Final Report for ECE 445, Senior Design, Spring 2019

Cat Collar

Sponsored by Petronics
Team 42
TA: Dongwei Shi

Taha Anwar
Ching Chieh Yang
Junnun Safoan

Abstract

This document outlines the design and implementation of a motion-based collar for cats, along with the challenges and results of the project. The project was designed for Petronics, a company based in Research Park, UIUC, to allow them to monitor the cat's activity and measure the engagement between the cat and their product Mousr. The final product was able to collect accelerometer data from the collar using the IMU, and transmit it wirelessly to a Raspberry Pi or a remote desktop for data analytics.

TABLE OF CONTENTS

1. Introduction

1.1 Objective

Problem

Cat owners do not always have time to play with their cats. The Mousr is a clever solution that accompanies your cat in your absence. However, Petronics still does not have a way to check if there is a direct correlation between a cat's overall activity and Mousr's activity.

Solution

The Mousr unit developed by Petronics is already able to make event predictions such as "inactive", "engaged", "needs charging" and so on. The motivation for the cat collar is to use the data collected from it to confirm the event predictions by the Mousr in order to assess the cat's actual engagement with Mousr. This will be instrumental for Petronics, as it will allow them to measure the effectiveness of Mousr in engaging with the cat.

1.2 Background

This project is sponsored by Petronics, a company aiming to build interactive robots for pets to engage with them in the absence of humans. They have already developed a prototype called Mousr, that is able to play with the pet for upto 40 minutes continuously. However, they don't have a way to show any

correlation between the activity of the cat and Mousr in operation. So they have asked tasked us to develop a collar that monitors the activities of the cat and reports it wirelessly to a remote desktop or Raspberry Pi for data analytics. This solution will play a crucial role for Petronics in their ability to validate Mousr and market it to the users using the statistics found from our dataset.

1.3 High-level requirements list

1. Must be able to transmit and interpret IMU sensor data to Raspberry Pi wirelessly using the ESP32.
2. The entire PCB circuit should be small enough to be integrated within a traditional collar.
3. The battery lifetime should be at least 1 hour to enable collection of sufficient data for a play session.

2. Design

2.1 Block Diagram

The cat's activities will be measured by the IMU sensor. The time-stamped acceleration data will be sent to the ESP32 by the I2C protocol. The IMU data will be sent through WIFI using socket IO protocol to the Raspberry Pi's python program, where it will be formatted into a csv file. An algorithm will use csv files to detect the cat's activities. It will attempt to detect cat's engagement activity in real time and turn on the camera. Below in figure 1 we have a high level block diagram explaining the communication between different modules in our design. We also have a physical layout of how everything will look while testing as shown below in figure 2.

