# trailTracer

Rev. 6

Submitted in partial fulfillment of the requirements of Embedded Senior Project (CST 472)

| Rev. | Date | Notes |
|---|---|---|
| 1 | 5/13/2019 | Initial Release |
| 2 | 6/07/2019 | Spring term final revision |
| 3 | 11/18/2019 | Fall 2019 revision |
| 4 | 12/4/2019 | Late Fall 2019 revision |
| 5 | 2/24/2020 | Winter 2020 revision |
| 6 | 3/12/2020 | Late Winter 2020 revision |
|  |  |  |

Table. 1 Revisions

Colin Lundquist Date

Program Director: Kevin Pintong Date

## Abstract

trailTracer is an automatic video nature camera which can track animal targets. The system can detect movement nearby, recording and following animal movement with the camera's pan and tilt axes. This is called *Tracing* in the context of this project. The solution competes with others such as trailcams by offering high quality tracing video, indefinite runtime, and wireless image preview and settings adjustment through a web app.

The project was inspired by my hobby for nature. Observing the wildlife of Klamath Falls is exciting, but often times too time consuming. Adding to the difficulty, every animal in the area has adapted to the open terrain, making them very elusive. trailTracer will be a platform for studying the behaviors and interactions of wildlife in Klamath Falls. The goal is to compile an assortment of data, and learn something new about the past, present, and future of local animal species.

The project is designed for hobbyists, but it could be used in formal science as well. The system will consist of a Single Board Computer, with an HD camera and servo motors, plus any necessary sensors. The system is implemented with a 'hat' PCB which sits atop the Raspberry Pi. The cost of materials is estimated to be $350.

# Table of Contents

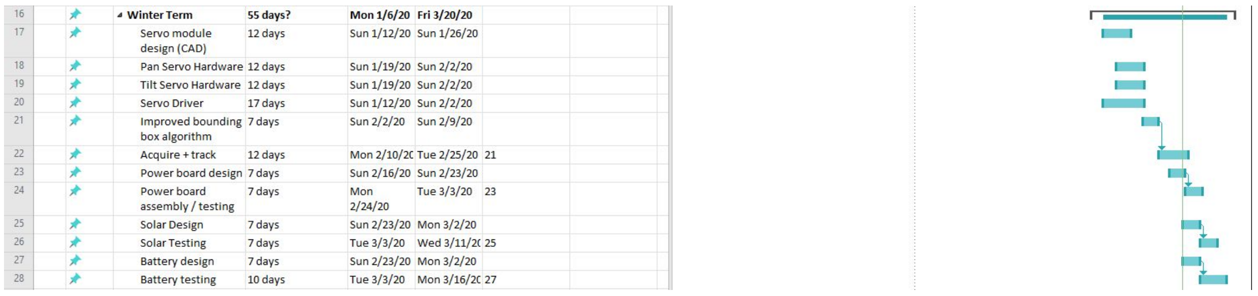
## Table of Contents

Table. 2 Contents

# Project Management

## Schedule Overview

Figure 1 (below) shows the winter term schedule. The project management was implemented in Microsoft Project.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | | Winter Term | 55 days? | Mon 1/6/20 | Fri 3/20/20 | | |
| 17 | | Servo module design (CAD) | 12 days | Sun 1/12/20 | Sun 1/26/20 | | |
| 18 | | Pan Servo Hardware | 12 days | Sun 1/19/20 | Sun 2/2/20 | | |
| 19 | | Tilt Servo Hardware | 12 days | Sun 1/19/20 | Sun 2/2/20 | | |
| 20 | | Servo Driver | 17 days | Sun 1/12/20 | Sun 2/2/20 | | |
| 21 | | Improved bounding box algorithm | 7 days | Sun 2/2/20 | Sun 2/9/20 | | |
| 22 | | Acquire + track | 12 days | Mon 2/10/2( | Tue 2/25/20 | 21 | |
| 23 | | Power board design | 7 days | Sun 2/16/20 | Sun 2/23/20 | | |
| 24 | | Power board assembly / testing | 7 days | Mon 2/24/20 | Tue 3/3/20 | 23 | |
| 25 | | Solar Design | 7 days | Sun 2/23/20 | Mon 3/2/20 | | |
| 26 | | Solar Testing | 7 days | Tue 3/3/20 | Wed 3/11/2( | 25 | |
| 27 | | Battery design | 7 days | Sun 2/23/20 | Mon 3/2/20 | | |
| 28 | | Battery testing | 10 days | Tue 3/3/20 | Mon 3/16/2( | 27 | |

Fig. 1 Winter Schedule

The following items were completed on time:

- Servo module design
- Pan servo hardware
- Tilt servo hardware
- Servo driver
- Acquire + Trace
- Battery design
- Battery testing
- Solar design
- Solar testing

The following items are in progress:

- Power board design
- Improved bounding box algorithm

Some Items will be pushed to Spring term:

- Power board assembly / testing

# Conceptual Overview

## Introduction

The trailTracer project developed out of an interest for wildlife in the Klamath Falls area. The open terrain lends itself to excellent animal-watching, and the wild seasonal changes make it more interesting to observe. I knew I would need to spend a lot of time working on a project, so I took the opportunity to combine the hobby with school.

Last year, I started noticing Yellow-Bellied Marmots crossing the road on my way home from school. I took immediate interest in the creatures, and followed them to a pile of rocks. As I researched the marmots more, I learned that the male claims a rock-palace and attracts multiple females throughout the summer mating season. The marmot family takes turns looking out for predators, and warning the others with their signature scream.

I challenged myself to approach the marmot family near my house without the marmots noticing me. Bringing my binoculars, I moved from bush to bush in an attempt to outsmart the animals. In the end, my eyesight could not compete with the marmots, even with binoculars, and prior knowledge of their location. This showed the incredible power of evolution to generate life seemingly perfectly adapted to its environment. After this, I was inspired to document the survival mechanisms of other animals, which led me to this engineering / scientific project.

## Problem Statement

When viewing many animals, standard equipment such as binoculars and camera is appropriate. Animals like deer are large enough to be spotted from a distance, and birds are often at close range. Some animals, though, are small, evasive, quick, or even dangerous. It is more difficult to observe bunnies, rabbits, cougars, lynx, marmots, and coyotes. Getting up-close images of predators is particularly difficult and risky.

## Intended Audience

The product is intended for use by nature hobbyists, as well as people studying animals. The intended price target makes trailTracer much more expensive than trailcams, and therefore targets hobbyists and scientists. The product could be adapted for uses such as security, but this is not the focus of the project. Some stretch goals may also change the target audience. For example, the aircraft tracer stretch goal would appeal to aviation hobbyists as well.

## Proposed Project

The project is an automated nature camera which can detect and follow movement at close range. The system consists of a camera unit and a web app. The camera is mounted on two servos, which allow it to follow moving objects around it. The camera unit is capable of operating for long periods due to its solar panel, and motion sensor that allows it to conserve power until activated by movement.

## State of the Art

In order to photograph wildlife in the proposed manner, a human would be required to manually follow the animal target. This method produces the highest quality footage, but also requires immense patience and/or money. Another solution may be to use a trailcam. Such a device remains idle until a close-range sensor activates the camera, which  may capture images or video. This solution does not offer a tracing feature, meaning it is inadequate for capturing moving video, especially of small animals.

| Method | Cost | Benefits | Drawbacks |
|---|---|---|---|
| Human Operator | $50,000 / year | Skill, Animal tracing | Size, Cost |
| TrailCam | $100+ | Inexpensive | No tracing |
| trailTracer | ~$300 | Tracing | |

Table. 3 Alternatives

## Deliverables

The system will be completed and tested outdoors until a sample of image and video data has been collected. Select images and video from the tests will be presented along with the system hardware.

## Stretch Goals

Some stretch goals have also been considered, but it is unlikely these will be developed.

| | |
|---|---|
| LTE Connectivity | Web app is available from anywhere in the world through a cellular connection. |
| AI Acceleration | The system incorporates the Jetson Nano AI dev kit for learning / tracking application. |
| Sensor Field | Inexpensive sensors can be scattered throughout the terrain, and each one can trigger a notification on the app. |

Table. 4 Stretch Goals

# Requirements

1. The system shall include a single unit with a camera
    a. The resolution shall be at least 640x480 pixels
    b. The video frame rate shall be at least 30 FPS
2. The system shall weigh less than 10 pounds
3. The camera unit shall not be larger than 21 inches in any dimension
4. The system shall be capable of tilting and panning the camera
    a. The panning range shall be at least 180 degrees
    b. The tilting range shall be at least 60 degrees
    c. The servos shall be capable of rotating the camera at least 15 degrees per second in both axes

5. The system shall have a user interface accessible wirelessly from a smart device via WiFi
    a. The interface will be a website
    b. A camera preview shall be visible from the interface
        i. The preview shall update at least once every 3 seconds
    c. The interface shall allow the user to adjust system settings, including:
        i. Camera tilt and pan angle
        ii. System shutdown
        iii. Capture modes
        iv. View captured images and videos
        v. Download / Delete images and videos
6. The system shall be capable of choosing one of the following conditions to become ***active*** upon:
    a. Battery percentage reaches a predetermined amount
    b. an object is detected within 10 feet of the sensor
    c. after a predetermined length of time
7. The system shall be capable of choosing the following conditions to become ***inactive*** upon:
    a. Battery percentage reaches a predetermined percentage
    b. After a predetermined length of time
    c. System reaches a predetermined value
8. The system shall capture video and images at least 640x480 pixel in resolution while ***active***

9. The system shall remain operational after 24 hours of use outdoors in Klamath Falls.
10. The system shall be capable of withstanding rain. (the device need not withstand being submerged)
11. The system shall be capable of storing at least 4 hours of video at 640x480 resolution @ 30 fps.
12. The system shall feature a solar panel
    a. The solar panel shall be capable of supplying enough power for the system to be ***active*** 10 percent of the time
13. The system shall source power from a battery pack
    a. The standby time shall be at least 24 hours
    b. The system must be capable of being ***active*** for 4 hours
    c. The battery shall charge from an AC adaptor with a DC plug
    d. The battery shall charge from solar
14. The system shall feature adequate cooling so that all components remain within operating temperature
15. The system shall consume less power while not ***active.***

Definition of ***active***: The system has turned on and begins to record the scene. The power consumption is lower when the device is not active.

