Spring 2018

Autonomous UAV Battery Swapping

Reed Jacobsen
raj37@zips.uakron.edu

Nikolai Ruhe
nmr33@zips.uakron.edu

Nathan Dornback
nad46@zips.uakron.edu

THE UNIVERSITY OF AKRON

Autonomous UAV Battery Swapping

Final Report

DT #07
12/1/2017

Creighton Cloud

Nathan Dornback

Walker Grossman

Reed Jacobsen

Nikolai Ruhe

Faculty Advisor: Dr. Choi

Senior Design Coordinator: Gregory A. Lewis

Nathan Dornback

As the software lead on this project, I was responsible for the integration of all software into a cohesive working system. Specifically, I developed all of the flight control software and the state machine that performs the battery swapping. All software written by other team members needed to go through me to that it could be integrated into the system. Lastly I managed the repository where all code was kept.

Reed Jacobsen

As the robotics integration lead for this project, I designed and manufactured both quadcopter frames, additionally I designed the driver circuitry for both electromagnet swapping mechanisms. Finally I contributed to software by integrating low level code for the infrared camera computer vision system

Nikolai Ruhe

As the team leader and test engineer, my responsibilities included developing network communications software for the companion computers associated with each drone. I was also responsible for flight testing as the main pilot for both drones. Lastly I created test code for the electromagnet swapping mechanisms.

1.0 Table of Contents

1.1 List of Figures

1.2 List of Tables

2.0 Abstract

One of the main hindrances of unmanned aerial vehicle (UAV) technology are power constraints. One way to alleviate some power constraints would be for two UAVs to exchange batteries while both are in flight. Autonomous mid-air battery swapping will expand the scope of UAV technology by allowing for indefinite flight times and longer missions. A single board computer will control each UAV's flight software to respond to inputs to align with each other mid-flight. When the two UAVs have joined, mechanical components will exchange a depleted battery on the worker UAV for a freshly charged battery that belongs to the battery supply UAV. After the exchange, the drones will then detach themselves from each other, and the worker UAV will resume its mission while the battery supply UAV returns back to the ground control station.

3.0 Problem Statement

3.1 Need

As Unmanned Aerial Vehicles (UAVs) become more ubiquitous in both private and commercial use, effective power management will become paramount to the growth of the UAV industry. Many UAVs are being designed to deliver packages, perform surveillance, and even create new types of art. Because of the expanding mission scopes of UAVs, either larger power supplies or more frequent battery replacements will be needed to service these complex tasks. Due to these power constraints, UAV operators face a bottleneck in the amount of clients they can serve as well as the types of jobs they can perform. Eliminating some of these power

constraints not only represents an increase in time efficiency, but it could also allow UAV companies to service more clients which poses a significant financial interest as well.

3.2 Objective

The mission of the project is to successfully develop and test a system that broadens the scope of UAV technology by utilizing robotic battery replacements to eliminate power constraints.

3.3 Background

During the design phase process, research was needed to better understand the technical domain encompassing the system. The subsequent documentation lists the wide array of articles and other scholarly research pertaining to this design project.

3.3.1 Patent Search

Because UAV technology has had many patents recently prosecuted, there are numerous pieces of intellectual property that have direct or auxiliary relevance to the design of this project.

Published on 08-20-2013, US patent number 8511606B1 assigned to Boeing Co established a method and device that created a system for UAVs to land, receive power, and transmit data. This grant describes a device that would allow a ground station to serve numerous UAVs through power replenishment, data transfer, and housing [4]. While vaguely written to encompass as many commercial applications as possible, Boeing's patent does not delve into exact technical detail of how the UAVs will land or otherwise make physical contact with their device. This is one of the feature problems the Senior Design Project must resolve.

On 01-07-2010, International Business Machines Corp (IBM) published US patent number 20100004802A1 to establish a process for navigating a UAV using an on-board digital

camera. While the patent is still pending, IBM's case consists of using machine vision to create a resolution of the UAV's environment to allow computers to navigate the UAV based on this artificial vision [1]. Again, this patent was vaguely written in order to engulf many applications, but this Senior Design Project will require machine vision on-board the UAV to determine collision detection, landing, flight patterns, etc. Part of this project will be to develop a method of machine vision to guide the UAV at times thus making this patent relevant.

Verizon Patent and Licensing Inc. published US patent number 20150336669A1 on 11-26-2015 creating both a physical and digital network of UAVs and recharging base stations. Verizon has developed a method that allows UAVs to land on a power station and calculate how far they can fly until they need to land at another power station. Verizon has created a derivative of the pony express using UAVS, and they have also created battery and GPS software to determine what power stations the UAVs can reach [3]. This grant is relevant to the Senior Design Project because it covers software that UAVs utilize to determine where a UAV needs to fly and when to successfully have its power bank replenished. While this patent would cover large scale application of ground stations, the project does not require the network of stations described in the patent.

The next patent belongs to Amazon Technologies Inc. Published on 06-28-2016, US patent number 9376208B1 outlines an on-board redundant power source for UAVs. With this grant, Amazon owns the intellectual property rights to any type of UAV that utilizes hybrid power generation techniques (such as batteries and electricity-generating gas engines) [2]. While this patent is not directly relevant to the technical scope of the project, this grant confirms the corporate interest in UAV mission time elongation, the main focus of the Senior Design Project.

The final patent, US patent number 9440545B2 granted to SZ DJI Technology Co Ltd demonstrates a system and method for autonomously exchanging batteries in a UAV on a ground station. This patent consists of robotics on a ground station that can both charge a UAV's batteries or replace them with freshly powered batteries [5]. This directly relates to the scope of the Senior Design Project as autonomous power transfer will be critical in flight and mission elongation. This project does, however, encompass a larger scope than what this intellectual property covers as the UAVs will need to determine flight paths, task management, and navigation.

3.3.2 IEEE Explore Articles

Searching IEEE Explore yielded several articles on the issues of drone power and the growing interest in drone-based services, such as parcel delivery.

The first article found, entitled "Endless Flyer: A Continuous Flying Drone with Automatic Battery Replacement," details the use of drones for aerial monitoring and surveying purposes and elaborates the problem of drone power and various methods of enabling indefinite flight, such as wired drones or laser power. The experiment catalogued in this article entails having a drone land on a battery swapping station automatically when the drone's battery is close to being depletion. The automated landing of the drone is achieved using an Optitrack motion capturing system surrounding the battery station to communicate to the drone where to land [6].

The second article, "SAN: Self-Adaptive Navigation for Drone Battery Charging in Wireless Drone Networks," gives a brief overview into why research into drone batteries is more prevalent than other forms of drone power. Afterwards, the authors propose a method for drones located within a network of what the authors dubbed "Quick Battery Charging Machines," or

QCMs, to be able to navigate to a recharging station before the drone's battery is depleted. Similar to other methods, this requires a land-based battery swapping station [7].

Lastly, "How Reliable Does a Delivery Drone Have to Be?" is an article that discusses both the societal and technical problems that engineers have in integrating drones into commercial services. Of particular note, the author discusses several possibilities in which a drone used for package delivery causes harm to passersby or even local wildlife due to hardware malfunctions or power failure. The insight provided by the author in this article is particularly useful when thinking about market and safety requirements during design time as well as creating favorable public perception for new products [8].

Also included with these three papers is a short *IEEE Spectrum* article discussing several recent events in drone development. After updates on large companies Amazon and Google's interest in drone research are mentioned, the article shifts its focus onto PrecisionHawk, a company in Raleigh, N.C. developing an air traffic control scheme for drones. This reference has been included to demonstrate that the design proposed here is in a modern and highly relevant field [9].

One of the Articles found using IEEE Explore includes "A Simple Equivalent Circuit for Efficiency Calculation of Brushless DC Motor". This article highlights the different losses associated with brushless DC motors and how the efficiency and other aspects are affected because of the losses. The different losses include copper loss, eddy current loss, hysteresis loss in the motor and friction loss. An equivalent circuit model is used to show the impedances relating to the different losses. From the equivalent circuit model, different parameters of the motor may be found as well as the motor efficiency at different motor speeds. This article will be

helpful with modeling the motors and determining the efficiency of the motors at various rates [10].

Another article researched was “High Altitude UAV Navigation using IMU, GPS and Camera”. This article discusses obtaining a more accurate estimation of the location of the UAV using measurements obtained from inertial sensors onboard the UAV integrated with a GPS and onboard camera. The article also brings to attention the many advantages of having an onboard camera which includes higher data rate and being able to see where the UAV is when a GPS signal is not available. By becoming more precise with the location of the UAV, operators will be able to have better control over the UAV [11].

One of the major issues of flying quadcopters is the longevity of the system to stay aloft. In the paper entitled “Minimum-Energy Path Generation for a Quadrotor UAV” depicts efficient flight algorithm-level correction methods that use the least amount of energy. The crucks of their work rely on the untapped potential of more efficient movement of quadcopter from point A to B with the minimum amount of energy and constant energy scenarios. They mention that improved modeling will include not only brushless DC Motors, but motor controller (ECS) and microcontrollers [12].

Along a similar vein “Intelligent Agent Based Power Management for UAV Systems” outlines the use of an intelligent power allocation system that can respond quickly and easily to different fault scenarios. These scenarios include, Loss of Fuel, Evasive Maneuver, Power Shortage, Generator Fault and Load Fault. Of these scenarios, peak interest is taken in all cases except loss of fuel and generator fault because the system will not have gas or generators. For the

longevity of the autonomous system, systems that enable fault recovery for long distance fight must be included [13].

The final article, “Smart on-board UAV system: Using computer vision system to find a movable and stationary target,” explains the method and suggested devices to track both moving and stationary objects. This will be one of the most critical systems to enable autonomous fight. The flow chart for this system goes into detail. The simplified version of this system takes visual information, determines coordinates of the target object in relation to the UAV and then relays that information to the fight controller at a standard refresh rate. The refresh rate will ensure that any course correction can be completed as the UAV approaches the target [14].

3.3.3 Other Articles

The paper “Automated Batter Swap and Recharge to Enable Persistent UAV Missions” depicts a system replace drone batteries and maintain a squad of unmanned autonomous vehicles (UAV) quadcopters. In short, the paper discusses mechanical, electrical, and software solutions needed to keep a squad of quad copters aloft for extended periods of time. The mechanical solution includes a landing pad and rotating carousel for battery storage. The electrical solution includes charging contacts to prevent quadcopter shutdown and automated battery charging. The software solution includes computer vision and rotor control to locate the landing zone and to maneuver to the landing spot appropriately. Electrical and software solutions in the project will allow for complete end-to-end autonomy [15].

The paper “Real-Time Improved Power Management for Autonomous Systems” dives into the use of a smart system to allocate power to the systems that most need power at any given point. By comparing power sink requirements to optimize supply from power sources. This intelligent grid could be used to optimize what systems need to be on at a given moment. Real-Time power management could allow us to employ a hybrid system using other power sources other than batteries [16].

The article “Design Space Exploration of Drone Infrastructure for Large-Scale Delivery Services” discusses the use of UAVs in areas outside military surveillance. Specifically the efficiency of using drones to deliver packages. The article discusses how the operation costs and design parameters compare to the efficiency and profitability. The core of the design must be efficiency; otherwise the purpose will go unfulfilled [17].

3.4 Marketing Requirements – Definitions

To establish the marketing requirements of the project, definitions of key terminology and nomenclature are described below.