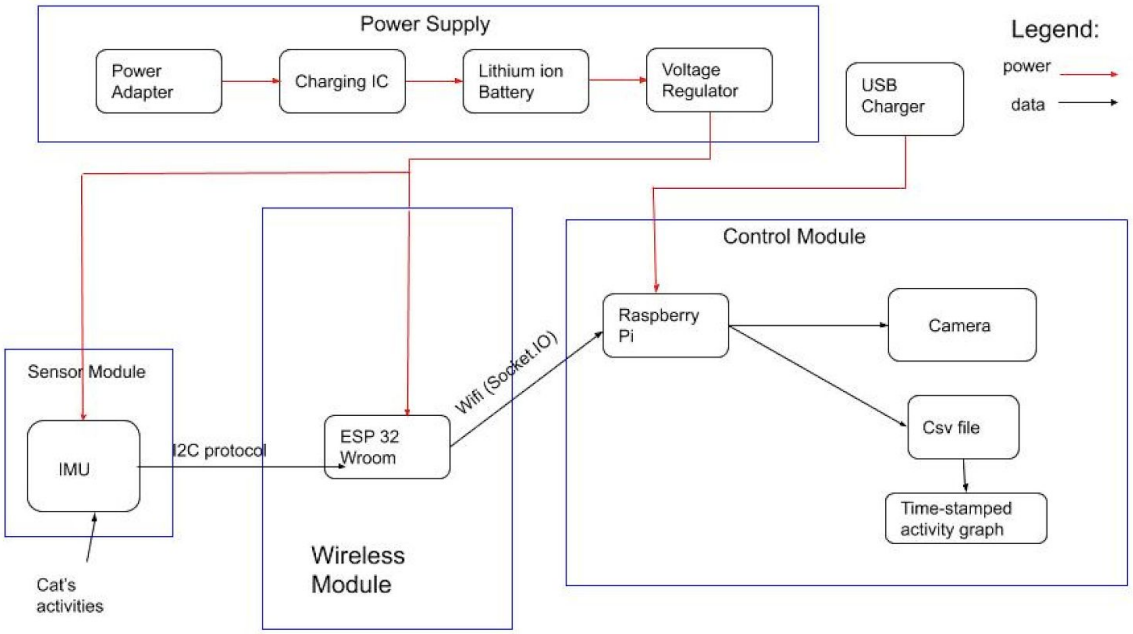

Fig. 1. High level block diagram

2.2 PCB Design

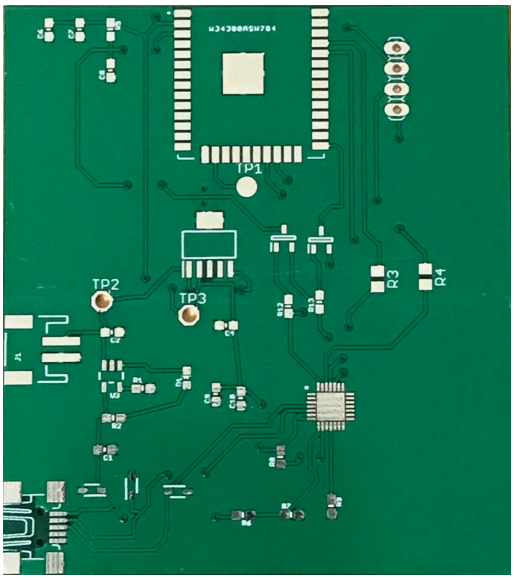

Fig. 2. PCB Board Layout

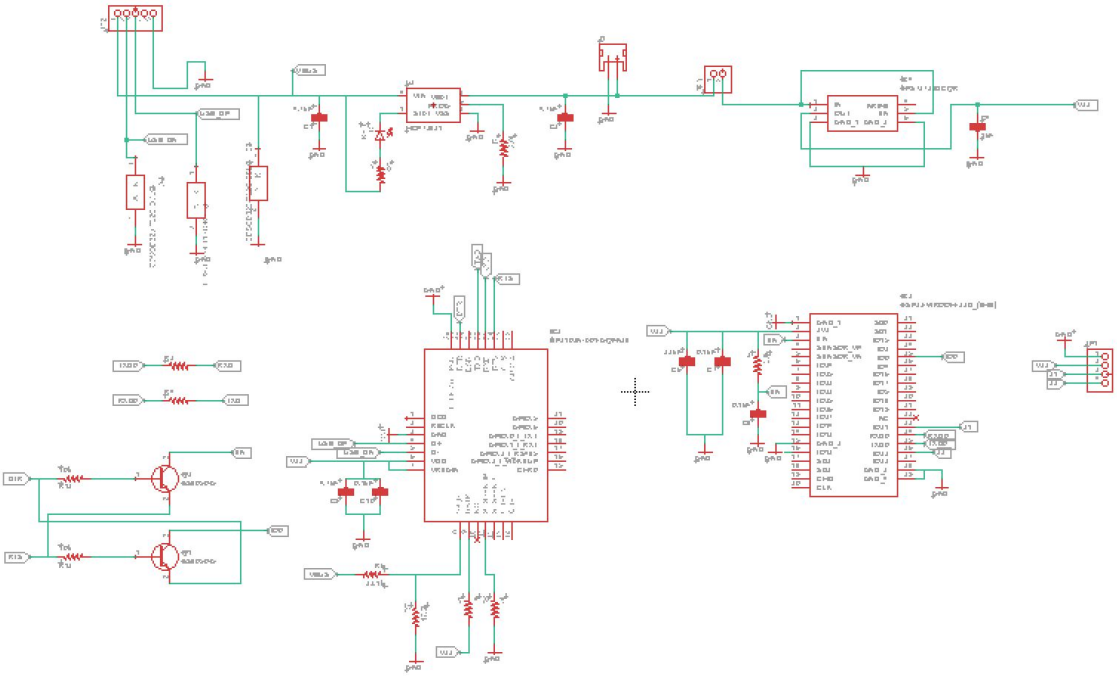

Fig. 3. PCB Schematic Layout

2.3 Physical Layout

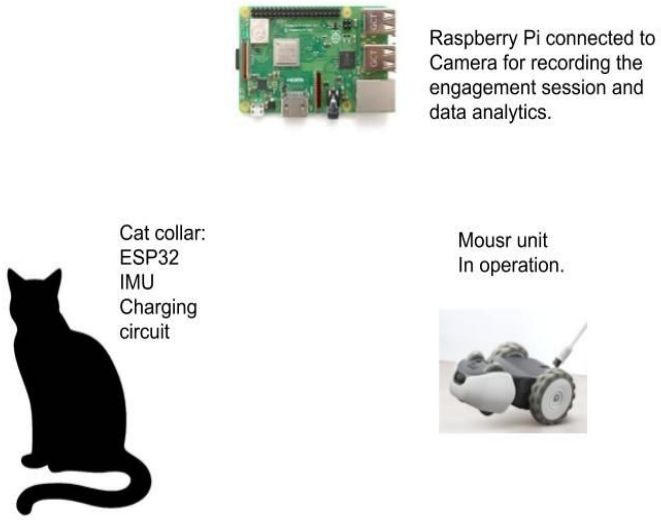

Fig. 4. Physical Layout

2.4 Power Supply

2.4.1 Charging Circuit

We will be using a the Sparkfun 3.7 V, 400mAh Lithium Ion Battery in order to power our entire collar circuit [3] . The main components it will need to power are the ESP32 and the IMU. We are estimating ~180 mA of current draw from the ESP and ~6mA from the IMU. Rounding that to 200mA, we see that this will ideally put as at 2 hours of constant usage.
We will be using the Sparkfun PRT-10217 Lipo charger. The charger charges 3.7V LiPo cells at a maximum rate of 500mA and includes a micro-USB connector, a charging IC, status LEDs, and an appropriate port to connect to the lithium ion battery. We will initially be using this board but we can implement the board on our PCB afterwards. The schematic for its implementation is shown in figure 5 below.