# System Description

## General Description

The trailTracer is intended to be a long-term nature imager and video recorder. In order to accommodate this, a large battery and a solar panel are included in the camera unit. These power a single board computer, which interfaces with the camera, sensor, and servo modules. This allows the camera unit to move the camera based on video tracking information, and sleep until activated by close-range motion or other conditions.

trailTracer depends on its power system to stay alive. Because there are no power outlets in nature, the system must gather a large amount of energy from the sun, and also be capable of storing a large amount in its batteries. The battery is constructed of high-energy lithium cells, providing hours of continuous runtime. The 12 volt solar panel(s) will output enough power to keep the system on for around 50 percent of the time. In order to make use of these sources, the power system must also be smart. A small, low- power microcontroller manages the power system. The main high-power computer can request to sleep for a duration of time, saving power until time's up, when the computer is powered up and begins waiting for an animal to cross it's path.  During the inactive time, the system sleeps and charges its batteries. Other conditions can trigger the device to become active or inactive, such as temperature or battery level readings. This means the system can intelligently manage how it spends its energy. An operating scheme might involve long periods of charging, followed by an entire day of recording, or a more balanced schedule.

The camera module is heavily interfaced with the servo module. Of course, in order to trace an animal's movement, the servos must move the camera in the pan and tilt directions. The camera is attached to the tilt servo, which is in turn attached to the pan servo. The video data is interpreted using a tracking algorithm, and the servo module uses an advanced control scheme to smoothly follow the movement detected. With each module working as a system, trailTracer will be able to perform long recording missions completely autonomously, and without interrupting it's animal targets.

## System Block Diagram

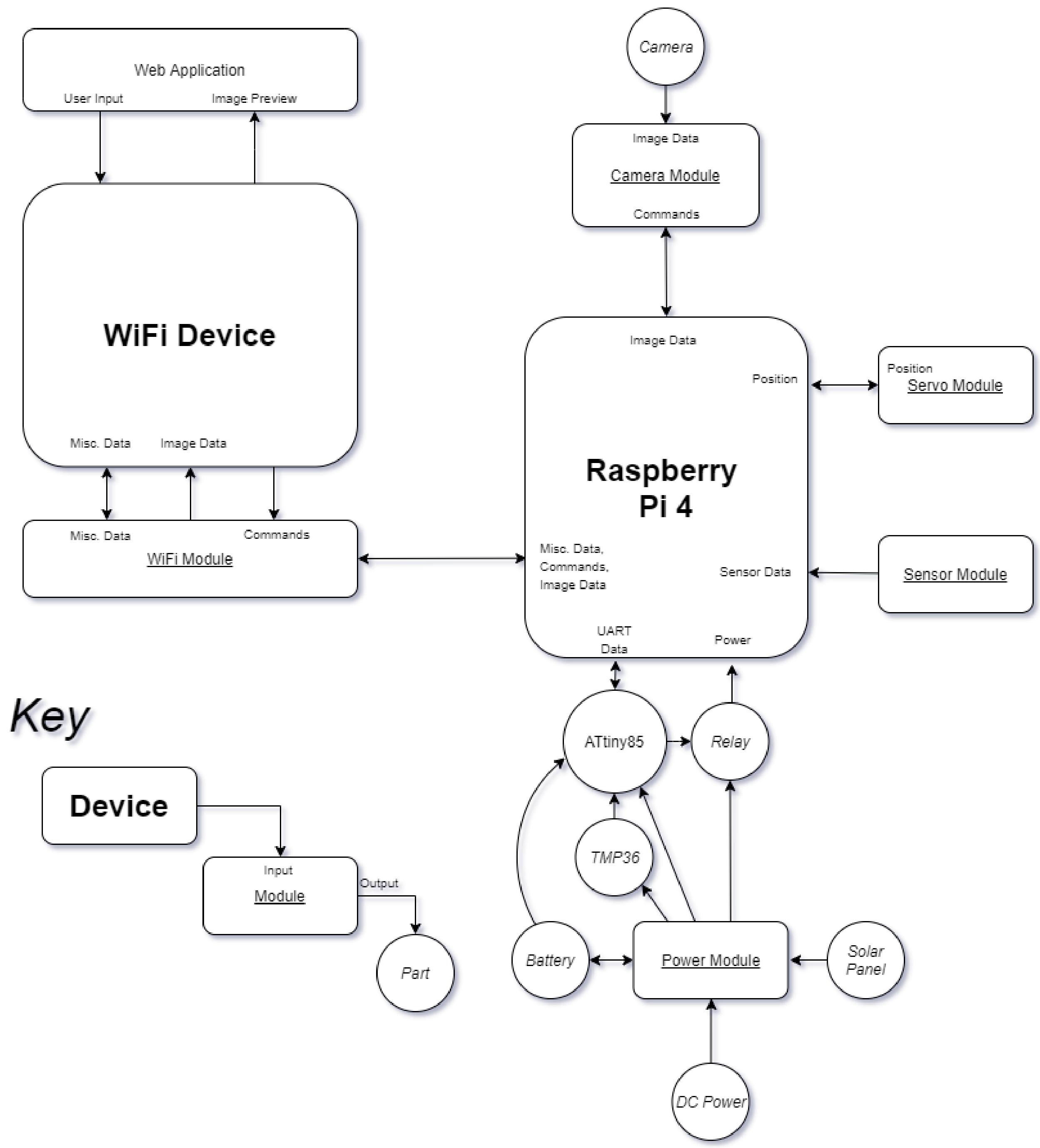


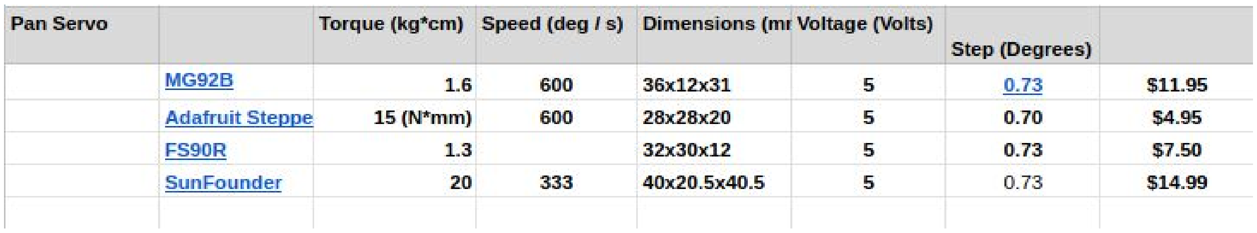
Fig. 2 Block Diagram

Since revision 4, the ATTiny85 module has been added. The microcontroller reads data from the TMP36 temperature sensor, and the voltage level directly from the battery. The ATTiny is responsible for powering down, and powering up the Raspberry Pi.

## Major Subsystems

The *Camera Module* is the first subsystem. It consists of a camera with the capability to capture images and video, and transfer raw data to the camera unit. The PiCamera V2 is a bulky unit, so an extender cable is used to separate the headboard from the sensor itself. This reduces the load on the tilt servo.

The *Servo Module* controls the pan and tilt angle of the camera. The subsystem must track or monitor current angular position, and move the camera to a position requested by the system.



| Pan Servo | | Torque (kg*cm) | Speed (deg / s) | Dimensions (mr | Voltage (Volts) | Step (Degrees) | |
|---|---|---|---|---|---|---|---|
| | MG92B | 1.6 | 600 | 36x12x31 | 5 | 0.73 | $11.95 |
| | Adafruit Steppe | 15 (N*mm) | 600 | 28x28x20 | 5 | 0.70 | $4.95 |
| | FS90R | 1.3 | | 32x30x12 | 5 | 0.73 | $7.50 |
| | SunFounder | 20 | 333 | 40x20.5x40.5 | 5 | 0.73 | $14.99 |

Fig. 3 Pan Servo Decision

Of the original servo motors compared, they all have similar torque outputs. The main deciding factor is the need for an encoder. The MG92B micro servo does not need an encoder, since it's position is controlled by PWM, and is self-correcting. The limiting factor for this servo is the 180 degree rotation range. If the servo doesn't have enough torque to directly pan the mechanism, then this servo won't work. In this case, a gearing mechanism will be used to increase the effective torque of a 360 degree servo. Servo step size is not advertised by manufacturers, so a generic measurement was used [1].

Currently, both pan and tilt motors are general purpose servos with self-correcting behavior. The SunFounder server was found to be the smoothest and most accurate servo, likely due do it's metal gears and high torque output.