- The "**worker UAV**" is defined as an unmanned aerial vehicle designed to perform user-generated tasks (i.e. hovering in place, delivering packages, filming, etc.).
- The term, "**service**" is defined as the autonomous robotic process of replacing a depleted battery from a UAV with a fully charged battery.
- The "**battery UAV**" is defined as an unmanned aerial vehicle designed to carry numerous batteries and service a worker UAV mid-flight.

3.4.1 Marketing Requirements

1) The battery UAV shall locate a worker UAV using machine vision and other forms of wireless communication.
2) The battery UAV must be able to exchange a depleted battery with a charged battery on the worker UAV while mid-flight.
3) The battery UAV must be able to carry at least two worker UAV batteries.
4) The worker UAV and battery UAV shall be able to communicate their battery voltage levels to other devices related to the project.
5) The UAVs will have necessary protocols to ensure safe landing in the event that a system failure occurs.
6) The battery exchange will occur while both drones have power maintained. Neither drone will lose power during the exchange.
7) The UAVs must be operable indoors.
8) The UAVs will have autonomous autopilot functions for basic flight routines.

3.5 Objective Tree

Figure 01 is an illustrated form of the marketing requirements detailed in Section 1.4. The diagram is divided into three columns that represent the requirements for each functional device in the project.

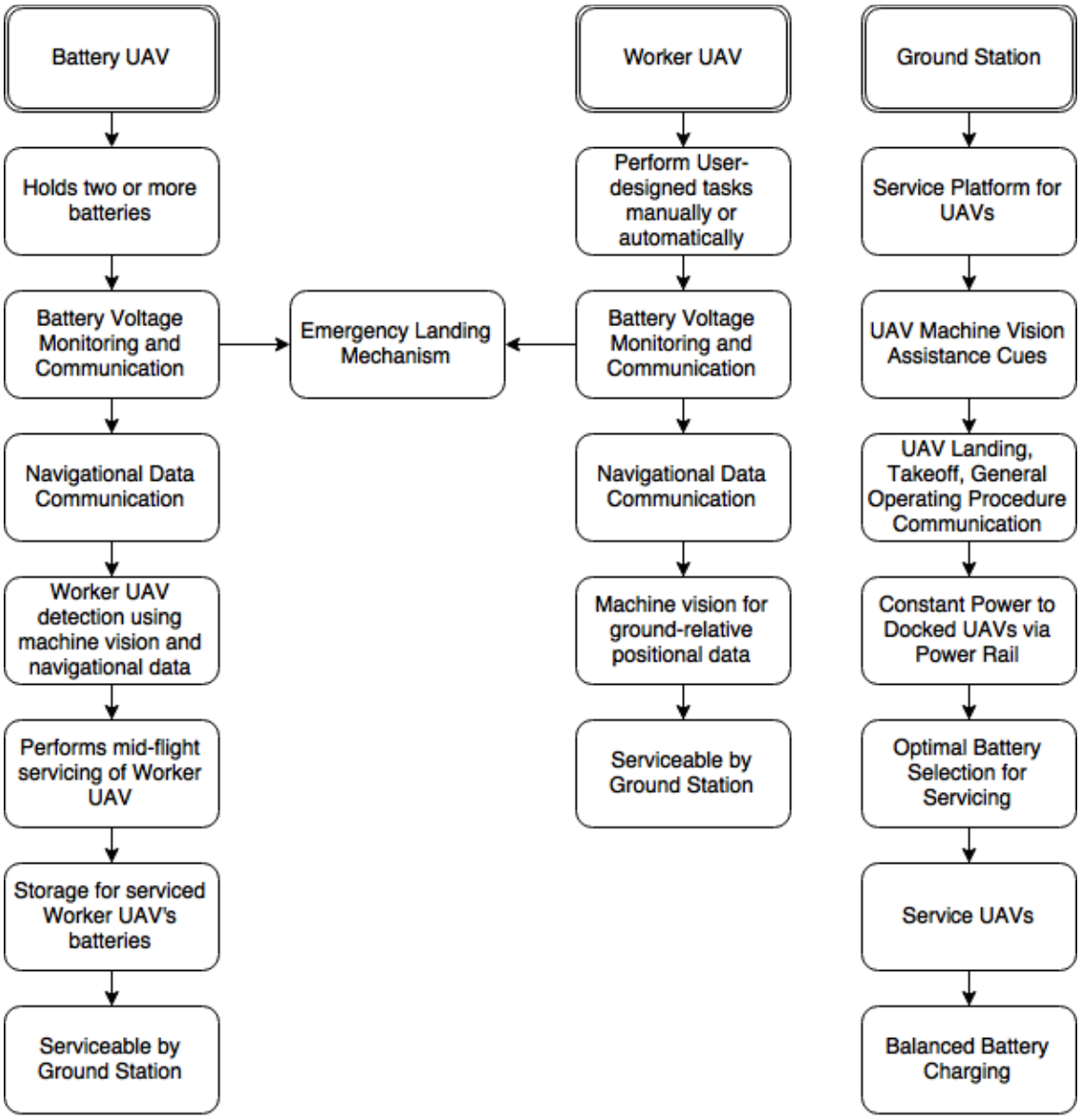


Figure 01 – Objective Tree [ND]

4.0 Design Requirements Specification

4.1 Design Requirements Table

	Engineering Requirement	Justification
1	The battery UAV will be able to align itself vertically with a worker UAV (z axis) to within a five cm magnitude margin of error in the x, y, and z directions.	Five cm is the maximum range for error that the magnetic tether can compensate for. (MR: 1)
2	Once aligned with the worker UAV, the battery UAV will perform the exchange in under 60 seconds.	Fast battery exchanges are critical to yield a net power benefit to the worker UAV. (MR: 2,3)
3	The battery UAV will operate with autonomy only when both the user relinquishes control and the battery UAV has vision of the worker UAV.	This check provides a level of redundancy in the safety protocols to ensure safe operating conditions. (MR: 1, 5)
4	The battery UAV will have capacity to carry at least two batteries related to the worker UAV.	The battery UAV needs to be able to deliver at least one battery and extract at least one battery. (MR: 2,3)
5	Both UAVs will have the ability to operate without a GPS.	Much of the testing is performed indoors where GPS signal cannot be relied on. (MR: 7)
6	Both UAVs will communicate their voltage levels wirelessly to a ground station within a 10m range.	This ensures the engineers can deploy the battery UAV at the proper times. (MR: 4, 5)
7	Both UAVs will maintain stable flight in autonomous fashion.	Having numerous drone operators creates needless points for failure. (MR: 8)
8	When either UAV's main battery voltage falls below a critical power level, an emergency landing routine is performed by the UAV.	This ensures the UAVs do not discharge a battery to an unsafe level. (MR: 5)
9	Batteries used in the system will have necessary protections against shorts and other hazardous situations.	Shorting batteries can yield catastrophic failure and unsafe conditions for users and the UAVs. (MR: 2, 5, 6)
10	Power in the worker UAV is maintained at all phases of the battery swapping process. No UAV's power will be lost during the battery exchanges.	Hot swapping is important for the project so data related to the worker UAV's mission is not lost. (MR: 2, 6)

Table 01 – Design Requirements [NR]

4.2 Hardware Level 0

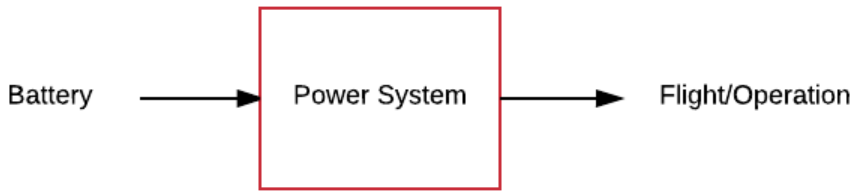


Figure 02 – Hardware Level 0 Block Diagram

Module	Power Management System Level 0
Designer	Walker Grossman
Inputs	Battery
Outputs	Flight/Operation
Theory of Operation	The battery will supply power to all necessary components for proper flight and functionality.

Table 02 – Level 0 Block Diagram [WG]

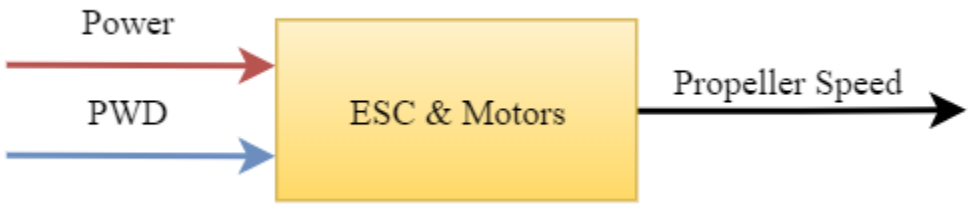


Figure 03 - ESC & Motors Level 0 Block Diagram [CC]

FR-Table	
Module	ESC & Motors
Designer	Reed Jacobsen
Input	Power PWD
Output	4x Propeller Speed
Theory of Operation	The flight controller board feeds Power and PWD signal for the ESC that Determines the propeller speed

Table 03 - ESC & Motor Level 0 FR Table [CC]

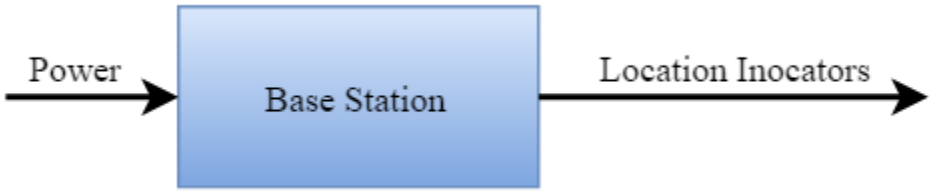


Figure 04 – Base Station Level 0 Block Diagram [CC]

FR-Table	
Module	Base Station
Designer	Creighton Cloud
Input	Power
Output	Location indicators
Theory of Operation	Acts as a dummy drone for battery swap drone to practice autonomous approaches

Table 04 – Base Station Level 0 FR Table [CC]

4.3 Hardware Level 1

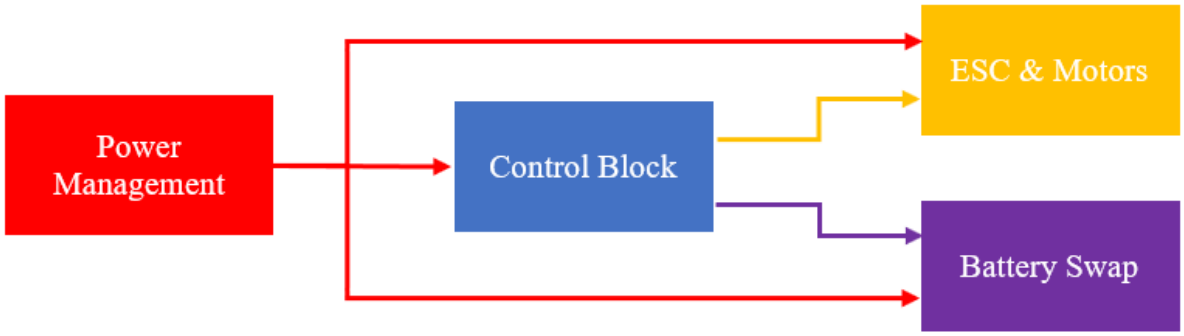


Figure 05 – Hardware Block Diagram Level 1 [CC]

FR-Table	
Module	Power Management
Designer	Walker Gross
Input	Battery Power
Output	Regulated Voltage
Theory of Operation	Regulates battery voltage down to required voltages for any device on the quadcopter.

Table 05 – Level 1 FR Table [CC]

FR-Table	
Module	Control Block
Designer	Nathan Dornback Nikolai Ruhe Reed Jacobsen
Input	Power
Output	ESC & Motor PWD Battery Swap Control
Theory of Operation	The control block is the main decision maker. It reads information from sensors and controls the ESC & Motors and Battery Swap mechanism.