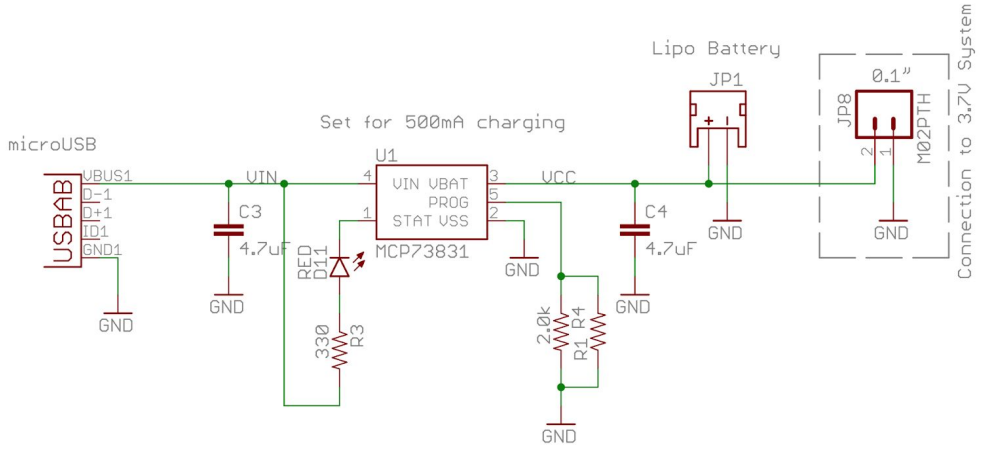

Fig. 5. Battery Charging Circuit Schematic [3]

2.4.2 Voltage Regulator

Our voltage regulator should take our Lithium Batteries as an input and output around 3.3 V to the ESP32 and IMU. We plan on using the TLV755P Voltage regulator IC Chip in order to do this. The chip can take input voltage of 1.45 to 5.5 V and provides can provide an output voltage of 0.6 to 5 V with a low 238 mV dropout voltage [6]. The regulator can also source up to 500 mA of current which should be sufficient for our purposes.

2.5 Sensor Unit

The cat's activities will be measured by the IMU sensor. The time-stamped acceleration data will be sent to the ESP32 by the I2C protocol. The IMU has 4 ranges of acceleration to select from: *±2/±4/±8/±16 g linear acceleration [4],* where each has a different sensitivity, with a larger scale having greater sensitivity. Since a cat is not expect to have a linear acceleration exceeding +-2g, even in jumping from free fall, we will choose +-2g as our scale. The default frequency of the IMU measures acceleration every 250 ms. It is possible to set the frequency even higher, but doing so produces tremendous lagging and lost data.

2.6 Control Module

2.6.1 Raspberry Pi

This part of the control module contains the Raspberry pi and a Picamera. The raspberry pi receives the IMU data from the ESP32 Wifi signals. Specifically, a python program with Socket IO receives the acceleration time-stamped data. We will be using the Picamera to record the entire play session to serve as ground truth for our algorithm.

Cat's sensor activities is inputted to our analysis program. This software stores the time-stamped data in a csv file. It also runs an algorithm at the end of each session to determine when the cat is in engagement. At the end of each session, which lasts approximately an hour, an bar graph of the cat's activity against time will be generated. Further details of the graph is explained in the software module.

2.6.2 ESP 32 Microcontroller

The ESP32 microcontroller will be programmed to receive the IMU sensor signals and transmit the data wireless to the raspberry pi. The communication between IMU and ESP32 is programmed using Arduino code with I2C protocol. I2C protocol is used instead of SPI protocol because it is the default settings on the LSM9DS1 IMU and I2C only requires two pins (SDA data pin and SCA clock pin) versus the four pins of SPI. Once the acceleration data stream arrives on the ESP32, it is immediately sent wirelessly through WiFi to the Raspberry Pi, which is implemented with Socket IO. The ESP32 transmits through wifi 802.11g OFDM at 54 mbps at 190mA power consumption [8].