The *Sensor Module* sends close-range movement data to the camera unit, in order to activate it. The sensor module may include one or both of the following: Passive Infrared, Radar.

| Radar Sensor | | Power | Range | Voltage (Volts) | FOV Vertical | FOV Horizontal | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | HB100 | 200 mW | 15 m | 5 | 36 deg. @ -3dB | 72 deg. @ -3dB | $7.99 | Analog Output, needs amplifier |
| | RCWL-0516 | ~10 mW | ~3 m | 4-28 | ? | ? | $8.99 | Digital Output, 5 pack |
| | HiLetGo PIR | 0.330 mW ? | 3- 5 m | 2.7 - 12 | 100 | 100 | $8.49 | |

Fig. 4 Sensor Decision

The choice of sensor was narrowed down to Radar and Passive Infrared technologies or 'PIR'. PIR sensors rely on a large difference in temperature between objects and the surrounding environment. It is possible that the difference in temperature is not great enough during certain times of the year, limiting the effectiveness of a PIR sensor. A radar sensor is not vulnerable to this weakness, but may cause false positives as it is susceptible to background movement. Originally, the sensor was intended to only trigger upon an animal entering the camera frame. In other words, the sensor's field of view would be restricted. No sensor with a narrow enough field could be found. One solution to this problem would be to relocate the sensor to somewhere within the frame instead of being mounted on the system itself. An engineering change notice was given during winter term which allowed the use of an additional sensor unit instead of integrating the sensor into the camera unit. This unit may house multiple PIR sensors and a single radar sensor. If this unit is not implemented, however, only a single PIR sensor will likely be used in the camera unit.

The *Power Module* charges the battery from the DC power port, and the solar panel. The battery is a bank of 18650 Lithium-Ion cells. The battery capacity must be large in order to meet the continuous capture requirement of 4 hours. In testing, the system would draw a maximum of 7 Watts. This means a battery around 28Wh would be sufficient to meet this requirement.

| Battery | Cell Capacity (mAh) | Number of Cells | Cell Voltage Max | Cell Voltage Min | Total Cell Capacity (Ah) | Battery Capacity (Wh) | |
|---|---|---|---|---|---|---|---|
| | 735 | 6 | 3.7 | 2.7 | 4.41 | 14.112 | |
| | | | | | | | |
| Battery Mgmt. | Battery output (Wh) | Efficiency (%) | BMS Output (Wh) | | | | |
| | 14.11 | 90.00 | 12.7008 | | | | |
| Buck Converter | BMS output (Wh) | Efficiency (%) | Buck Output (Wh) | | | | |
| | 12.7008 | 90.00 | 11.43072 | | | | |
| Solar Panel | Output (Watts) | Efficiency % | Actual Output (W) | | | | |
| | 5 | 70 | 3.5 | | | | |
| Power Draw (W) | Raspberry Pi | Camera | Sensor (meas.) | Tilt Servo | Pan Servo | Solar | Total |
| Active | 4 | 0.25 | 0.015 | 1 | 1 | -3.5 | 2.765 |
| Inactive | 0 | 0 | 0.015 | 0 | 0 | -3.5 | -3.485 |
| | | | | | | | |
| Operation Time (Hr:Min) | Always Inactive | 25% Duty Cycle | 50% Duty Cycle | 75% Duty Cycle | 90% Duty Cycle | Always Active | Break Even % |
| | 0:00 | 0:00 | 0:00 | 9:30 | 5:20 | 4:08 | 55.76% |

Fig. 5 Power Calculations

Figure 5 details the power flow through the system beginning with the total capacity of the battery. The battery management system (BMS) and buck converter are two factors of inefficiencies taken into account. Two power states are considered for an estimation of run-time capability: Active and Inactive. In the active state, the power draw is much higher, as all the subsystems are in use. Different duty cycles are considered. Each one represents the percentage of time the system will be active, and the result is the amount of time the system can stay active before running out of energy in the battery. It was found that the system could only be active for 4 hours using the battery system currently implemented. This is because each low quality cell can only store 735 mAh. The battery pack was intended only as a test, and a new one will be constructed using higher quality cells. The new cells will store between 2000 and 3000 mAh.
Added to the figure since revision 4, is the solar duty cycle calculation (Break Even %). This shows what percentage of time the system can be active, while 'breaking even' between power consumption and solar output. In other words, the system can run indefinitely at the given duty cycle. It was found that with a 5 Watt panel, operating at 70% efficiency, the system could stay active around 50% of the time. Unfortunately, the solar panel purchased only outputs around 1.4 watts, so two of these panels would be necessary to achieve such a duty cycle.

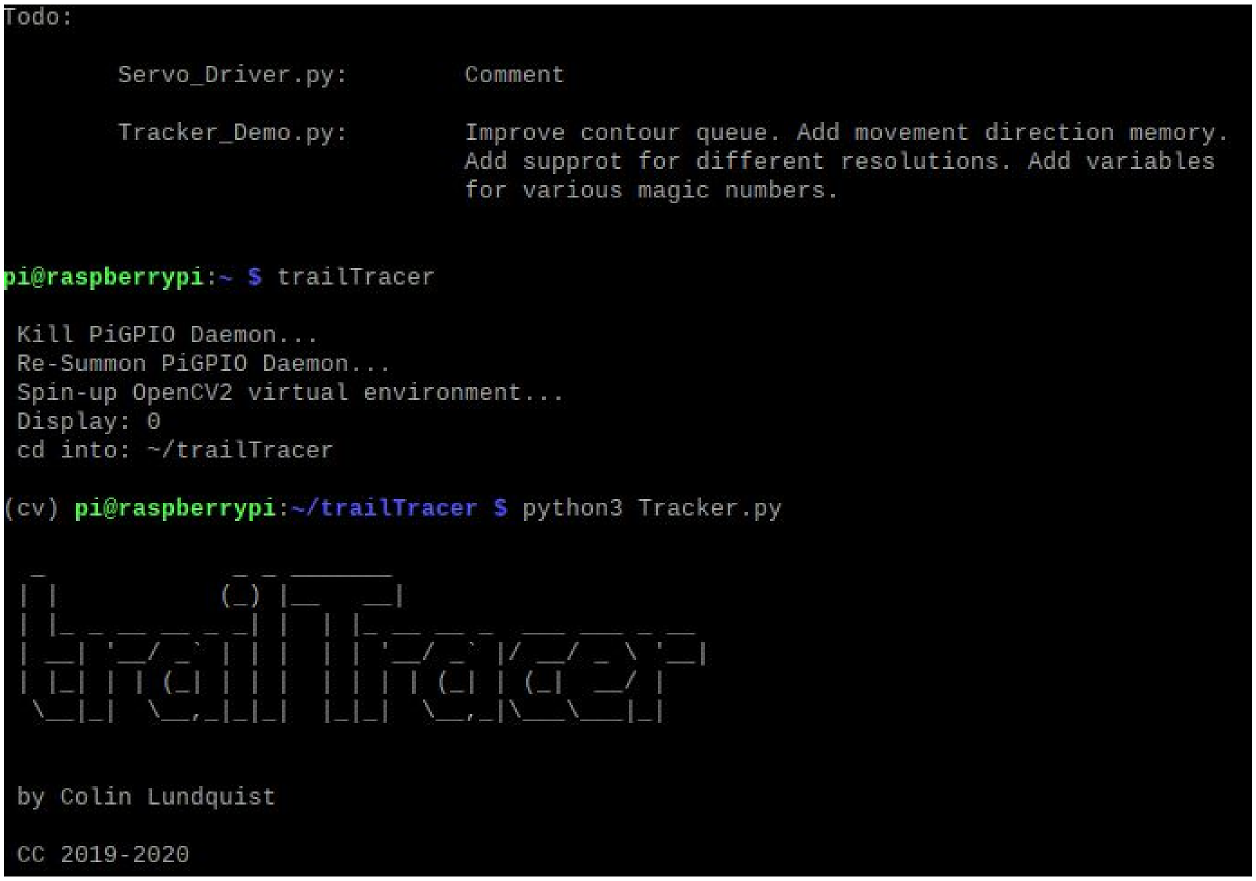
The *WiFi Module* sends and receives image data, miscellaneous data, and commands in order to implement the software requirements. The interface is accessed through a web application after joining the system's WiFi network. A WiFi access point will be established using a wireless adapter dongle, or the built-in WiFi radio on the Pi 4.

## Hardware Platform

The Raspberry Pi 4 model B is used as the main controller in the camera unit. The Pi handles object tracking, web app hosting, and file storage. An additional microcontroller, the ATtiny85, will be used to manage the Pi power connection. The Pi draws full power even when turned off; Because of this, a relay is needed to properly disconnect the Pi. The ATtiny85 will wait for a wake condition (timer, sensor, battery), then reconnect power to the Pi, which boots automatically.

## Software Platform

The Pi runs a Linux distribution in order to simplify a number of things. Bluetooth, File I/O, and especially Motion tracking is more achievable with open source software on this platform. OpenCV is the most popular and well documented software platform for object tracking. For these reasons, it is used for computer vision tasks.The project planning software used is Microsoft Project, because it is free from the CSET department, and is designed for my operating system.