Table 06 – Level 1 FR Table [CC]

FR-Table	
Module	ECS & Motors
Designer	Reed Jacobsen
Input	Power Control PWD
Output	Propeller Speed
Theory of Operation	Controls the propeller speed based on the input of the control block.

Table 07 – Level 1 FR Table [CC]

FR-Table	
Module	Battery Swap
Designer	Creighton Cloud Walker Grossman
Input	Power Control Block
Output	Battery Transfer
Theory of Operation	A system of mechanical and electrical components that control and sense the process of battery transferring.

Table 08 – Level 1 FR Table [CC]

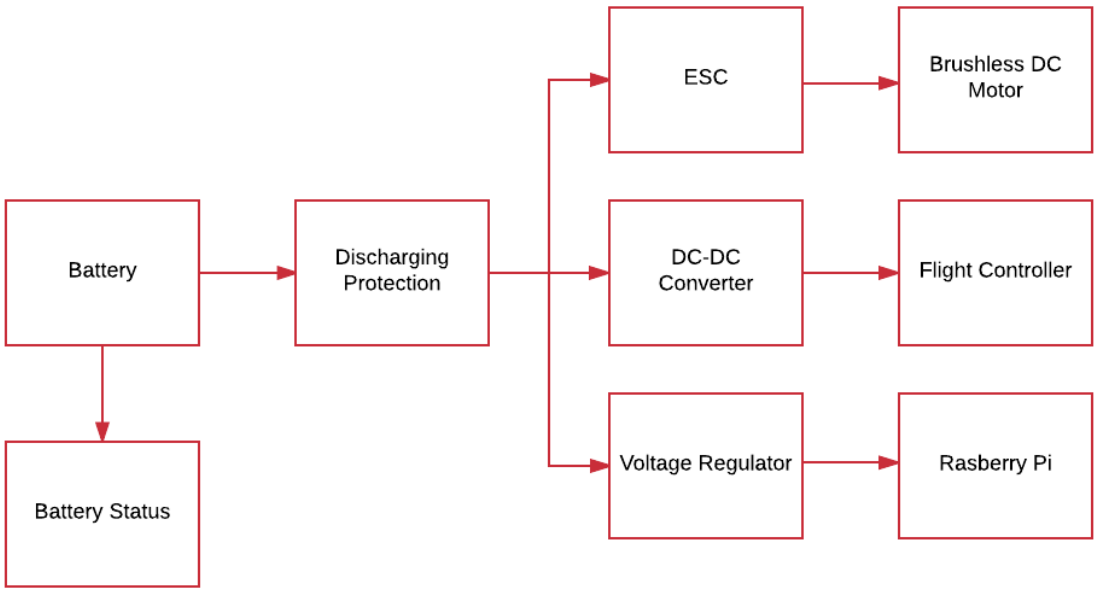


Figure 06 - Power Management System Level 1 Block Diagram

Module	Power Management System Level 1
Designer	Walker Grossman
Inputs	Battery
Outputs	Motor Operation Flight Controller Operation Raspberry Pi Operation
Theory of Operation	Utilizing pre-equipped ESC's and constructing a DC-DC Converter and voltage regulator, the flight controller, motors and raspberry Pi shall operate using power supplied by the battery.

Table 09 – Power Management Level 1 FR Table [WG]

4.4 Hardware Level 2

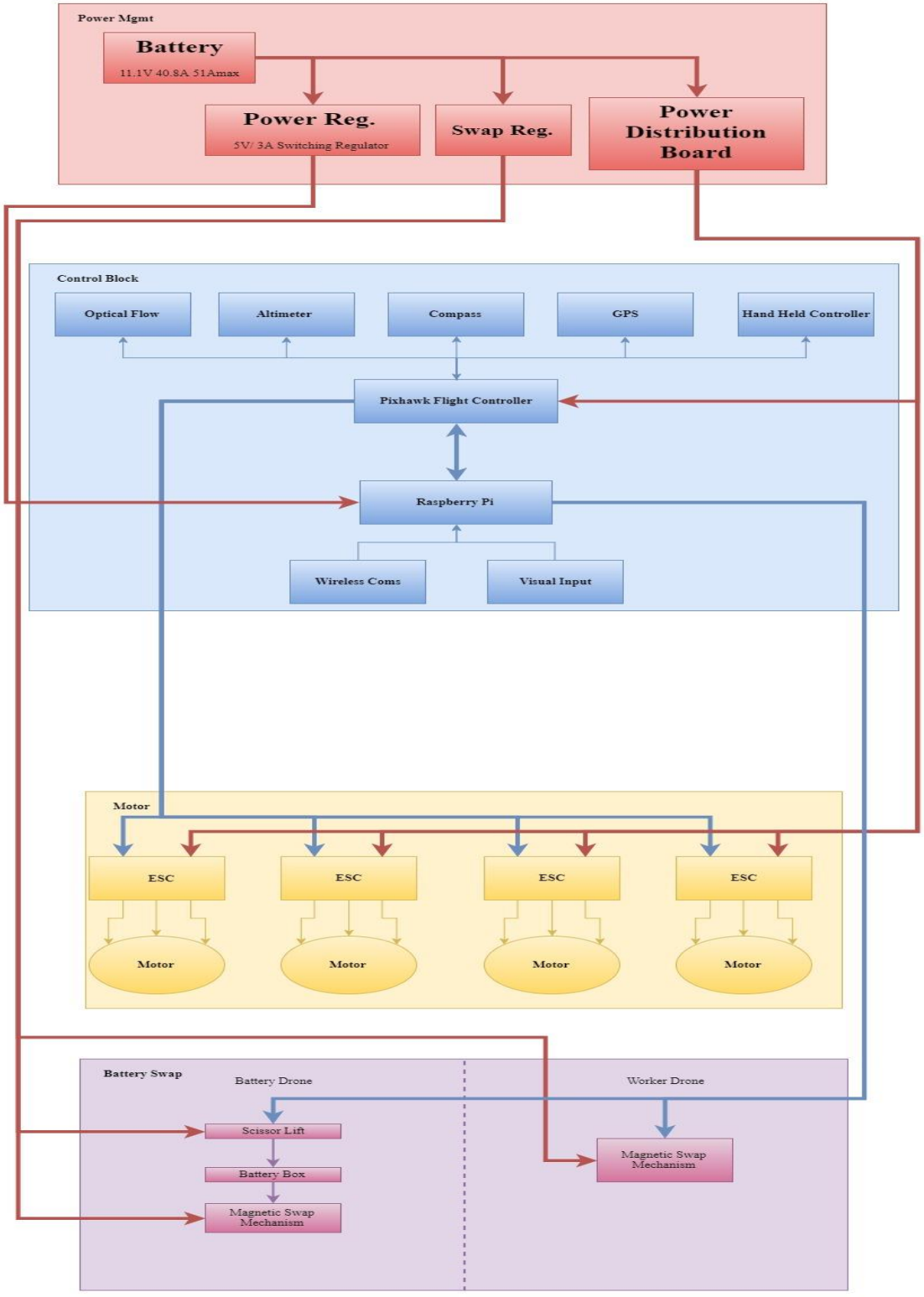


Figure 07 – Hardware Level 2 Block Diagram [CC]

POWER

FR-Table	
Module	Power Regulator
Designers	Walker Grossman Creighton Cloud
Input	Battery 11.1 V
Output	Raspberry Pi
Theory of Operation	Linear regulator to step down the input voltage of 11.1V to 5V to support the Raspberry Pi and visual input circuit

Table 10 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Swap Regulator
Designers	Walker Grossman Creighton Cloud
Input	Battery 11.1 V
Output	Supporting Power for swap electrical devices
Theory of Operation	Swap Regulator supplies the required voltages and power to all of the electrical devices attached to swap mechanism.

Table 11 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Power Distribution Board
Designers	Walker Grossman Creighton Cloud
Input	Battery 11.1 V
Output	Pixhawk Flight Controller ESC
Theory of Operation	Supplies power to all ECS, motors, and PixHawk Flight controller.

Table 12 – Hardware Level 2 FR Table [CC]

CONTROL BLOCK

FR-Table	
Module	Raspberry Pi
Designers	Pre-existing hardware
Input	5V Power Regulator Wireless Coms Visual Input
Output	Pixhawk Flight Controller Battery Swap
Theory of Operation	The Raspberry Pi is the logic controller of the drone. It reads state information from the PixHawk flight controller, wireless coms, and visual input. It then does calculations to control the drone to match the current desired state.

Table 13 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Wireless Coms
Designers	Internal Component
Input	Wireless signals
Output	Command Information Software updates
Theory of Operation	Allows for wireless access to internal flight control software and allows for the sending and receiving of flight data and software updates.

Table 14 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Visual Input
Designers	Pre-existing hardware
Input	Visual information
Output	Digital Coms indicating flight information
Theory of Operation	The Visual Input is a fine tuned method of determining the position and altitude of the drone relative to the ground and other drones in the area.

Table 15 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	PixHawk Flight Controller
Designers	Pre-existing hardware
Input	Optical Flow Altimeter Compass GPS Hand Held Controller
Output	ECS
Theory of Operation	The Flight Controller stabilizes and moves the drone with internal logic by reading drone pitch and altitude information from the optical flow sensor, altimeter, compass, GPS, and hand held controller.

Table 16 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Optical Flow
Designers	Pre-existing hardware
Input	Visual information
Output	Flight controller
Theory of Operation	Relays altitude information to verify information from the altimeter.

Table 17 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Altimeter
Designers	Internal hardware
Input	Altitude statistics
Output	PixHawk flight Controller
Theory of Operation	Relays altitude information to PixHawk controller

Table 18 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	GPS
Designers	Internal hardware
Input	satellite positioning information
Output	PixHawk flight controller
Theory of Operation	Informs the PixHawk fight controller on current position.

Table 19 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Hand Held Controller
Designers	Pre-existing hardware
Input	Human bystander
Output	PixHawk Flight Controller
Theory of Operation	Allows for a safety operator to take control of the drone if it breaks protocol or expected behavior.

Table 20 – Hardware Level 2 FR Table [CC]

MOTOR

FR-Table	
Module	ESC
Designers	Pre-existing hardware
Input	Power PixHawk Flight Controller
Output	Motors
Theory of Operation	Controls motor speed by PWD.

Table 21 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Motor
Designers	Pre-existing hardware
Input	ESC
Output	Propeller Speed
Theory of Operation	The flight controller board feeds power and PWD signal to the ESC that determines the propeller speed.

Table 22 – Hardware Level 2 FR Table [CC]

BATTERY SWAP

FR-Table	
Module	Scissor Lift
Designers	Walker Grossman Creighton Cloud
Input	Power Raspberry Pi
Output	Battery Box
Theory of Operation	Mechanical system used to lower the battery box down to the worker drone.

Table 23 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Battery Box
Designers	Walker Grossman Creighton Cloud
Input	Scissor Lift
Output	Magnetic Swap Mechanism
Theory of Operation	Container to hold battery and allow for better control of battery and electrical connections

Table 24 – Hardware Level 2 FR Table [CC]

FR-Table	
Module	Magnetic Swap Mechanism
Designers	Walker Grossman Creighton Cloud
Input	Battery Box
Output	Swapped Battery
Theory of Operation	Electro-mechanical method to connect the battery box to the scissor lift and the worker drone.

