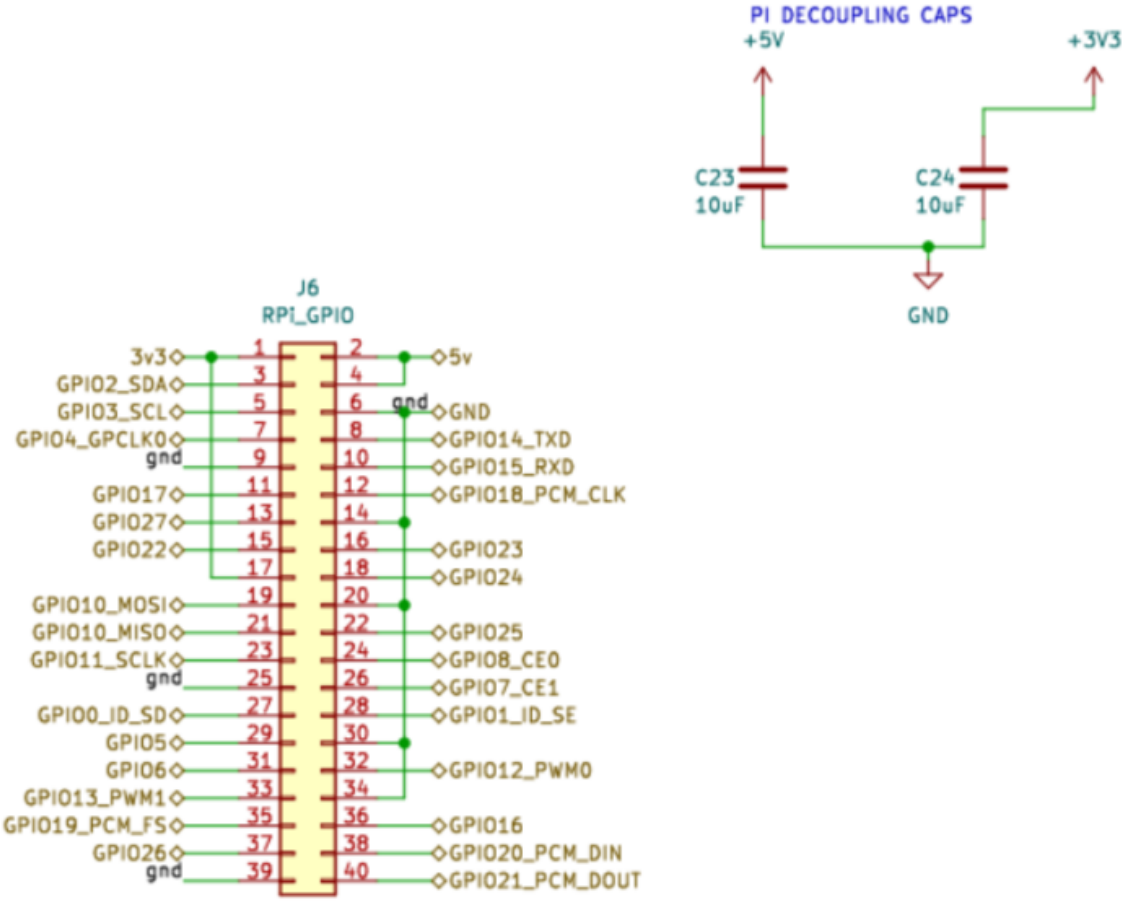


Figure 6.9 Raspberry Pi Connector Schematic