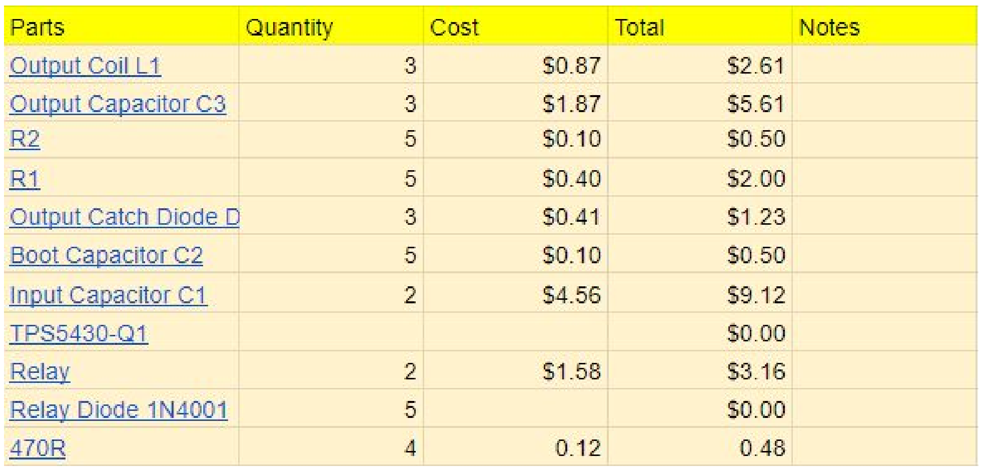


```
Todo:

        Servo_Driver.py:        Comment

        Tracker_Demo.py:        Improve contour queue. Add movement direction memory.
                                Add supprot for different resolutions. Add variables
                                for various magic numbers.


pi@raspberrypi:~ $ trailTracer

 Kill PiGPIO Daemon...
 Re-Summon PiGPIO Daemon...
 Spin-up OpenCV2 virtual environment...
 Display: 0
 cd into: ~/trailTracer

(cv) pi@raspberrypi:~/trailTracer $ python3 Tracker.py

  _             _ _ _______
 | |           (_) |__   __|
 | |_ _ __ __ _ _| |  | |_ __ __ _  ___ ___ _ __
 | __| '__/ _` | | |  | | '__/ _` |/ __/ _ \ '__|
 | |_| | | (_| | | |  | | | | (_| | (_|  __/ |
  \__|_|  \__,_|_|_|  |_|_|  \__,_|\___\___|_|


 by Colin Lundquist

 CC 2019-2020
```

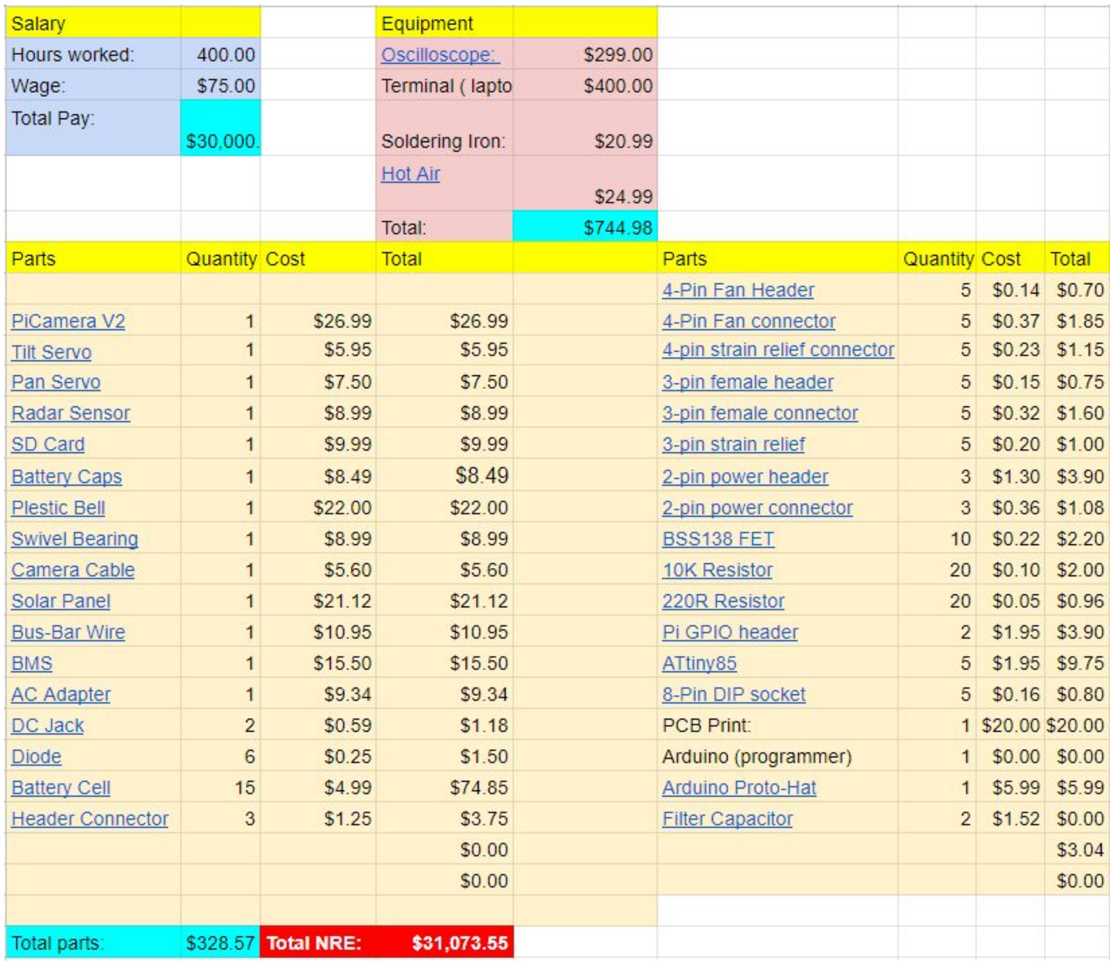
Fig. 6 Development Environment

Figure 6 shows the development environment used for trailTracer. The trailTracer alias runs a script which starts the necessary services for operation. The GPIO daemon, and computer vision virtual environment are started, and the proper monitor display is assigned to the Operating system. Also displayed is the trailTracer splash screen, and todo list which appears on login.

# Non-Recurring Engineering Costs



| Parts | Quantity | Cost | Total | Notes |
|---|---|---|---|---|
| Output Coil L1 | 3 | $0.87 | $2.61 | |
| Output Capacitor C3 | 3 | $1.87 | $5.61 | |
| R2 | 5 | $0.10 | $0.50 | |
| R1 | 5 | $0.40 | $2.00 | |
| Output Catch Diode D | 3 | $0.41 | $1.23 | |
| Boot Capacitor C2 | 5 | $0.10 | $0.50 | |
| Input Capacitor C1 | 2 | $4.56 | $9.12 | |
| TPS5430-Q1 | | | $0.00 | |
| Relay | 2 | $1.58 | $3.16 | |
| Relay Diode 1N4001 | 5 | | $0.00 | |
| 470R | 4 | 0.12 | 0.48 | |



| Salary | |
|---|---|
| Hours worked: | 400.00 |
| Wage: | $75.00 |
| Total Pay: | $30,000. |

| Equipment | |
|---|---|
| Oscilloscope: | $299.00 |
| Terminal ( lapto | $400.00 |
| Soldering Iron: | $20.99 |
| Hot Air | $24.99 |
| Total: | $744.98 |

| Parts | Quantity | Cost | Total |
|---|---|---|---|
| | | | |
| PiCamera V2 | 1 | $26.99 | $26.99 |
| Tilt Servo | 1 | $5.95 | $5.95 |
| Pan Servo | 1 | $7.50 | $7.50 |
| Radar Sensor | 1 | $8.99 | $8.99 |
| SD Card | 1 | $9.99 | $9.99 |
| Battery Caps | 1 | $8.49 | $8.49 |
| Plestic Bell | 1 | $22.00 | $22.00 |
| Swivel Bearing | 1 | $8.99 | $8.99 |
| Camera Cable | 1 | $5.60 | $5.60 |
| Solar Panel | 1 | $21.12 | $21.12 |
| Bus-Bar Wire | 1 | $10.95 | $10.95 |
| BMS | 1 | $15.50 | $15.50 |
| AC Adapter | 1 | $9.34 | $9.34 |
| DC Jack | 2 | $0.59 | $1.18 |
| Diode | 6 | $0.25 | $1.50 |
| Battery Cell | 15 | $4.99 | $74.85 |
| Header Connector | 3 | $1.25 | $3.75 |
| | | | $0.00 |
| | | | $0.00 |
| | | | |
| Total parts: | $328.57 | Total NRE: | $31,073.55 |

| Parts | Quantity | Cost | Total |
|---|---|---|---|
| 4-Pin Fan Header | 5 | $0.14 | $0.70 |
| 4-Pin Fan connector | 5 | $0.37 | $1.85 |
| 4-pin strain relief connector | 5 | $0.23 | $1.15 |
| 3-pin female header | 5 | $0.15 | $0.75 |
| 3-pin female connector | 5 | $0.32 | $1.60 |
| 3-pin strain relief | 5 | $0.20 | $1.00 |
| 2-pin power header | 3 | $1.30 | $3.90 |
| 2-pin power connector | 3 | $0.36 | $1.08 |
| BSS138 FET | 10 | $0.22 | $2.20 |
| 10K Resistor | 20 | $0.10 | $2.00 |
| 220R Resistor | 20 | $0.05 | $0.96 |
| Pi GPIO header | 2 | $1.95 | $3.90 |
| ATtiny85 | 5 | $1.95 | $9.75 |
| 8-Pin DIP socket | 5 | $0.16 | $0.80 |
| PCB Print: | 1 | $20.00 | $20.00 |
| Arduino (programmer) | 1 | $0.00 | $0.00 |
| Arduino Proto-Hat | 1 | $5.99 | $5.99 |
| Filter Capacitor | 2 | $1.52 | $0.00 |
| | | | $3.04 |
| | | | $0.00 |