Table 25 – Hardware Level 2 FR Table [CC]

4.5 Software Level 0

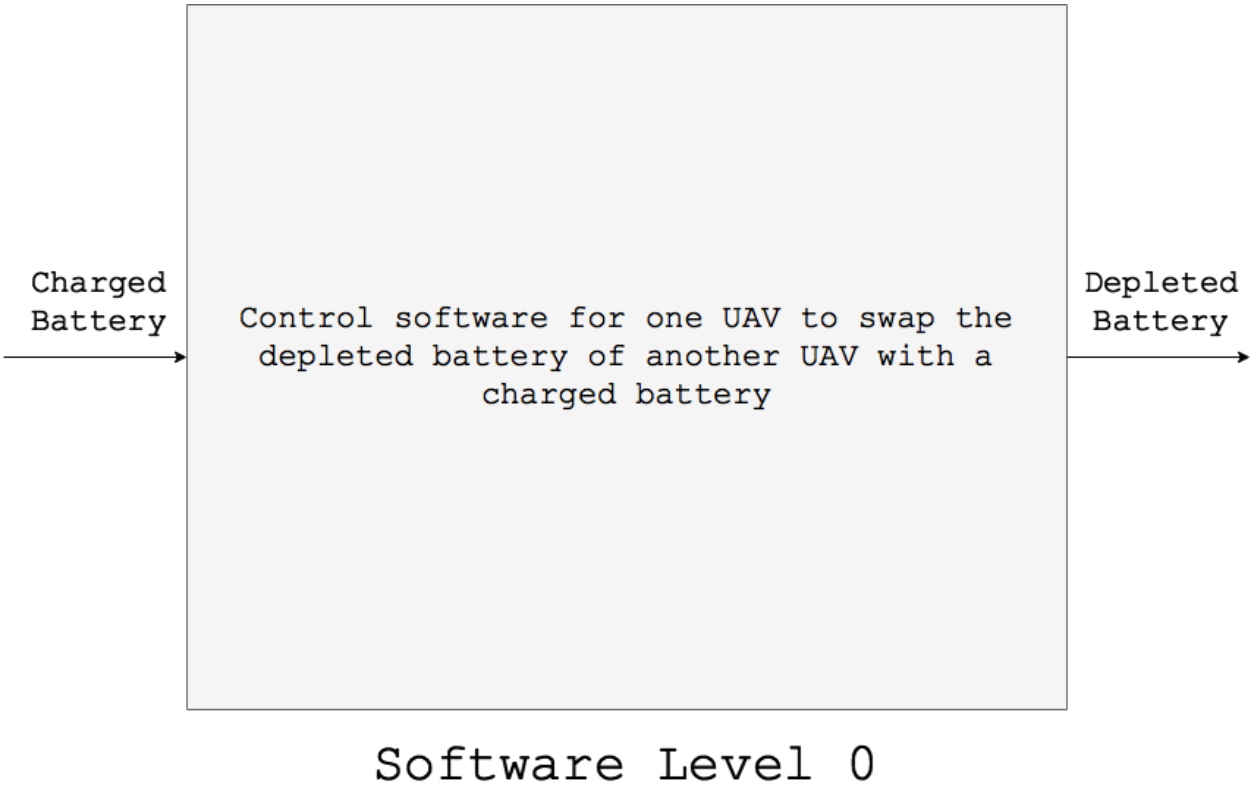


Figure 08 – Level 0 Software Block Diagram

Module	Control Software for 2 UAVs
Designer	Nathan Dornback, Reed Jacobsen, Niko Ruhe
Inputs	Charged Battery
Outputs	Depleted Battery
Theory of Operation	A depleted battery will be replaced by a charged battery in this single system view.

Table 26 – Level 0 FR Table [ND]

4.6 Software Level 1

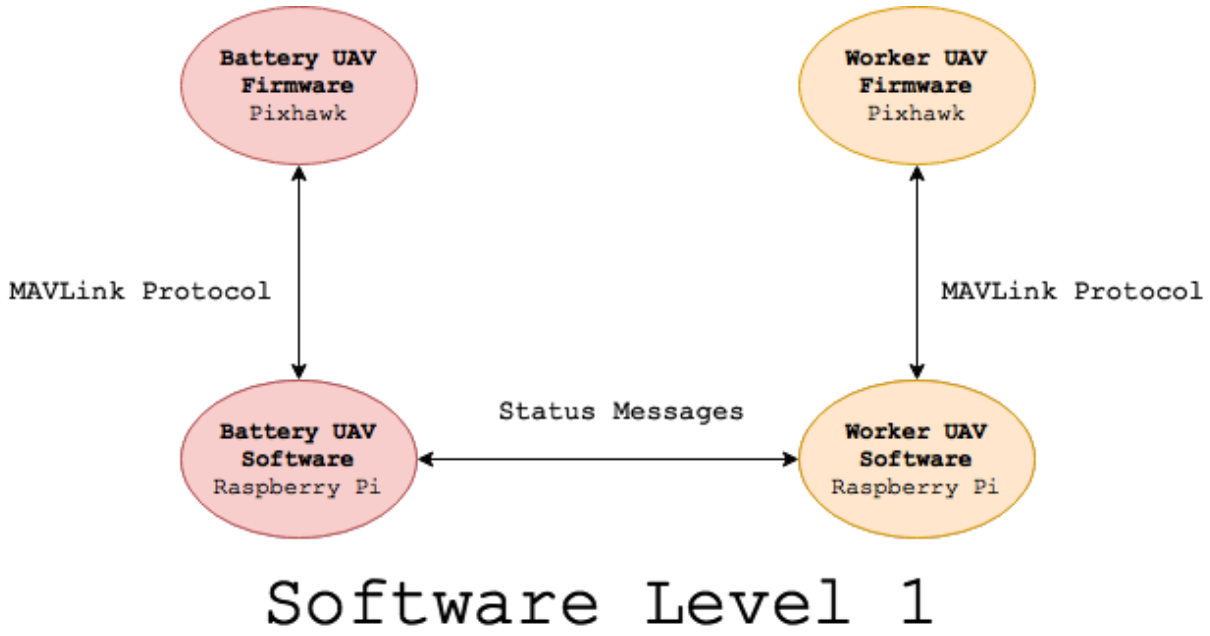


Figure 09 – Level 1 Software Block Diagram [ND]

Module	Battery UAV Software
Designer	Nathan Dornback, Reed Jacobsen
Inputs	MAVLink messages from Battery UAV's Pixhawk microcontroller Messages to the Worker UAV regarding battery swap information
Outputs	MAVLink messages to Battery UAV's Pixhawk microcontroller Messages to the Worker UAV regarding battery swap information
Theory of Operation	This software encompasses all of the software running on the Raspberry Pi that is associated with the Battery UAV. This software will determine and perform the physical actions necessary for this Battery UAV to successfully perform a battery swap on a target Worker UAV.

Table 27 – Level 1 FR Table [ND]

Module	Battery UAV Firmware
Designer	Integrated by Nathan Dornback
Inputs	MAVLink Commands
Outputs	Flight control data packaged into the MAVLink protocol
Theory of Operation	This is the flight controller for the Battery UAV that handles all mission critical components, such as motors, barometer, altimeter, etc. The Dronekit API is used by the battery UAV Software to interact with this firmware to obtain sensor data.

Table 28 – Level 1 FR Table [ND]

Module	Worker UAV Software
Designer	Nathan Dornback, Reed Jacobsen
Inputs	MAVLink messages from Worker UAV's Pixhawk microcontroller Messages to the Worker UAV regarding battery swap information
Outputs	MAVLink messages to Worker UAV's Pixhawk microcontroller Messages to the Worker UAV regarding battery swap information
Theory of Operation	This software encompasses all of the software running on the Raspberry Pi that is associated with the Worker UAV. This software will determine and perform the physical actions necessary for a target battery UAV to successfully perform a battery swap on the Worker UAV whose Raspberry Pi is running this software.

Table 29 – Level 1 FR Table [ND]

Module	Worker UAV Firmware
Designer	Integrated by Nathan Dornback
Inputs	MAVLink Commands
Outputs	Flight control data packaged into the MAVLink protocol
Theory of Operation	This is the flight controller for the Worker UAV that handles all mission critical components, such as motors, barometer, altimeter, etc. The Dronekit API is used by the Worker UAV Software to interact with this firmware to obtain sensor data.

Table 30 – Level 1 FR Table [ND]

4.7 Software Level 2

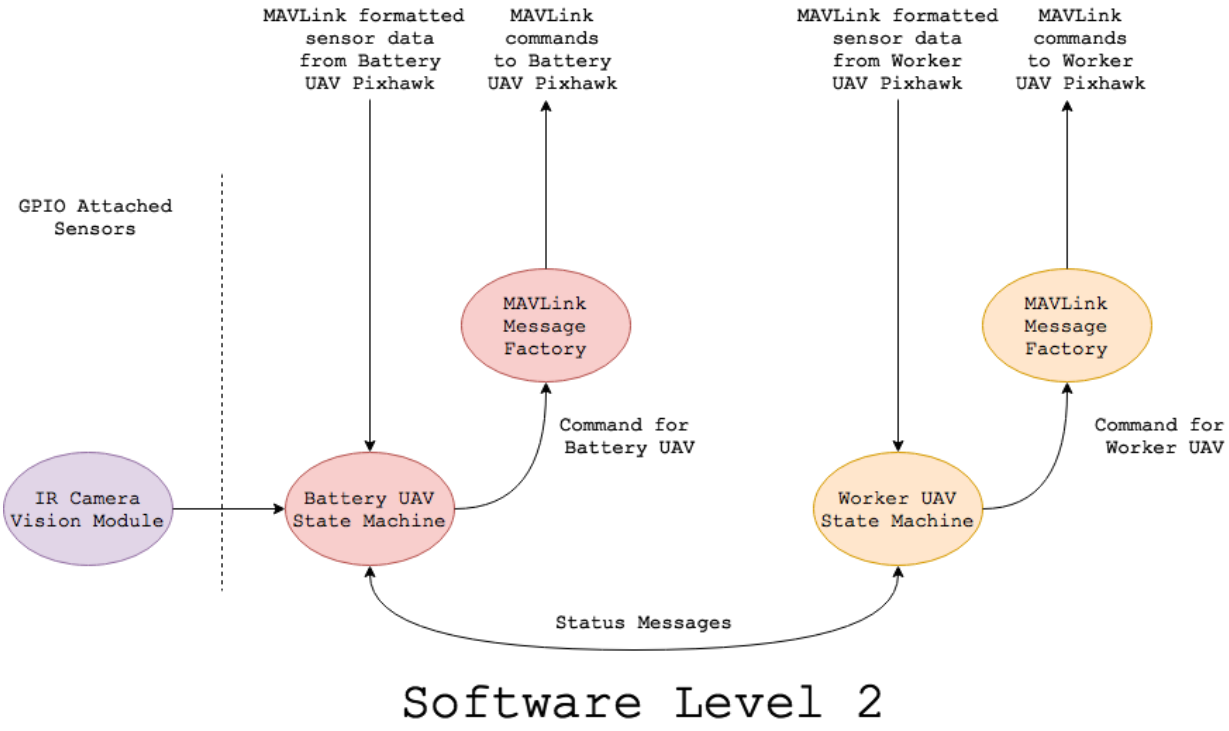


Figure 10 – Level 2 Software Block Diagram [ND]

FR Table	
Module	IR Camera Vision Module
Designer	Reed Jacobsen
Input	Data from Camera over I2C Requests for data
Output	If target is visible or not Target position Target direction Distance to target
Theory of Operation	This is a low level software library that allows the companion computer on the battery UAV to see the worker UAV using the IR camera. The data from this module should allow the battery UAV to determine the distance from the worker UAV and the direction it is facing.