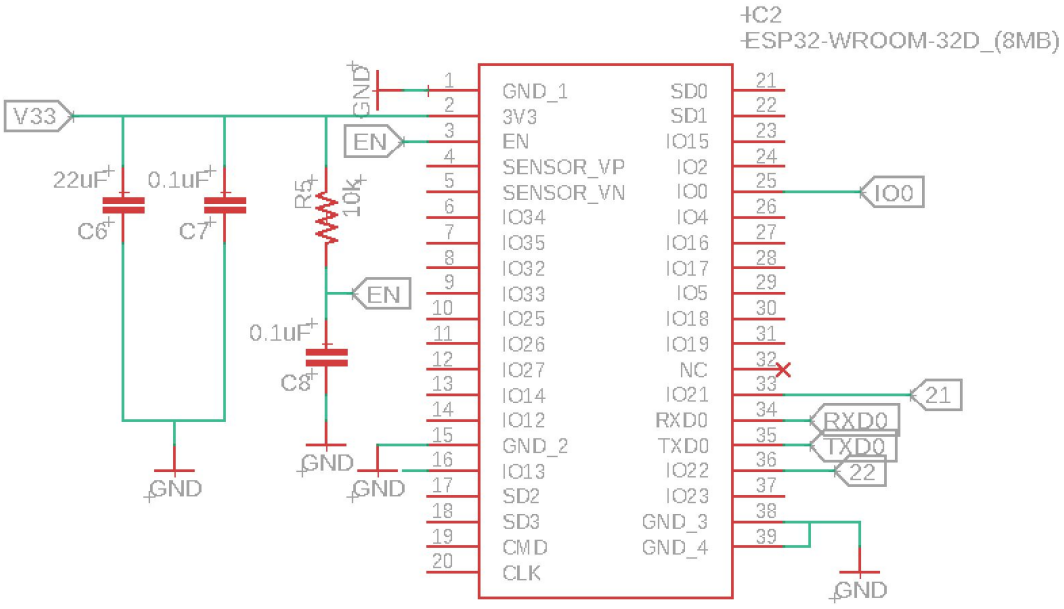

Fig. 6. ESP32 Wroom Schematic

2.7 Software Module

The software module can be broken down into three sections: Transmitting data from IMU to ESP32 via I2C protocol, connecting the ESP32 to Wifi and sending the data via Socket.IO, and finally the algorithm for detecting the cat's activity based on the accelerometer data.

A python program on the Raspberry Pi will be responsible for receiving real time data from the ESP32. This program can be run anytime, as it will keep polling for signals from the ESP32. As soon as the program is run, the Pi camera also turns on for ground truth. After receiving the first batch of acceleration data, the python program runs for an hour and shuts down along with the camera due to our battery life of one hour. Another program will take in the IMU data csv file and detect cat activities.

Once the Raspberry Pi has received the accelerometer data from the wireless module, it runs a Python program to plot the acceleration graphs along x, y and z axes as shown in Figure X. Finally it runs a activity detection program, which is essentially calculating the energy from the acceleration and binning it into one of three energy states, which are categorized as walking, playing and resting as specified in the software flow chart in Figure 7. The bar graph generated is our final output.