Fig. 7 Non-Recurring Engineering Costs

## Design

*Software Design*

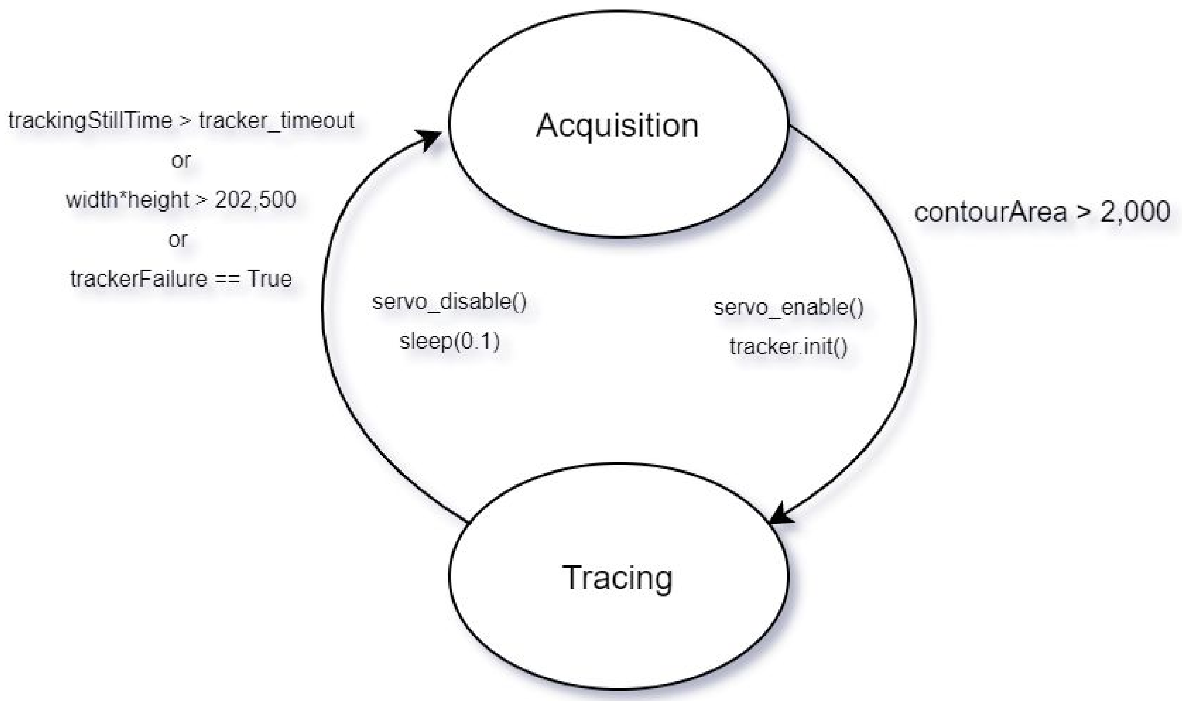
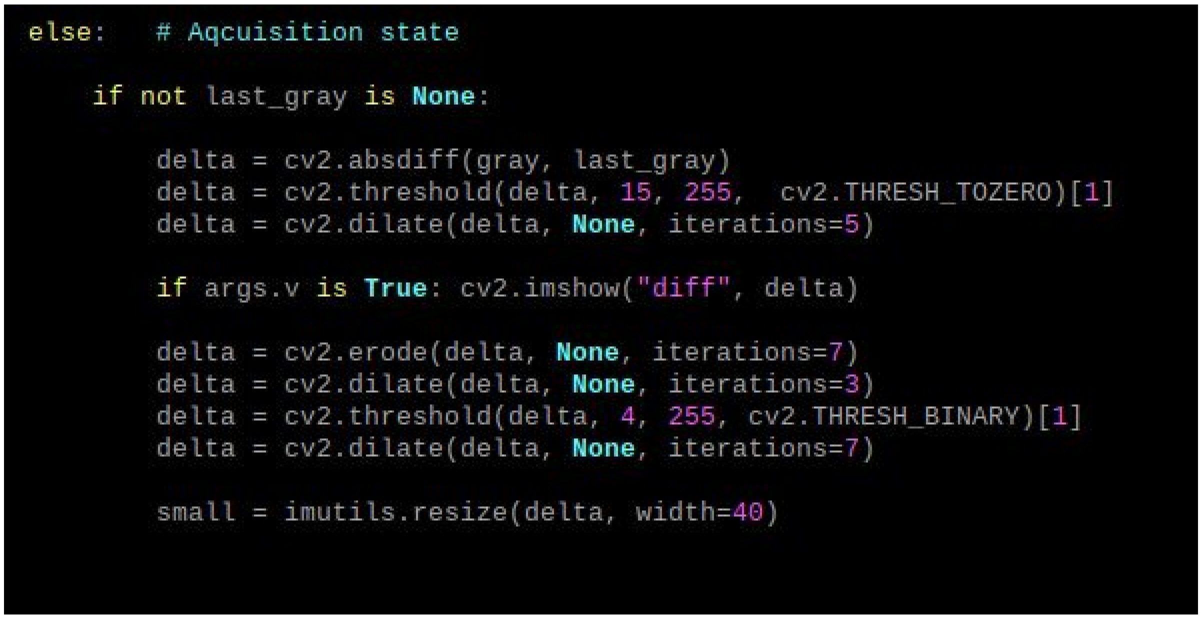


Fig. 8 State Machine

Research into tracking algorithms using OpenCV yielded a few tutorials into the subject. The code generated demonstrated robust object tracking given a user defined bounding box. This showed that even as the camera rotates, and the background moves, an object can be tracked. Initial testing software was based on the assumption that the camera will be stationary when the system is first activated. With this assumption in mind, basic (and low power) algorithms for detecting motion can define the bounding boxes of moving objects. Once this is done, the bounding box information can be passed to one of the tracking algorithms included in OpenCV. Finally, the bounding box location returned by this algorithm guides the motion of the camera.

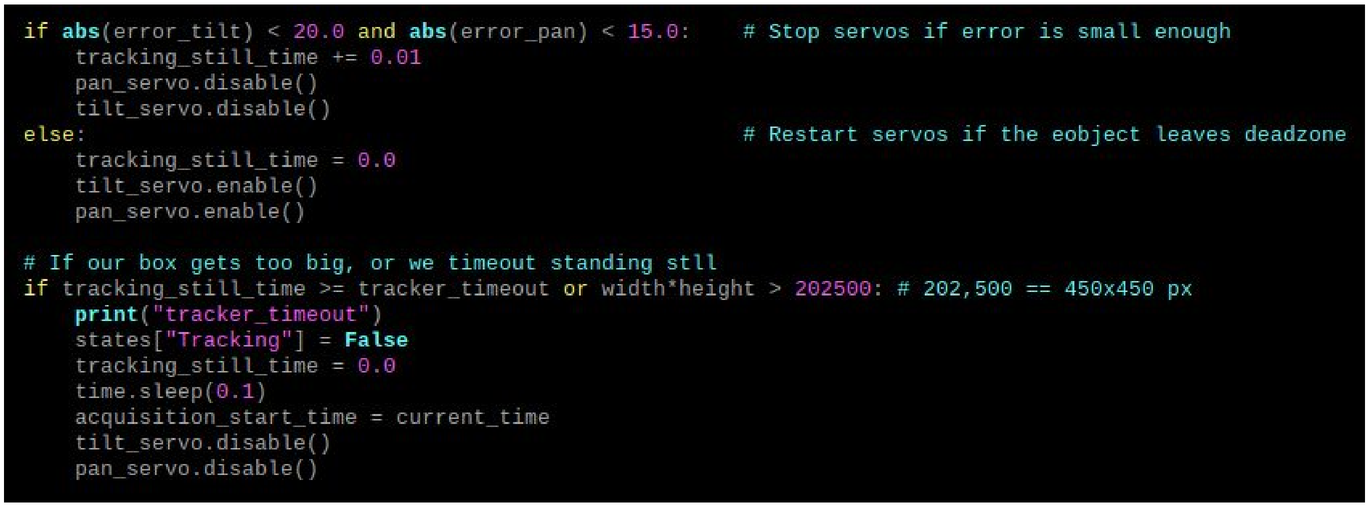
Only two states are currently implemented. In the acquisition state, a bounding box is produced from the difference between consecutive frames. Once a contour of great enough area is detected, the system enables the servos and initiates the tracking algorithm using the bounding box obtained. The system stays in the tracing state until one of the following conditions: The object is stationary for too long, the bounding box area is too large, or the tracker returns a failure.



```python
else:   # Aqcuisition state

    if not last_gray is None:

        delta = cv2.absdiff(gray, last_gray)
        delta = cv2.threshold(delta, 15, 255,  cv2.THRESH_TOZERO)[1]
        delta = cv2.dilate(delta, None, iterations=5)

        if args.v is True: cv2.imshow("diff", delta)

        delta = cv2.erode(delta, None, iterations=7)
        delta = cv2.dilate(delta, None, iterations=3)
        delta = cv2.threshold(delta, 4, 255, cv2.THRESH_BINARY)[1]
        delta = cv2.dilate(delta, None, iterations=7)

        small = imutils.resize(delta, width=40)
```