Table 31 – Level 2 FR Table [RJ]

Module	MAVLink Message Factory
Designer	Open source, integrated by Nathan Dornback
Inputs	A logical command with sufficient data to be converted to a MAVLink command
Outputs	MAVLink command
Theory of Operation	This module is an object provided by the DroneKit open source library to format messages into a format that can be interpreted by PixHawk microcontrollers.

Table 32 – Level 2 FR Table [ND]

Module	Battery UAV State Machine
Designer	Nathan Dornback
Inputs	Infrared Camera data mounted on Battery UAV frame. Sensor data from Battery UAV Pixhawk
Outputs	A logical command with sufficient data to be converted to a MAVLink command.
Theory of Operation	This software handles inputs from the Battery UAV PixHawk and IR Camera Vision Module and sends logical commands to the MAVLink Message Factory based on these inputs.

Table 33 – Level 2 FR Table [ND]

Module	Worker UAV State Machine
Designer	Nathan Dornback
Inputs	Sensor data from Worker UAV Pixhawk
Outputs	A logical command with sufficient data to be converted to a MAVLink command
Theory of Operation	This software handles inputs from the Worker UAV PixHawk and sends logical commands to the MAVLink Message Factory.

Table 34 – Level 2 FR Table [ND]

4.8 Software Level 3

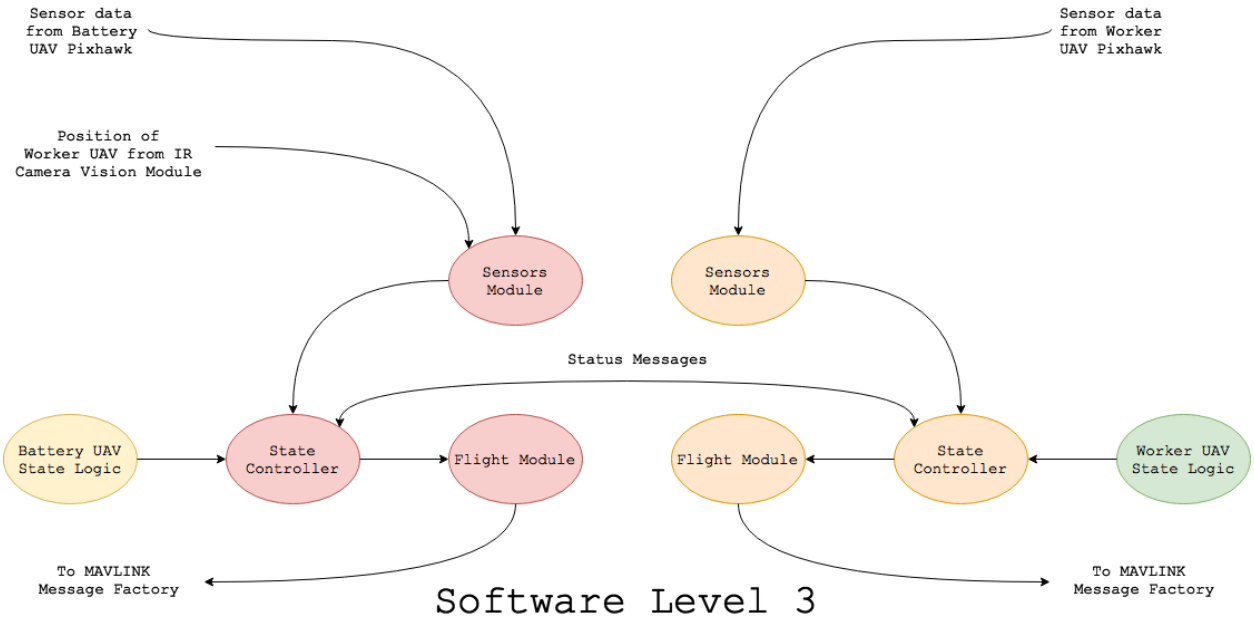


Figure 11 – Level 3 Software Block Diagram [ND]

Module	Battery UAV State Controller
Designer	Nathan Dornback
Inputs	Battery UAV State Logic Battery UAV Diagnostics Module data Battery UAV Flight Module data Battery UAV Communication Module data
Outputs	Battery UAV State
Theory of Operation	This module uses sensor data from the Battery UAV Sensors Module and previous state information to determine the next state of the Battery UAV system.

Table 35 – Level 3 FR Table [ND]

Module	Battery UAV Sensors Module
Designer	Nathan Dornback
Inputs	Worker UAV position Battery UAV Pixhawk sensor data
Outputs	State relevant sensor data
Theory of Operation	This module aggregates sensor data and filters state relevant data into a structure that can be used by the Battery UAV State Controller along with Battery UAV State Logic to determine the state of the Battery UAV software.

Table 36 – Level 3 FR Table [ND]

Module	Battery UAV Flight Module
Designer	Nathan Dornback
Inputs	Battery UAV Machine State
Outputs	Flight commands for the Battery UAV Pixhawk
Theory of Operation	This module determines what commands to send to the Communication module based on the current state of the Battery UAV and then sends the command to the MAVLink Message Factory to be converted to a MAVLink command.

Table 37 – Level 3 FR Table [ND]

Module	Worker UAV State Controller
Designer	Nathan Dornback
Inputs	Worker UAV State Logic Worker UAV Diagnostics Module data Worker UAV Flight Module data Worker UAV Communication Module data
Outputs	Worker UAV State
Theory of Operation	This module uses sensor data from the Worker UAV Sensors Module and previous state information to determine the next state of the Worker UAV system.

Table 38 – Level 3 FR Table [ND]

Module	Worker UAV Sensors Module
Designer	Nathan Dornback
Inputs	Worker UAV Pixhawk sensor data
Outputs	State relevant sensor data
Theory of Operation	This module aggregates sensor data and filters state relevant data into a structure that can be used by the Worker UAV State Controller along with Worker UAV State Logic to determine the state of the Worker UAV software.

Table 39 – Level 3 FR Table [ND]

Module	Worker UAV Flight Module
Designer	Nathan Dornback
Inputs	Worker UAV Machine State
Outputs	Flight commands for the Worker UAV PixHawk
Theory of Operation	This module determines what commands to send to the Communication module based on the current state of the Worker UAV and then sends the command to the MAVLink Message Factory to be converted to a MAVLink command.

Table 40 – Level 3 FR Table [ND]

4.9 Mechanical Level 0

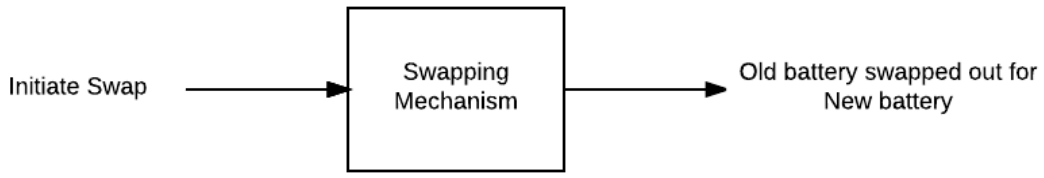


Figure 12 – Mechanical Level 0 Block Diagram [WG]

Module	Swapping Mechanism Level 0
Designer	Walker Grossman/Creighton Cloud
Inputs	Signal To Initiate Swap
Outputs	Spent Battery Swapped Out For Fresh Battery
Theory of Operation	Once the system is ready for the battery swap the swapping mechanism autonomously changes out the battery of UAV

Table 41 – Level 0 FR Table [WG]

4.10 Mechanical Level 1

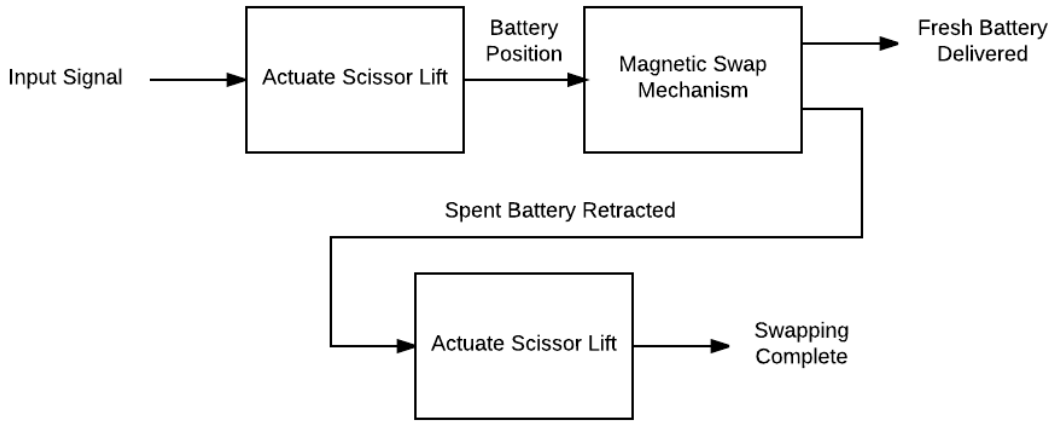


Figure 13 – Level 1 Mechanical Block Diagram [WG]

FR-Table	
Module	Swap Mechanism
Designers	Walker Grossman Creighton Cloud
Input	Initiate Swap Command
Output	Batteries are swapped
Theory of Operation	Swap mechanism positions the fresh battery and retrieves the spent battery.

Table 42 – Swap Mechanism FR Table [CC]

FR-Table	
Module	Actuate Mechanical Linkage
Designers	Walker Grossman Creighton Cloud
Input	Initiate Swap Command
Output	Position Battery
Theory of Operation	Mechanical linkage lowers fresh battery box into position and retrieves spent battery.

Table 43 – Actuate Mechanical Linkage FR Table [CC]

FR-Table	
Module	Magnetic Swap Mechanism
Designers	Walker Grossman Creighton Cloud
Input	Battery is in position
Output	Fresh battery is placed Spent battery is retrieved
Theory of Operation	Magnetically connects depleted battery and magnetically releases charged battery to be connected to worker UAV.

Table 44 – Magnetic Swap Mechanism [CC]

5.0 Accepted Technical Design

5.1 Hardware Design

5.1.1 Battery Management Calculations

The average battery life of any UAV is determined by its battery's capacity as well as the load that the battery is supplying current to. This can be seen by the relationship in Eq.1

$$\text{Average Battery Life (h)} = \frac{\text{Battery Capacity (mAh)}}{\text{Average Current Drawn (mA)}}$$

The problem with determining the average battery life is that the battery is not always guaranteed to last as long as the average battery life, so it is not the safest option with UAV technology. Finding the minimum battery life will be more appropriate as well as being the safer route for determining the life of the battery. Eq.2 shows how to determine the minimum battery life.

$$\text{Minimum Battery Life (h)} = \frac{\text{Battery Capacity (mAh)}}{\text{Max Continuous Current Drawn (mA)}}$$

With the given information we can determine the battery life at various throttle levels.