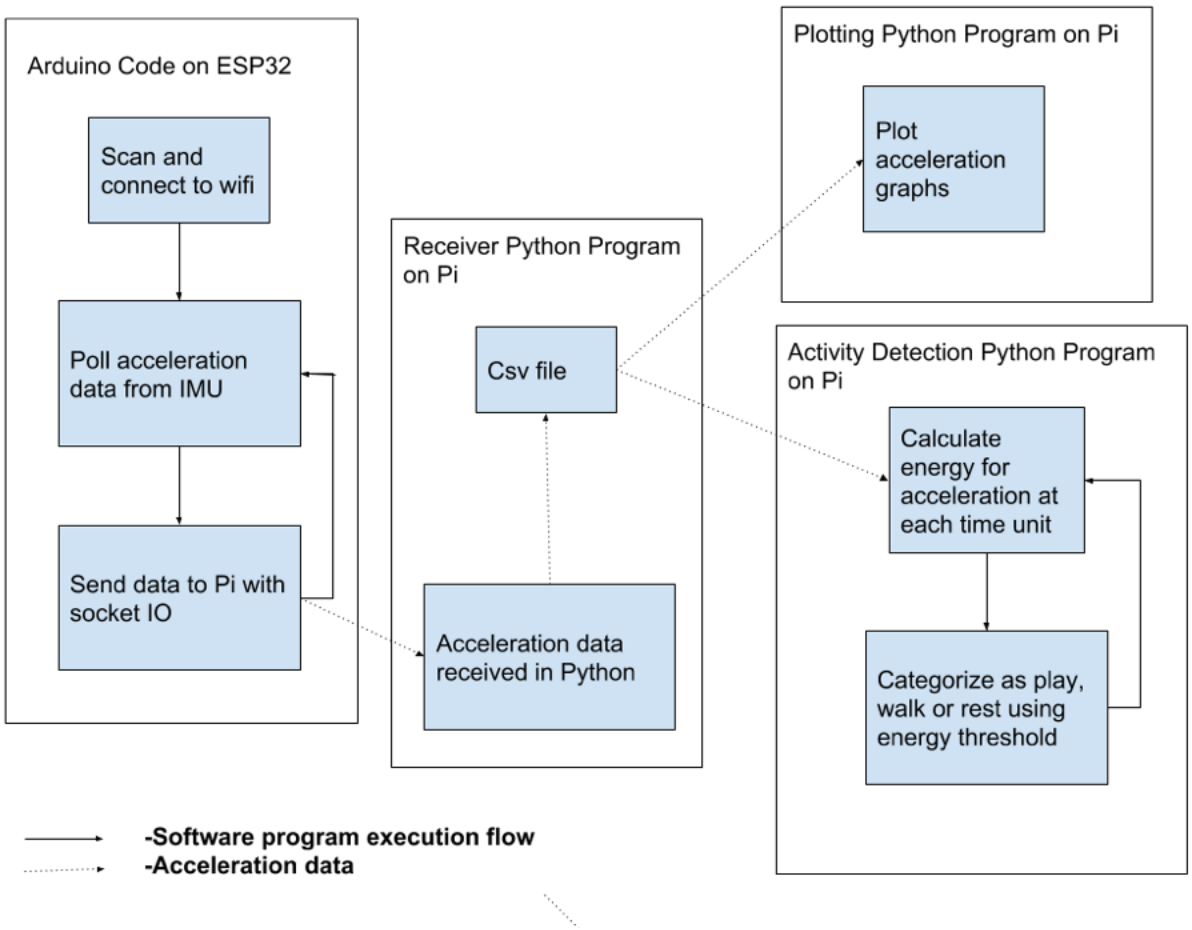

Fig. 7. Software Module Flowchart

3. Design Verification

3.1 Power Supply

3.1.1 Charging Circuit

We were successfully able to meet the requirements that we set for our charging circuit. We were able to charge our lithium ion battery in about an hour. We also were able to successfully supply the power from the battery to the rest of our circuit. Our circuit was able to run for one hour and ten minutes using the battery which satisfied the requirements that we set.

3.1.2 Voltage Regulator

The voltage regulator functioned as we had planned. We were successfully able to take the 3.7 V input and regulate it to 3.3 V to supply our ESP32 and IMU.

3.2. Control Unit

3.2.1 Raspberry Pi

We were able to connect the Raspberry Pi to the same Wifi network as the ESP32 and receive the IMU data, which was displayed in the terminal output. As we moved our PCB in different orientations, the changes were reflected in the Raspberry Pi terminal. The IMU data reported 1g m/s^2 along the three axes when it was placed in different orientation, for e.g 1g in x-axis when it was tilted sideways.

3.2.2 ESP32

In order to verify the functionality of the ESP32 on our PCB, we first tried to program a simple Wifi scan. After verifying that the ESP32 was working as expected, we then programed our IMU data acquisition code. After fixing some minor mistakes, we were able to successfully develop communication between our IMU sensor and the ESP32 as desired.

3.3 Software

3.3.1 ESP to Raspberry Pi

We were able to successfully transmit the IMU data from the ESP32 to Raspberry PI. The collected accelerometer data is then plotted out as shown in Figure 8 below. The graphs are acceleration in x, y, z and all three together. The curve shown is generated by swinging the IMU for roughly 70 seconds. The PCB is first moved up and down, generating the first set of spikes near the 30 seconds mark and then side to side around the 50 seconds mark, as shown in Figure 8.

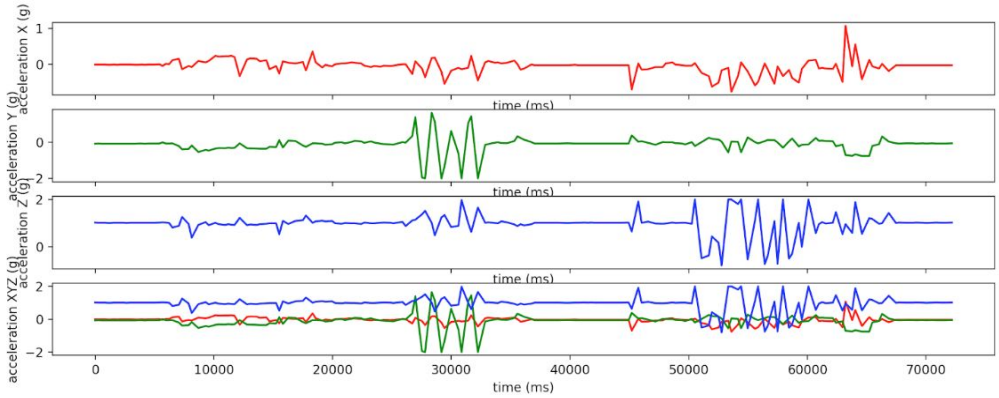

Fig. 8. IMU Acceleration Data

3.3.2 Activity Detection

The energy equations can be represented in two versions. Equation 2 below shows the formula for energy given by the summation of the acceleration at the current iteration and the previous energy value, where each portion is weight by λ. The weight has to sum up to 1 so each weight is λ or (1- λ).

A higher λ means more weight given to acceleration and vice versa. Equation 1 below is a general form of the same formula that takes into account the exact time increment of each measurement. Specifically, the equation calculates E at time t by taking the last seen energy at

t-c, where c is a positive constant representing the time increment period for the IMU sampling rate.

$$E(t) = \lambda * a(t) + (1-\lambda) * E(t-c) \qquad (1)$$

$$E_i = \lambda * a(t) + (1-\lambda) * E_{i-1} \qquad (2)$$

Where λ is a lambda value between 0 and 1
a(t) is acceleration of the cat
E_i is energy of the cat at the ith time iteration (period is 250 ms)
E(t) is the energy of the cat at time t
C is time increment per sample
In our trials of hand movements, we optimized the lambda constant to be around 0.2

The energy function generates the plot as shown below in figure 9.
E < threshold_1 outputs blue meaning low energy
threshold_1 < E < threshold_2 outputs green meaning medium energy
E > threshold_2 outputs yellow meaning high energy
Where threshold_1 < threshold_2.
In our trials of hand movement, we optimized threshold_1 = 1.05 and threshold = 1.3