Fig. 9 Acquisition Code

The code in Figure 9 finds the difference between frames, then performs some processing to make the image more friendly for motion detection. The difference frame is first dilated, then eroded to smooth out the edges of shapes. OpenCV provides the ‘findContours()’ function for finding the shapes within the frame and creates contour objects. The largest contour is presumed to be the subject of the frame, and the extreme points are extracted from this. These extreme points form the bounding box used by the tracking algorithm.



```
if abs(error_tilt) < 20.0 and abs(error_pan) < 15.0:    # Stop servos if error is small enough
    tracking_still_time += 0.01
    pan_servo.disable()
    tilt_servo.disable()
else:                                                   # Restart servos if the eobject leaves deadzone
    tracking_still_time = 0.0
    tilt_servo.enable()
    pan_servo.enable()

# If our box gets too big, or we timeout standing stll
if tracking_still_time >= tracker_timeout or width*height > 202500: # 202,500 == 450x450 px
    print("tracker_timeout")
    states["Tracking"] = False
    tracking_still_time = 0.0
    time.sleep(0.1)
    acquisition_start_time = current_time
    tilt_servo.disable()
    pan_servo.disable()
```

Fig. 10 Servo Pause

Figure 10 shows code for another new stability feature: ‘Servo Pause State’. A ‘Deadzone’ is defined in both axes, which represents a square in which the camera should not move to center the object. In other words, the system waits until the object moves a certain distance away from the center before beginning tracing movement. When the object is within the ‘deadzone’, the servos are both disabled until it leaves the deadzone. The tracker continues to run in this state, returning updated bounding box information. While this algorithm is effective, it can be improved. Since the camera is stationary during this state, both the formal tracker and the acquisition algorithm may run simultaneously. This would improve reliability by comparing the bounding box results and detecting errors in the tracker. This special state will be implemented in the future as the program migrates away from a demonstration script to a formal state machine. The bounding box can be erratic, and is not accurate most of the time. In order to gain an accurate box, some anomalies have to be considered. The code was modified to reject bounding boxes which are too large. Such a box appears when the object gets too close to the camera. This condition would make the tracing unstable, so the system waits until the box appears smaller before beginning the tracing state. This is also shown in figure 10; The condition appears in the code as “width*height > 202500”.

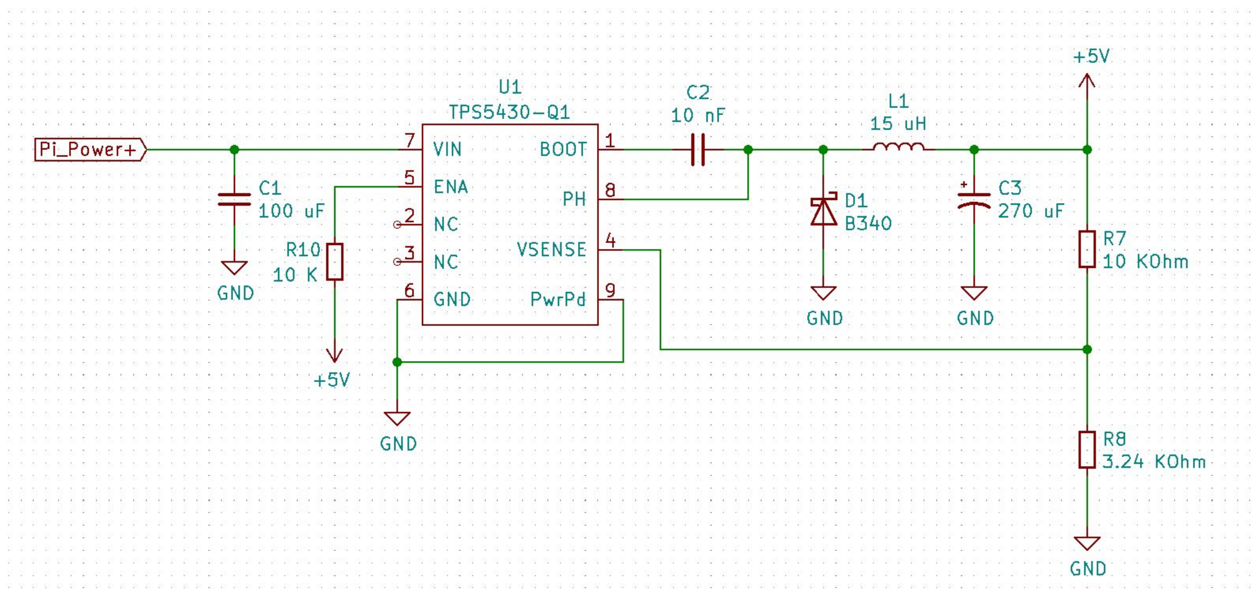


Fig. 11 Buck Converter Schematic

The buck converter accepts the 12 Volt power from the battery management system, and lowers the voltage to 5 Volts in order to power the Pi, ATtiny, servos, and sensors. The Texas Instruments TPS5430-Q1 was chosen due to its wide input voltage range (5.5 - 36 Volts), excellent documentation, and small number of required external components (7). Additionally, the chip includes features such as undervoltage lockout and slow-start, which may not be needed, but could prevent stability problems.

Texas Instruments provides useful equations for choosing the proper external components based on the use-case. For trailTracer, the 3 Amp current target, which is massively greater than the expected draw (for future expansion), matched the example schematic. This made it easy to verify the calculated componentry by comparing against TI's example. The size of the input and output filtering capacitors are the only difference between TI's example, and the implementation for trailTracer. Despite very little noise present on the battery management system, the capacitors were selected for maximal noise reduction, just in case.

*See the testing section for more details on the system noise.*

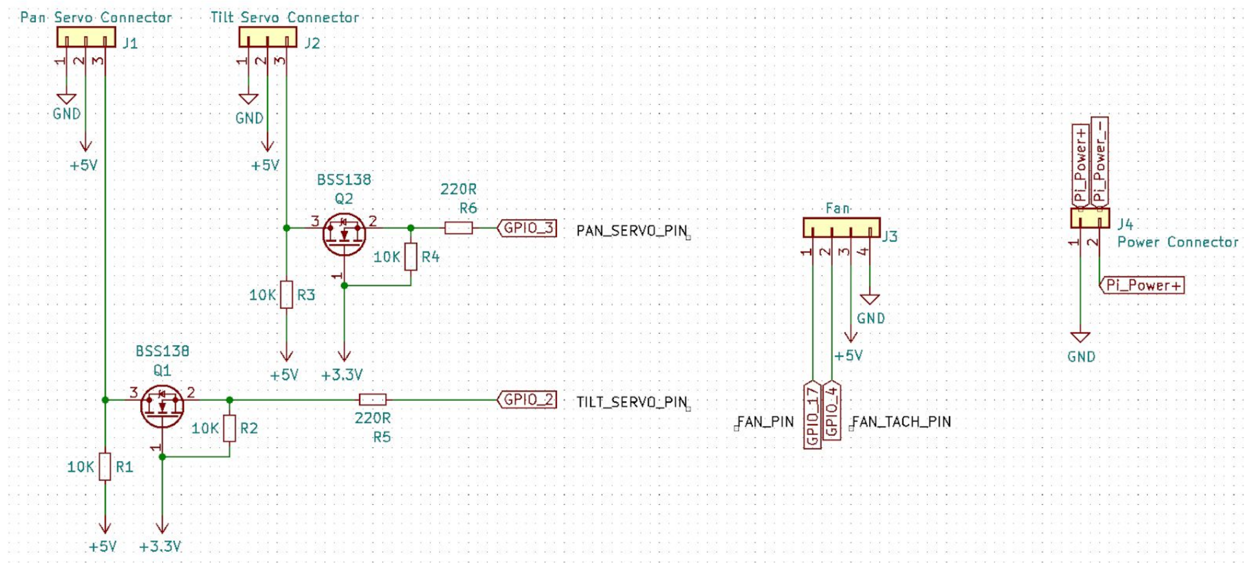


Fig. 12 Pin Header Schematic

Figure 12 shows the pin headers present on the board, as well as the servo module's logic level converter circuits. The logic level conversion is necessary because the Raspberry Pi 4 uses 3.3 volt logic on it's IO pins. The hardware performs the level conversion by pulling the output side up to 5 volts when the transistor is off (IO pin is high), and connecting directly to the IO when the transistor is on (IO pin is low). One circuit is needed for each servo, for two total. J4 is the 12 Volt power connector which connects the board to the battery management system, and J3 connects to the PWM controlled system fan.