Battery capacity = 5,100mAh

Current drawn by on-board electronics = 3,000mA

Future current drawn by swapping mechanism = X

Battery Life:

Motor Current: @50% Throttle per motor = 2,000mA * 4(motors) = 8,000mA

@75% Throttle per motor = 5,500mA * 4(motors) = 22,000mA

@100% Throttle per motor = 9,800mA * 4(motors) = 39,200mA

The percent throttle will be determined from the gross weight of the drone and swapping mechanism combined. Thus adding the motor currents at different throttle ranges with the other currents being used, the battery life can be found at various throttle levels. The future current drawn "X" represents any future currents that may arise from future design additions. This value is set equal to 0 in the below calculations.

$$@50\%\ \text{Throttle} = \frac{5{,}100\text{mAh}}{11{,}000\text{mA}+X} = 27\text{min}$$

$$@75\%\ \text{Throttle} = \frac{5{,}100\text{mAh}}{25{,}000\text{mA}+X} = 12\text{min}$$

$$@100\%\ \text{Throttle} = \frac{5{,}100\text{mAh}}{42{,}200\text{mA}+X} = 7\text{min}$$

It can be seen that at 100% throttle the current drawn by the UAV is 42,200mA or 42.2A. However the continuous discharge rate of the battery is 40.8A and the max burst rate is 51A. This means that the drone is not able to sustain flight at 100% throttle and maintain all functionality. However it may peak at 100% throttle for a short time if necessary.

5.1.2) Voltage Regulation

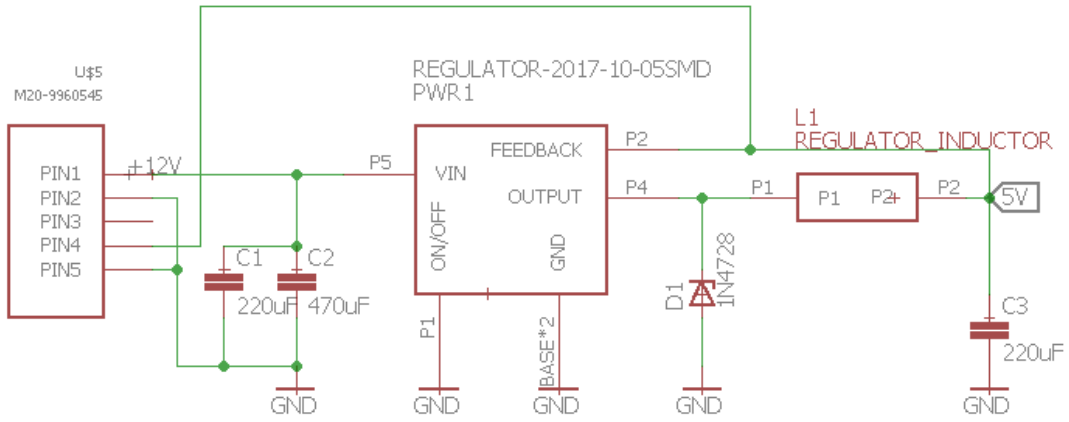


Figure 14 – Voltage Regulator [WG]

Module	Voltage Regulator
Designer	Walker Grossman
Inputs	11.1VDC
Outputs	5VDC
Theory of Operation	Supplying 11.1V to the voltage regulator will result in 5V at the output to operate the Raspberry Pi.

Table 45 – Voltage Regulator FR Table [WG]

5.2 Software Design

5.2.1 IR Camera Software Description

The IR camera is an I2C sensor recovered from a Nintendo Wii remote controller. It returns the four brightest IR points it sees with their respective X and Y coordinates. Additionally the brightness/diameter of each point can be returned. The purpose of this camera is

to recognize a target, allowing the battery drone equipped with this camera to locate the worker drone by finding the IR target on the top of the drone.

By making the target from 3 IR LEDs it is possible to verify that the correct target is acquired by comparing relative distances between the points seen by the camera. Then, the X,Y coordinate of the target can be found as either the coordinate of a single LED, or as the average of all LEDs in the target. Additionally, the target can be given a direction it points by making one of the LEDs dimmer and looking for the change in brightness relative to the other two LEDs. Finally, distance from the target can be determined by comparing the apparent distances between LEDs to their known distance.

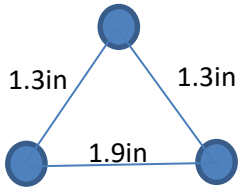


Figure 15: Example Target

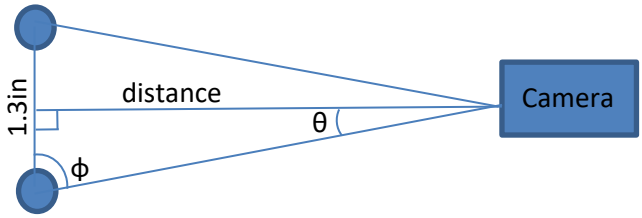


Figure 16: Distance to Target

5.2.2 IR Camera Pseudo-Code

```
//start
ObjectPositions[] = getObjectPositionsFromCam()
If (Number of Objects == 3)
        Target = found
        TargetPosition = AVG(ObjectPositions)
        TargetDirection = pointerLEDLocation – AVG(other LEDLocations)
        TargetDistance = GetDistance()
//end

GetDistance() {
        RelDistances[] = Mag(eachObject – eachOtherObject)
        minRelDistance = Min(RelDistances)
        Theat = minRelDistance*degPerPixel
        Phi = 90 – theta/2
        Return (knowDistance/2)*tan(phi) //know distance is min distance between LEDs on target
}
```

Figure 17 – IR Pseudo-Code [RJ]

5.2.3 IR Camera Test

```
from pyIRcam import pyIRcam
from time import sleep,time
from Tkinter import *

update_rate = 20# 20mS update rate

camera = pyIRcam() # Sensor initialization

master = Tk()
w = Canvas(master, width = 1024, height = 768) #camera dimmensions
w.pack(fill=BOTH, expand=1)

def update_Image():
   camera.getPositions() # Update found IR objects
   w.delete(ALL)
   if camera.positions['found']: # If an IR object is found, print the information
      camera.findTarget()
      if(camera.targetFound):
         color = "blue" #turn blue for target found
         camera.getDistance()
      else:
         color = "red" #otherwise red for no target
      w.create_rectangle(camera.positions['1'][0],camera.positions['1'][1],camera.positions['1']
[0]+10,camera.positions['1'][1]+10, fill=color)
      w.create_rectangle(camera.positions['2'][0],camera.positions['2'][1],camera.positions['2']
[0]+10,camera.positions['2'][1]+10, fill=color)
      w.create_rectangle(camera.positions['3'][0],camera.positions['3'][1],camera.positions['3']
[0]+10,camera.positions['3'][1]+10, fill=color)
      w.create_rectangle(camera.positions['4'][0],camera.positions['4'][1],camera.positions['4']
[0]+10,camera.positions['4'][1]+10, fill=color)
   master.after(update_rate, update_Image) #set this function up to be called in update_rate
mS later



master.after(0, update_Image)  # update_Image will run as soon as the mainloop starts.
master.mainloop()
```

Figure 18 – IR Camera Python Test Code [RJ]

5.2.4 IR Helper Library

```
#!/usr/bin/env python

"""pyIRcam.py: Module that talks to PixArt camera connected to the I2C bus. """

"""
Requirements:

- Positioning IR camera (http://www.dfrobot.com/wiki/index.php/Positioning_ir_camera), or PixArt WiiMote camera
- Raspberry Pi (or any computer with python capabilities and exposed I2C port)

"""

__author__ = "Aldo Vargas"
__copyright__ = "Copyright 2016 Altax.net"

__license__ = "GPL"
__version__ = "1"
__maintainer__ = "Aldo Vargas"
__email__ = "alduxvm@gmail.com"
__status__ = "Development"

import smbus
import math
from time import sleep
bus = smbus.SMBus(1)
degPerPixel = 0.0299947917 #0.0322265625
ledDistance = 1.3 #inches

class pyIRcam:
"""Class that handles the sensor communication"""
def __init__(self):
self.sensorAddress = 0x58 # Check before to be sure its the correct address
self.device = smbus.SMBus(1)
self.positions = {'found':False,'1':[0,0],'2':[0,0],'3':[0,0],'4':[0,0]}
self.targetFound = False
# Initialization of the IR sensor
self.initCMDs = [0x30, 0x01, 0x30, 0x08, 0x06, 0x90, 0x08, 0xC0, 0x1A, 0x40, 0x33, 0x33]
for i,j in zip(self.initCMDs[0::2], self.initCMDs[1::2]):
self.device.write_byte_data(self.sensorAddress, i, j)
sleep(0.01)

def getPositions(self):
self.device.write_byte(self.sensorAddress, 0x36)
```

```
data = self.device.read_i2c_block_data(self.sensorAddress, 0x36, 16) # Read the data from the I2C bus
x = [0x00]*4
y = [0x00]*4
i=0
for j in xrange(1,11,3): # Decode the data coming from the I2C bus
x[i]=data[j]+((data[j+2] & 0x30) << 4)
y[i]=data[j+1]+((data[j+2] & 0xC0) << 2)
i+=1
i=0
for j in ('1','2','3','4'): # Put the x and y positions into the dictionary
self.positions[j][0]=x[i]
self.positions[j][1]=y[i]
i+=1
if ( all(i == 1023 for i in x) and all(i == 1023 for i in y) ): # If all objects are 1023, then there is no IR object in front of the sensor
self.positions['found'] = False
else:
self.positions['found'] = True

def findTarget(self):
        counter = 0
        for j in ('1','2','3','4'):
           if(self.positions[j][0] == 1023 or self.positions[j][1] == 1023):
              counter+=1
        if(counter == 1): #if only one position unoccupied we have a target
           self.targetFound = True
        else:
           self.targetFound = False

def getDistance(self):
        #find min distance and then do trig
        distance12 = math.sqrt((self.positions['1'][0]-
self.positions['2'][0])**2+(self.positions['1'][1]-self.positions['2'][1])**2)
        distance23 = math.sqrt((self.positions['2'][0]-
self.positions['3'][0])**2+(self.positions['2'][1]-self.positions['3'][1])**2)
        distance13 = math.sqrt((self.positions['1'][0]-
self.positions['3'][0])**2+(self.positions['1'][1]-self.positions['3'][1])**2)
        distance = min(distance12, distance23, distance13)
        theta = distance*degPerPixel
        phi = 90-theta/2.0
        d = (ledDistance/2.0)*math.tan(phi*math.pi/180.0)
        print d
```

Figure 19 – IR Helper Library

The below figure is a picture of the code running. You can see the target as the three blue rectangles. If it could not find all 3 LEDs or if it found more than 3 LEDs it would show all points as red to indicate it wasn't a valid target. Then on the left you can see printouts of the distance of the camera to the target in inches.