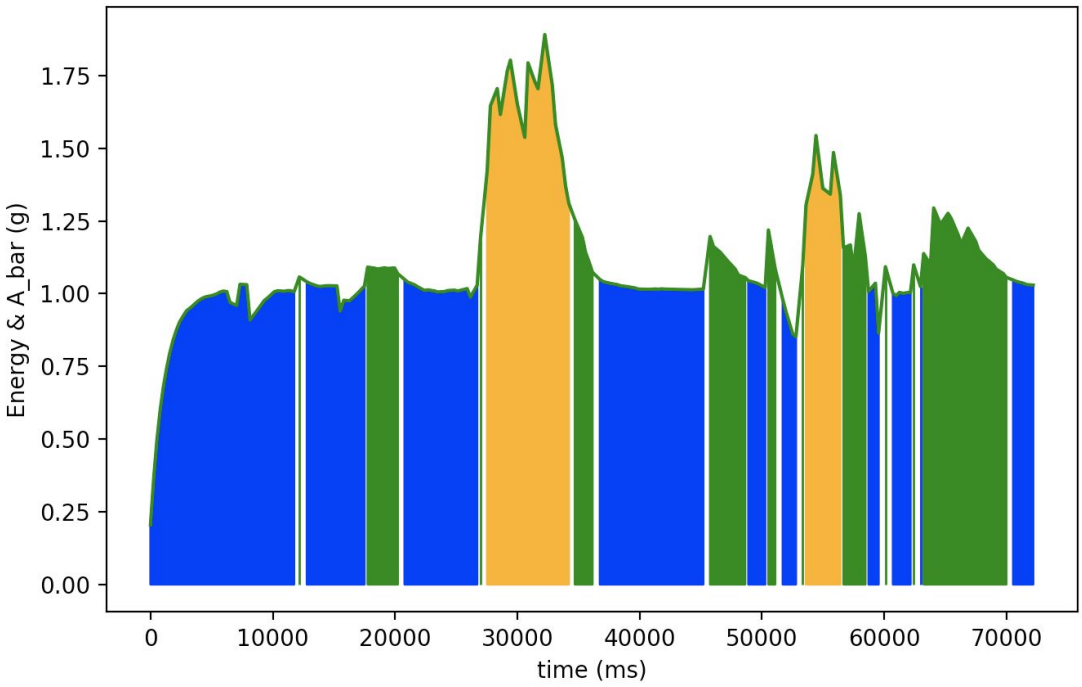

Fig. 9. Energy Detection vs. Time

The two graph comparison below in figure 10 demonstrates the effect of λ in the energy formula. λ is a variable value between 0 and 1 and is optimized with experimentations with actual cat data. A value closer to 0 produces a smoother curve while a value closer to 1 produces a spikier curve. This is because the value of x corresponds to how much weight is given to previous data versus acceleration in the current iteration. As shown below, the red curve is a(t) and green curve is e(t). On the left side, λ = 0.1, which has a smoother graph than the right side where λ = 0.2. λ = 0.2 is the final value that we settled with.

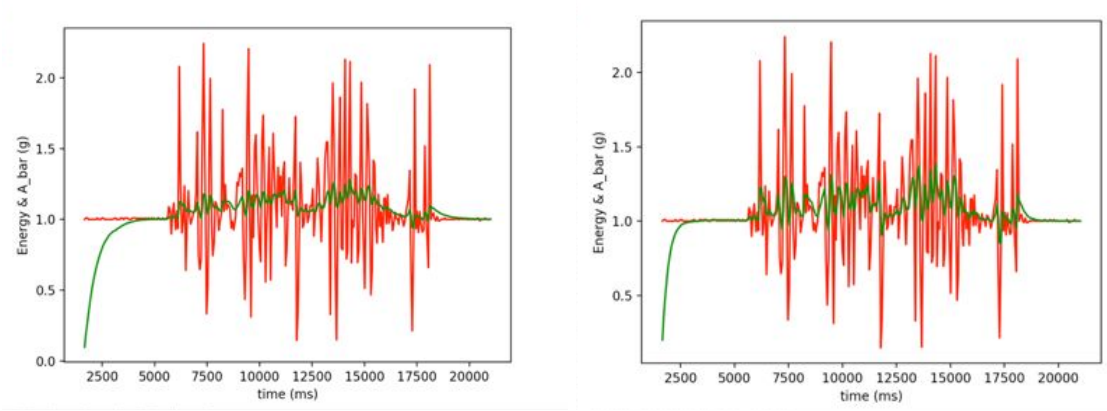

Fig. 10. Effect of λ Value on Energy Plot

4. Cost and Schedule

4.1 Cost Analysis

4.1.1 Labor

According to Engineering Report 2016-2017, reported by the Engineering Career Services(ECS), the average salary for an Engineering graduate was $75,450. So, on a regular work schedule of 40 hours per week and 52 weeks a year, it sums to a total of 2080 hours. The hourly labor rate is then approximately $36 for each of the three members of our Senior Design team. We are expecting to work around 10 hours per week for 10 weeks, so the total cost is as follows:

$36 x 10 hrs/week x 10 weeks x 3 members = $10800

4.1.2 Parts

Description	**Manufacturer**	**Part #**	**Quantity**	**Cost**
ESP32 Dev board	Expressif	DevKitC V4	1	$10.64
IMU	Sparkfun	LSM9DS1	1	$15.95
Lipo Charger	Sparkfun	PRT-10217	1	$8.95
Lithium Ion Battery	Sparkfun	PRT-1385	1	$4.95
USB Charger	Dericam	B07D3QSXQJ	1	$7.99
Lipo Charging IC	Mouser Electronics	MCP73831	1	$0.57
Voltage Regulator IC	Texas Instruments	TLV755P	1	$0.62
ESP32 Module Chip	Expressif	Wroom 32	1	$3.80
Multiple Passive Components	Digikey/Mouser	Resistors, Capacitors, etc.	20 each	$30.26

Total				$83.73

Table. 1. Cost of Parts

4.1.3 Grand Total

$10,800 + $53.47 = $ 10,853.47.