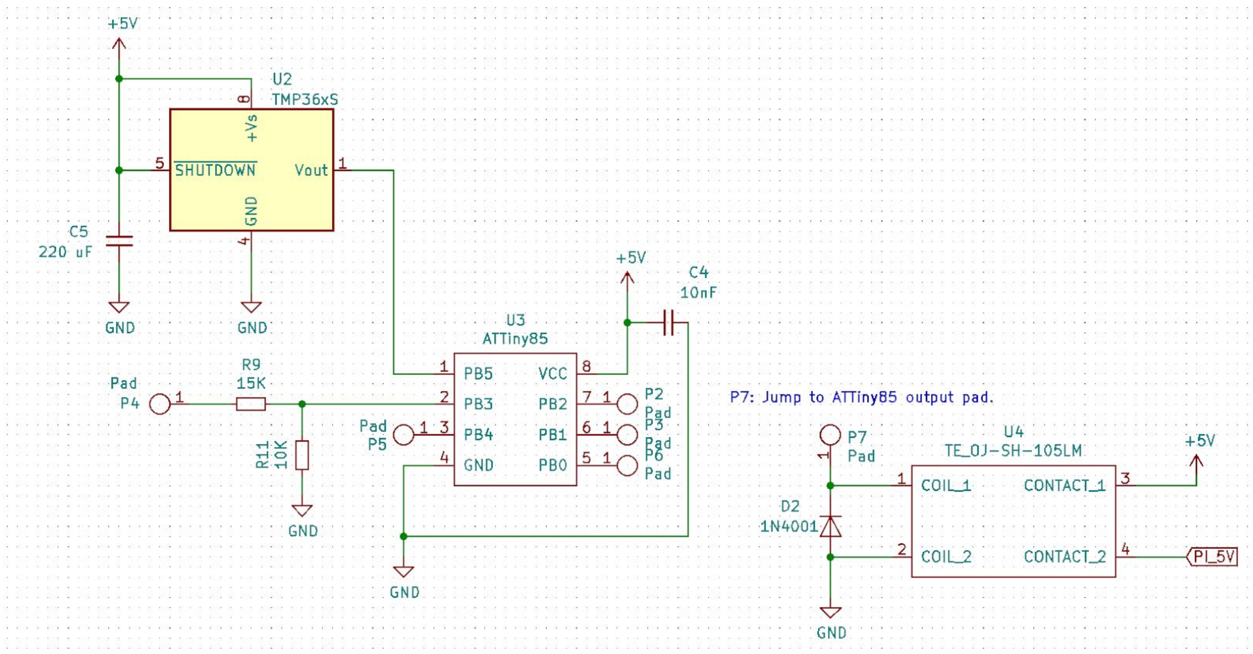


Fig. 13 ATTiny85 Schematic

The ATTiny is responsible for the power system, serial communication to the Pi, and sensor readings. The microcontroller reads data from the TMP36 temperature sensor, and will also receive data from the motion sensor. For the first revision of the PCB, some of the IO pins are left connected only to pads in order to make design work-arounds easier if they are necessary. If a shutdown or startup condition is detected, the ATTiny will operate the relay seen on the right of the figure.

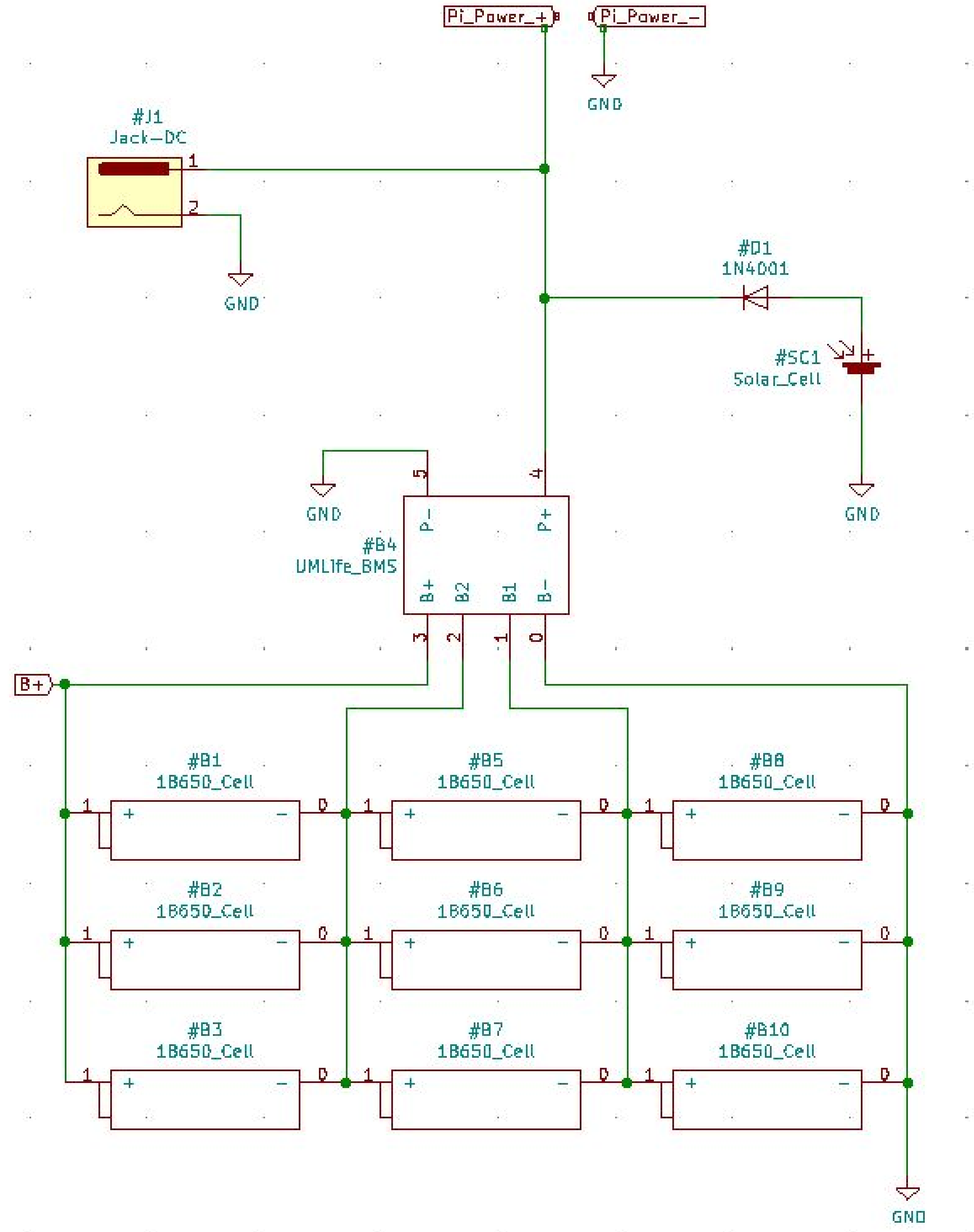
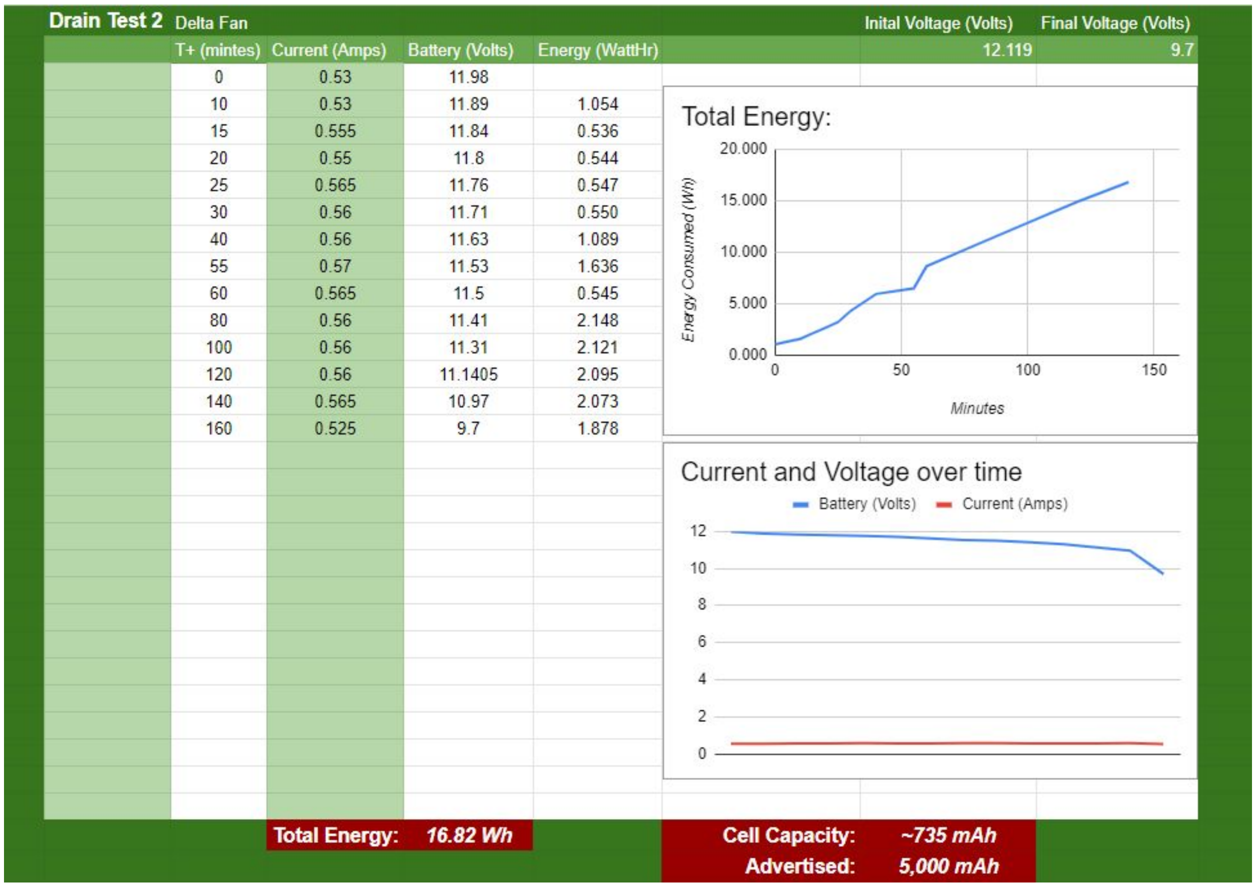


Fig. 14 Battery Schematic

The battery is assembled in the following way: three 18650 cells are arranged in parallel, and 3 sets of this arrangement are connected in series. This creates a 3S lithium battery pack with the capacity of 9 cells. The output voltage is up to 12.7 volts when fully charged. The battery management system is capable of discharging at a rate of 10 Amps.