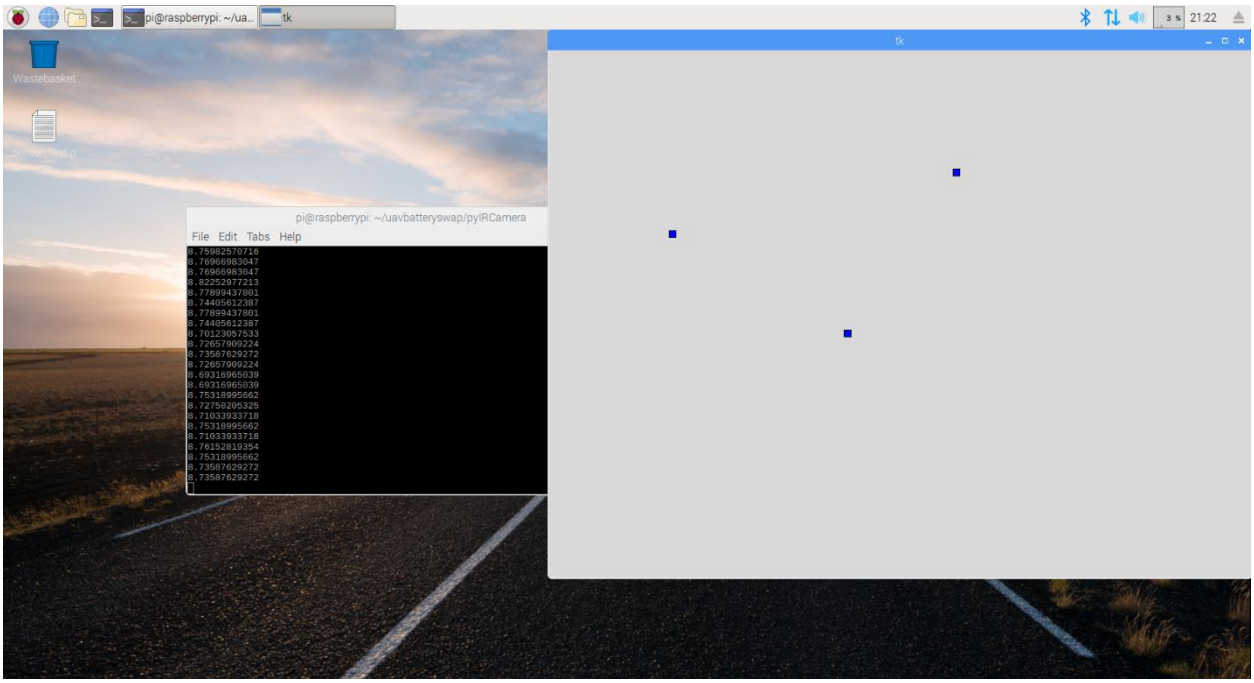


Figure 20 – IR Proof of Concept Screenshot [RJ]

5.2.5 Battery UAV State Machine

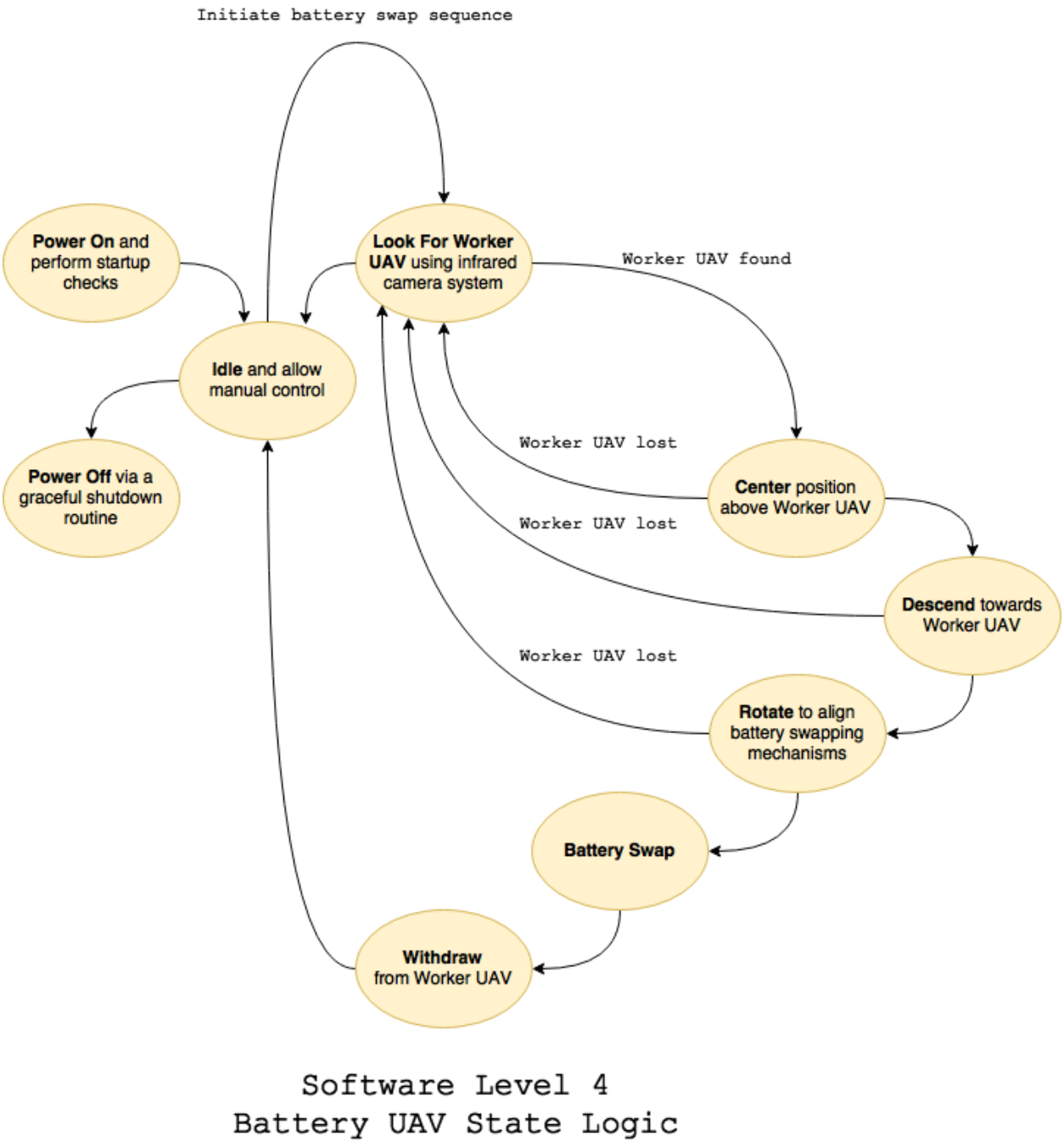


Figure 21 – Battery UAV State Machine [ND]

Worker UAV Pseudo Code
stateName (stateName on state diagram)

```
PowerOnState
// Connect to PixHawk
// Send arm command to UAV
// Wait for UAV to arm
// Build command for takeoff
// Send takeoff command to UAV
// Wait for UAV to reach predetermined safe height.
// Change state to IdleState

PowerOffState
// if (!LANDED)
//      Send LAND command
// Close all connections
// shutdown

IdleState
// if (allowManualControl != true)
//      Set flight mode to allow manual control
// if (newBatteryReceived)
//      Change state to SwitchToNewBatteryState

SwitchToNewBatteryState
// Change battery connection to new battery
// Change state to LockNewBatteryState

LockNewBatteryState
// Engage locks to lock new battery in place
// Change state to ReleaseOldBatteryState

ReleaseOldBatteryState
// Release the old battery by engaging electromagnets to repel the battery enclosure
// Change state to IdleState
```

Figure 22 – Worker UAV Pseudo Code [ND]

5.2.6 Worker UAV State Machine

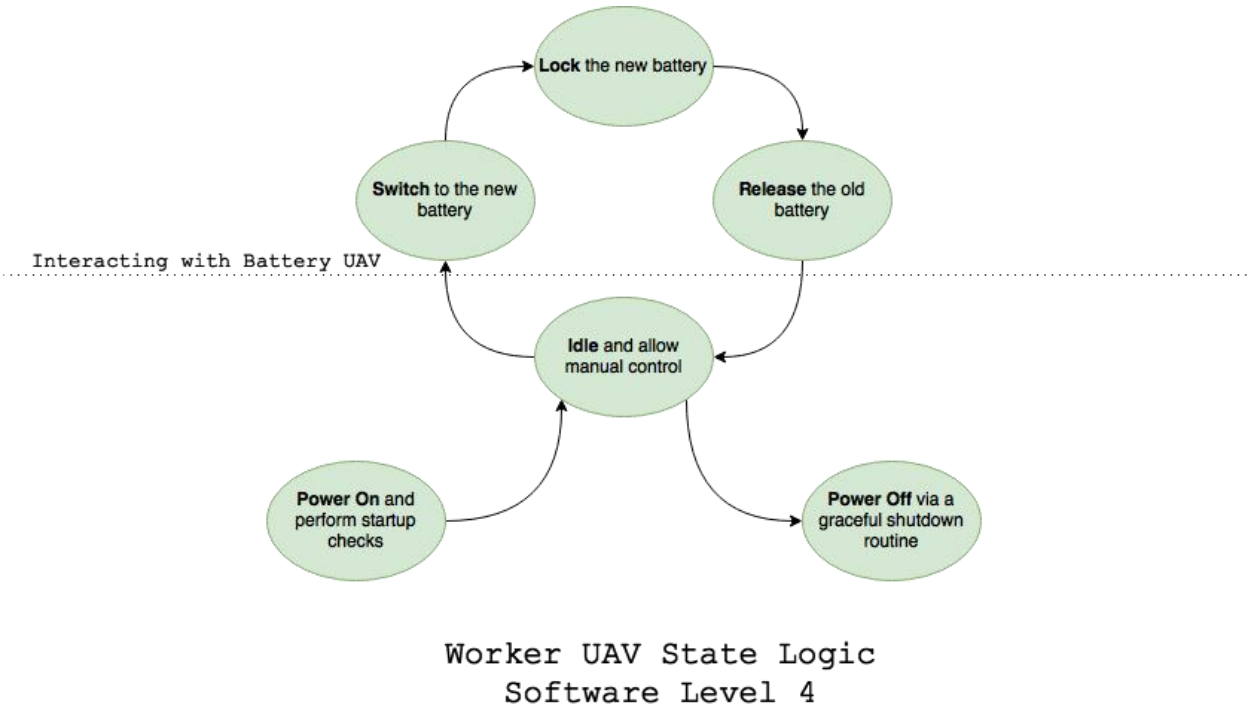


Figure 23 – Worker UAV State Machine [ND]

Battery UAV Pseudo Code
stateName (stateName on state diagram)

PowerOnState
// Connect to PixHawk
// Send arm command to UAV
// Wait for UAV to arm
// Build command for takeoff
// Send takeoff command to UAV
// Wait for UAV to reach predetermined safe height.
// Change state to IdleState

PowerOffState
// if (!LANDED)
// Send LAND command
// Close all connections
// shutdown

IdleState
// if (allowManualControl != true)
// Set flight mode to allow manual control
// if (LookForWorkerUAVSignal)
// Change state to LookForWorkerUAVState

```
LookForWorkerUAVState
// if (Worker UAV is in vision)
//     Connect to WorkerUAV over P2P wifi
//     Change state to CenterState

AlignXYState (CenterState)
// while X and Y position is not aligned and Worker UAV is in vision
//     Read infrared camera to obtain Worker UAV position information
//     Align X and Y using infrared camera information
// if X and Y position is aligned and Worker UAV is in vision
//     Change state to AlignZState
// else
//     Change state to LookForWorkerUAVState

AlignZState (DescendState)
// while Z position requirement is not met and Worker UAV is in vision
//     Read infrared camera to obtain Worker UAV position information
//     Descend in the Z direction to battery swap position
// if Z position is in battery swap position and Worker UAV is in vision
//     Change state to AlignThetaState
// else
//     Change state to LookForWorkerUAVState

AlignThetaState (RotateState)
// while Theta position is not aligned and Worker UAV is in vision
//     Read infrared camera to obtain Worker UAV position information
//     Align theta axis (yaw) using infrared camera information
// if Theta position is aligned and Worker UAV is in vision
//     Change state to BatterySwapState
// else
//     Change state to LookForWorkerUAVState

BatterySwapState
// Communicate to WorkerUAV that the battery swap sequence is being initiated
// Lower battery swapping mechanism into position
// Perform battery swap
// Change state to WithdrawState
```

Figure 24 – Worker UAV Pseudo Code [ND]

5.3 Mechanical Procedures

5.3.1 Structural Calculations

Since a full 3D model was already created to allow for the fabrication of the frame from raw material, it was also used for Solidworks FEA simulations to verify the design would be structurally sound. This model would also allow for the verification that the weight of the copter was within the ability of the motors to lift. All items in the model either have material properties applied to them to give them their physical attributes (like the aluminum body), or have their weight applied to the part if they are complicated and non-load bearing (micro-controllers or batteries). This gives the frame an approximate weight of 1.2kg, well within the lifting ability of the motors.