4.2 Schedule

Week	**Task**	**Person**
02/25/19	1) Experiment with IMU data, measure sensor bias to ensure IMU is working correctly 2) Set up raspberry pi	1) Junnun, Jeff 2) Taha
03/04/19	1) Order battery circuit, voltage regulator, and pi camera 2) Go through the tutorial for setting up ESP32 and interfacing with it.	1) Taha 2) Jeff, Junnun

03/11/19	1) Store IMU data into the SD card of ESP32 2) Implement communication (I2C protocol) between IMU and ESP32 dev board 3) Test/Verify battery circuit, voltage regulator and charging circuit 4) Setup pi camera with raspberry pi 5) Create Eagle schematic and order first round PCB	1) Jeff 2) Junnun 3) Taha, Junnun 4) Taha 5) Taha
03/18/19	1) Interpret and Process IMU data from ESP32 on the Raspberry Pi 2) Prototype final circuit 3) Order the components/parts needed for the PCB	1) Jeff, Junnun 2) All 3) Taha
03/25/19	1) Establish communication between Mousr and raspberry pi 2) Solder PCB	1) Junnun, Jeff 2) Taha
04/01/19	1) Develop algorithms for physical activity detection from accelerometer data. 2) Write code for data analytics of cat's engagement by comparing with data from Mousr unit. 3) Order second round PCB	1) Junnun 2) Jeff 3) Taha
04/08/19	1) Solder and debug new PCB	1) All
04/15/19	1) Perform mock demo 2) Finalize final working PCB	1) All 2) All
04/22/19	1) Final demonstration 2) Work on final report	1) All 2) All

Table. 2. Schedule

5. Conclusion

5.1 Accomplishments

We accomplished all of the functionalities as defined in our initial planning and requirements. The ESP32 chip successfully receives IMU sensor data and transmits them to the Raspberry Pi wirelessly. The PCB board was designed small enough to be integrated on a cat collar and mobile movements. Lastly, the PCB circuit can last up to 60 minutes of constant usage for every session.

5.2 Uncertainties

One uncertainty with our project is Wifi connectivity issues for data transmission. Our ESP32 and Raspberry Pi program needs to connect to the same WiFi signal for data transmission but can only connect to mobile hotspots or wifi signals with simple setup. We were not able to connect to wifi that requires authentication like IlliniNet. This creates a problem when the circuit is used inside buildings with weak mobile hotspot signals.

5.3 Future Considerations

One of the future improvements to add to the PCB collar is making the PCB board even more compact, so that it can be worn comfortably on a cat. As shown by the picture of our PCB, there are alot of unused spaces on the edges of the board and many components can be squeezed closer together. In addition, instead of using IMU breakout board LSM9DS1 and USB board, we can integrate the IMU chip and USB port into the PCB. These considerations were left out of our demo due to the difficulty in soldering components and challenges in debugging such circuit.

Another next step is to put the pcb in an enclosed protective case and test it on an actual cat. This will give us useful acceleration data to generate an accurate energy formula. As mentioned above, the λ constant value in the energy formula requires multiple experimentation with the cat for optimization. An important tradeoff is the frequency of IMU measurement. A faster frequency produces more accuracy but sinks more power; therefore it would be useful to observe the optimal time frequency in usage cases.

5.4 Ethics and Safety

One of the safety concerns in our project is battery's temperature. Misuse of the batteries can cause to overheating and burning the circuit. For most cells, charging significantly above 100% state of charge (SOC) can lead to rapid, exothermic degradation of the electrodes. Charging above the manufacturer's high voltage specification is referred to as overcharge [2]. A few indications of these hazards can be noticed.If a typical fully charged (or overcharged) Li-ion cell undergoes a thermal runaway reaction, a number of things occur, including [2]:

- Cell internal temperature increases;
- Cell internal pressure increases;
- Cell undergoes venting;
- Cell vent gases may ignite;
- Cell contents may be ejected; and
- Cell thermal runaway may propagate to adjacent cells.

In order to prevent this, we plan to use a thermistor to avoid overheating and make sure to use full batteries during demos. It is also to know the root causes of hazards and prevent them. Generally, the root causes of energetic cell and battery failures can be classified as [2]:

- Thermal abuse (e.g., external heating);
- Mechanical abuse (e.g., denting, dropping);
- Electrical abuse (e.g., overcharge, external short circuit, over discharge);
- Poor cell electrochemical design (e.g., imbalance between positive and negative electrodes); and
- Internal cell faults associated with cell manufacturing defects (e.g., foreign metallic particles, poor electrode alignment).