## Testing

*Battery Test*



Drain Test 2 — Delta Fan

| T+ (mintes) | Current (Amps) | Battery (Volts) | Energy (WattHr) |
|---|---|---|---|
| 0 | 0.53 | 11.98 | |
| 10 | 0.53 | 11.89 | 1.054 |
| 15 | 0.555 | 11.84 | 0.536 |
| 20 | 0.55 | 11.8 | 0.544 |
| 25 | 0.565 | 11.76 | 0.547 |
| 30 | 0.56 | 11.71 | 0.550 |
| 40 | 0.56 | 11.63 | 1.089 |
| 55 | 0.57 | 11.53 | 1.636 |
| 60 | 0.565 | 11.5 | 0.545 |
| 80 | 0.56 | 11.41 | 2.148 |
| 100 | 0.56 | 11.31 | 2.121 |
| 120 | 0.56 | 11.1405 | 2.095 |
| 140 | 0.565 | 10.97 | 2.073 |
| 160 | 0.525 | 9.7 | 1.878 |

| Inital Voltage (Volts) | Final Voltage (Volts) |
|---|---|
| 12.119 | 9.7 |

Total Energy: 16.82 Wh

Cell Capacity: ~735 mAh
Advertised: 5,000 mAh

Fig. 15 Battery Testing

The battery was tested to determine whether it met the performance requirements. The battery was charged to 12.1 volts, then discharged with a 7 Watt fan, and test equipment monitoring the power output. One multimeter was connected in parallel in order to measure the voltage of the battery. Another one was connected in series to measure current. This gave a power reading which was recorded at different time intervals. Integrating the power data gives the total energy consumed by the test load. The battery capacity barely meets the requirements, allowing for 4 hours of continuous use. This is due to the inexpensive and low quality cells used. Because of this, a new battery will be designed which can meet requirements.

*Voltage Ripple / Noise Test*

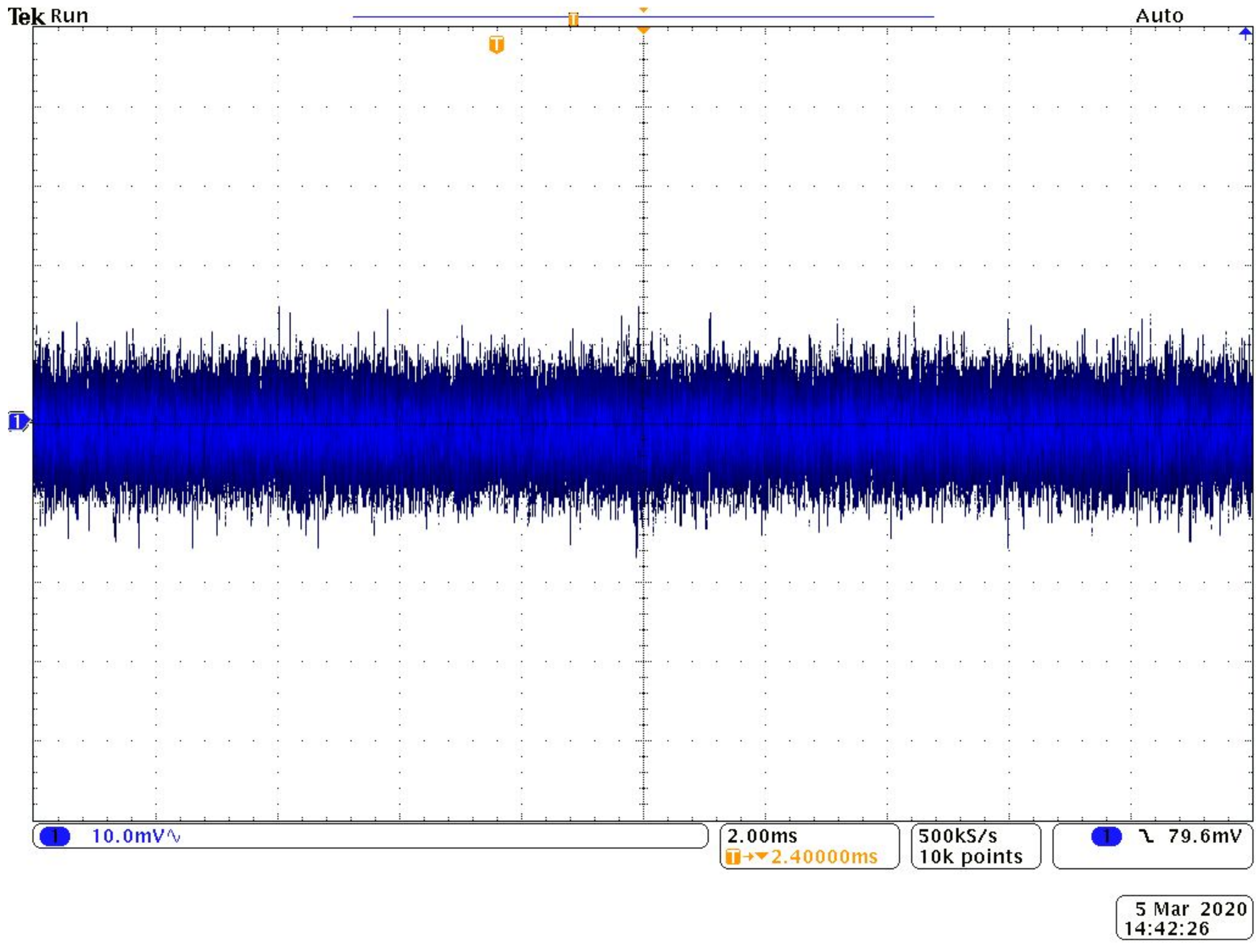


Fig. 16 Voltage Ripple / Noise

The voltage ripple and noise from the battery management system was measured. A resistive test load of 1 Watt was connected across the power terminals of the BMS, and the voltage across the load was measured by an oscilloscope. In future testing, a larger load of 7 Watts will be used to verify the results at loads more similar to the system load. While no measurable ripple existed, the noise amplitude was around 20 mV. In order to deal with this noise, a large 100 uF capacitor was chosen to decouple the input power from the buck converter. As further precaution, a 270 uF capacitor is used to smooth any additional noise generated by the switch-mode buck converter. These capacitors are oversized for the expected noise level, but for the first PCB revision, these were chosen as a precaution.

## Engineering Changes

*Change 1*

**Before:**

1. The system shall be a single unit with a camera

**After:**

1. The system shall include a unit with a camera

**Notes:** I might move the sensor (PIR, radar) to an additional unit.

*Change 2*

**Before:**

3. The system shall not be larger than 21 inches in any dimension

**After:**

3. The camera unit shall not be larger than 21 inches in any dimension

*Change 3*

**Before:**

12.

c. The battery shall charge from a USB port

**After:**

12.

c. The battery shall charge from an AC adapter with a DC plug

*Change 4*

**Before:**

13. The system shall feature ventilation so that all components remain within operating temperature

**After:**

13. The system shall feature adequate cooling so that all components remain within operating temperature

**Notes:**

I think this can be done without ventilation, but if not, ventilation is still an option.

*Change 5*

**Before:**

6. The system shall become ***active*** when a moving objects (sic) are detected within 10 feet of the system, and within a 30 degree field of view.

**After:**

6. The system shall be capable of choosing one of the following conditions to become ***active*** upon:

    a. Battery percentage reaches determined amount
    b. an object is detected within 10 feet of the sensor
    c. after a predetermined length of time

**Notes:**

The system will take far too long to boot to simply wake upon object detection in most cases. I didn’t want to sacrifice operational longevity, though, so these new wake conditions are a good compromise.

## Glossary

| | |
|---|---|
| Tracker | The formal tracking algorithm which continuously returns updated bounding box information on the animal movement. |
| Tracing | The pan and tilt servo motors are used to continuously keep the camera centered on the animal. |
| Acquisition State | The camera is stationary, and a difference between frames is used to acquire a bounding box of a moving animal. |
| Active | A system state in which the Pi is powered, and attempting to acquire motion. |
| Inactive | A system state in which the Pi is not powered, in order to save energy. |
| Pause State | The system pauses tracing, while continuing to update the formal tracker, but also acquiring new bounding boxes. |
| Camera Unit | The physical device and housing, which contains the camera and electronics. |
| Lithium Battery | Battery technology based on Lithium metal. Carries risk of fire. |
| SD Card | Flash storage card which may store the image data |
| CPU | Central Processing Unit. Controls the Linux Operating System. |
| Linux OS | Operating system which is used on the Pi. |
| 18650 Battery | Cylindrical Lithium-Ion battery common in electronics for it’s high capacity and compact size. |
| BMS | ‘Battery Management System’; Controls the charging and discharging of the batteries, and protects from hazardous states. |

## References

[1] T. Hoang, “Minimum Step on Servo.”