Another issue is using cats for experiment. Our project itself does not pose great danger on the cat, since it is only a cat collar. However, we will take extra measures to prevent battery related hazards as mentioned above or any physical discomfort for the cat when wearing the collar.

References

[1] Ieee.org, "IEEE Code of Ethics", 2016 [Online]. Available: www.ieee.org/about/corporate/governance/p7-8.html

[2] SFPE, “Lithium-Ion Battery Hazards”, 2012. [Online]. Available: www.sfpe.org/page/2012_Q4_2/LithiumIon-Battery-Hazards.htm

[3] SparkFun Electronics, “SparkFun LiPo Charger Basic - Micro-USB.” 2015. [Online]. Available: www.sparkfun.com/products/10217?_ga=2.229262751.2001694764.1550419447-1820395587.1543635422.

[4] Sparkfun Electronics, “LSM9DS1 Datasheet”, 2015. [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/3/7/3/LSM9DS1_Datasheet.pdf

[5] UIUC College of Engineering, “Engineering Report 2016-17”, 2017. [Online]. Available: https://ecs.engineering.illinois.edu/files/2018/03/Engineering_Report_2016-2017_FINAL.pdf

[6] Texas Instruments, “TLV755P Datasheet”, 2018 [Online]. Available: http://www.ti.com/lit/ds/symlink/tlv755p.pdf?HQS=TI-null-null-mousermode-df-pf-null-wwe&DCM=yes&ref_url=https%3A%2F%2Fwww.mouser.com%2Fcard

[7] GitHub. (2019). *espressif/esp-idf*, 2019 [Online] Available at: https://github.com/espressif/esp-idf/tree/master/examples/storage/sd_card
https://randomnerdtutorials.com/esp32-data-logging-temperature-to-microsd-card/

[8] Espressif, “ESP32 WROOM Datasheet”, 2018 [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[9] Resna.org, “Simple Activity Recognition”, 2015 [Online]. Available: https://www.resna.org/sites/default/files/conference/2015/other/gani.html

Appendix A

Requirement and Verification Table

Requirements:	*Verification:*
1) *Battery must supply at least 3.7V for a minimum of 1 hour* 2) *Battery must supply at least 180 mA to the ESP32 and 6.1mA to the IMU for a minimum of 1 hour*	1) a) *Connect fully charged lithium Ion battery as the battery shown in figure 3 circuit below* b) *Discharge battery at 400mA for 1 hour (Use R_1 = 1.25/(400mA) = 3.125 Ω)* c) *Use multimeter to verify that the battery voltage remains above 3.7 V.* 2) a) *Connect the lithium ion battery as the “battery” in the constant current test circuit (fig. 3) and draw 200 mA* b) *Measure the output current using a multimeter and ensure that there is sufficient current being supplied for 1 hour.*

Table. 3. Lithium Ion Battery R&V

Requirements:	*Verification:*
1) *Must be able to fully charge the lithium battery to 3.7 within 1-2 hours*	1) *Once the battery has depleted, begin charging and monitor how long it takes for the battery to become fully charged*

Table. 4. Charging Circuit R&V

Requirements:	*Verification:*
1) *Output a voltage of 3.3V (+/- 5%) to the ESP32 and IMU* 2) *Must be able to output the necessary current to the ESP and IMU (~200mA)*	*1&2)* a) *Connect the output of the voltage regulator as the "battery" in the constant current test circuit (fig. 3) and draw 200 mA* b) *Measure the output using an oscilloscope to ensure that the output voltage is +/- 5% of 3.3 V*

Table. 5. Voltage Regulator R&V

Requirements:	*Verification:*
1) *Must be able to send time-stamped ESP32 data to Raspberry Pi wirelessly using MQTT protocol* 2) *Must be able to store the sensor data to SD card*	1) *We will simply start by printing an acknowledgement message from the Raspberry Pi, once it has received the data packet via MQTT protocol. Then move on to sending the time-stamped sensor data.* 2) *Output the csv file's acceleration data into a graph using excel and compare that with our actual IMU motion: stationary is 0 m/s^2 and free fall is 9.8 m/s^2*

Table. 6. ESP32 R&V

Requirements:	*Verification:*
1) *Must be able to send information to ESP32.*	1) *We will keep the circuit stationary on a flat surface, so that it reports only an acceleration of g m/s^2 in the z-axis and confirm that is reflected in the output values of the ESP32, which is stored in a CSV file in the SD card.*

Table. 7. IMU R&V

Requirements:	*Verification:*
1) *Must be able to communicate with ESP32 through bluetooth or wifi*	1) *The ESP32 transmits through wifi 802.11g OFDM. Start with sending a simple string message from ESP to Pi. Then timestamped data at at 54 mbps. [8]*

Table. 8. Raspberry Pi R&V

Requirements:	*Verification:*
1) *Must generate timestamps of IMU sensor data and mousr activity in a csv file* 2) *Must detect cat's engagement with mousr with 75% accuracy* 3) *Must turn on accessory camera when cat's engagement is detected*	1) *Check the csv file to verify that the output contains rows of data, where the columns represent time, acceleration data from IMU and mousr activity.* 2) *We will check our final prediction result against ground truth from our camera. Perform the experiment 20* times to obtain a percentage accuracy 3) We will check whether the camera turns on when the cat is engaging. This will be performed as many times as possible.

Table. 9. Software R&V