Below are the graphical outputs of FEA analysis for Stress, Strain, and Displacement. All of these simulations were done at stead state with the legs of the copter supporting the full weight. The simulation for the motors supporting the weight (for example hovering in mid-air) is almost identical since the legs sit under the motors.

The first graph for Stress shows how much force the materials are under. In this case, red shows materials under or near their maximum allowable stress before failure (bad). Dark blues are materials under almost no load.

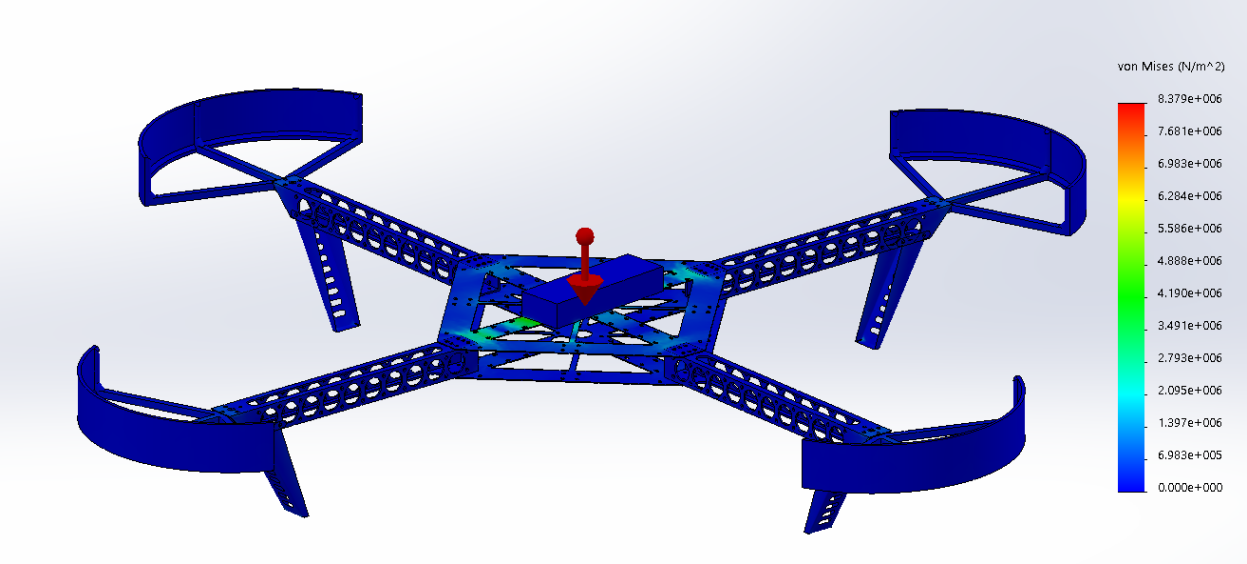


Figure 25

The next graph displays Strain, the amount the material is going to stretch per the force applied to it. Red shows the maximum value in the graph while blue shows the minimum (not necessarily good or bad). The maximum value is at the base of the prop guards. Since these don't experience any significant load this is acceptable.

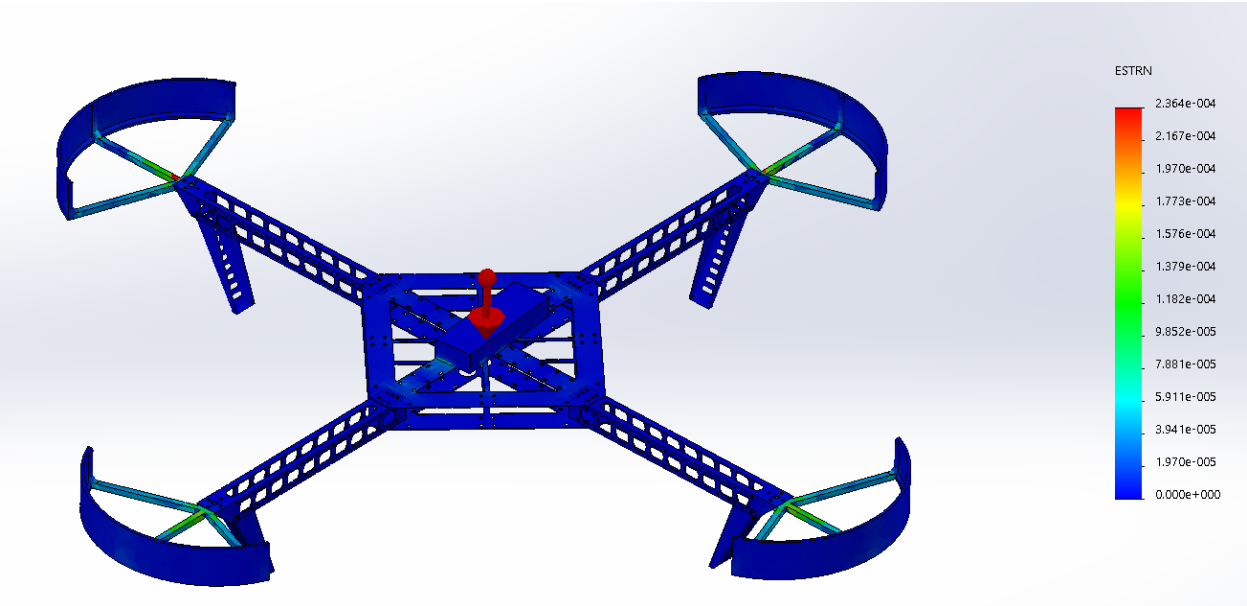


Figure 26

The final graph is for the Displacement of the quad. In this graph red shows the maximum deflection while blue shows the minimum (again, not necessarily good or bad). Under steady state forces (for example landed or hovering) the frame of the quad displaces a maximum of half a millimeter from where it should be. Additionally, the worst parts are the prop guards which are non-critical for flight stability. Load bearing parts of the frame deflect by less than 0.2mm.

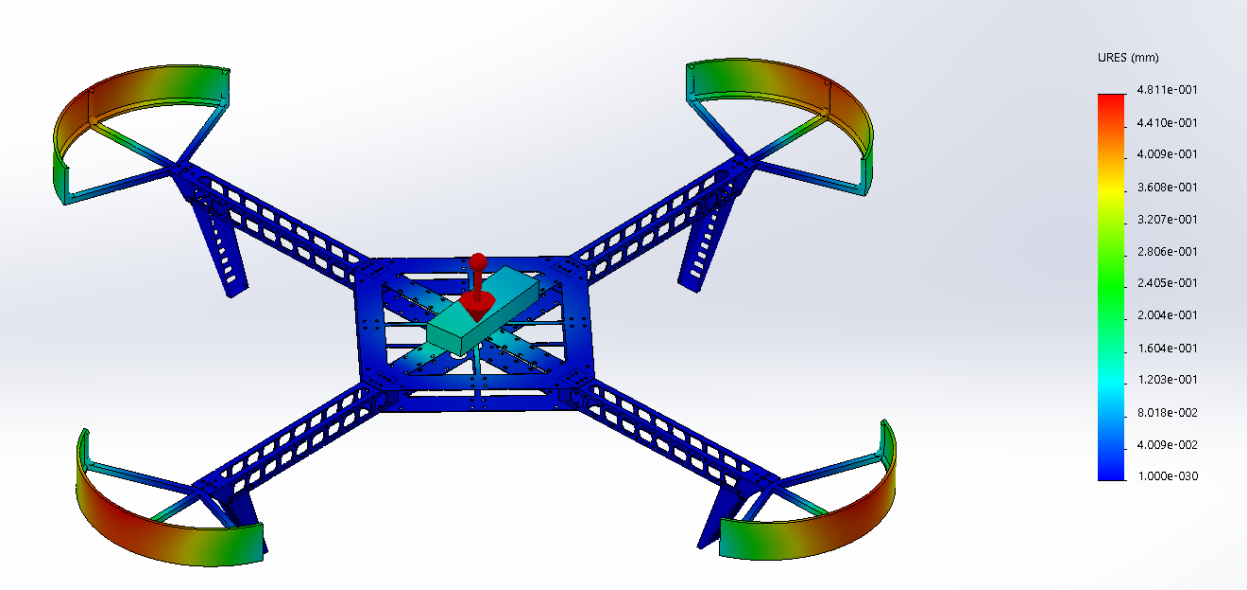


Figure 27

5.3.2 Mechanical Procedures of Battery Swapping

There are two critical systems that define this project, the machine vision that locates and aligns the battery UAV over the worker UAV and the mechanism that swaps the charged and depleted battery. These tasks have been broken into three different components; the scissor lift, battery box, and magnetic swap mechanisms. We will characterize and define the function and

requirements for each component and link them to the electrical and software domain of our project.

The figure below shows the lower half of the battery case that will be used to house the battery and act as the connector between the battery and the UAV's power rails.

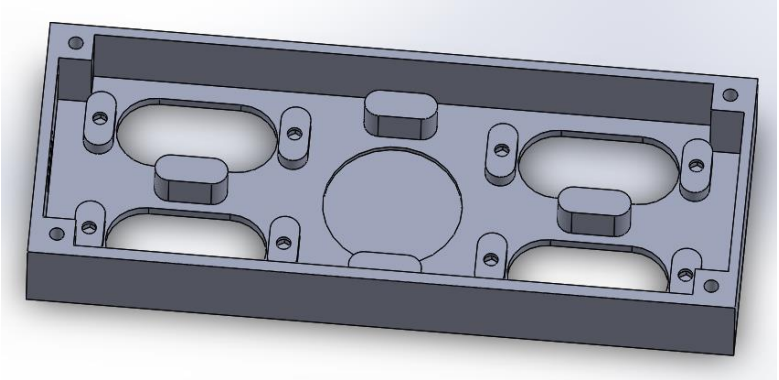


Figure 28 – Battery Case [WG]

The battery case has a vital role in the swapping process. It is designed so that the connecting and disconnecting of the batteries will be safe, quick and efficient. The battery case will act as a housing for the battery and fuse that will be protecting the battery and power system. Inside the battery case, the battery will be connected to conductive plates that will be mounted over the open slots cut into the bottom of the case. These plates will act as the contact points between the battery and the power rail on the UAV.

Magnets will also be stored on the top and bottom of the battery case. These magnets will be used as a means of dropping off a charged battery and picking up a depleted battery. Iron plates wrapped in wire will be used at the receiving and transmitting ends of the mechanical system. So in theory, when the battery is being replaced the permanent magnet on top of the battery case will be magnetically coupled to the iron plate on the transmitting end of the battery swap. Then, when the UAVs are aligned and a charged battery is ready to be received, the coil

wrapped around the iron plate on the transmitting end will be energized which will release the battery case from the transmitting end. Then, the permanent magnet on the bottom of the battery case will become magnetically coupled to the iron plate on the receiving end which will hold the battery case in place. This method of transporting the batteries allows the movement of the batteries to be bi-directional meaning they may be picked up or dropped off.

5.3.3 Scissor Lift Transfer

Of the three components, the most mechanically complex component of the swapping mechanism is the scissor lift. The scissor lift has two components with in its design. The first is the mounting system that attaches the scissor lift to the battery UAV. To create a stable system but allow for the two drones to move relative to each other, we will use a ball joint with springs to locate the scissor lift to a normal position in the XY-plane. Second, the key to the lift motion of the battery swap is vertical stability and control. It has been determined that the traditional scissor lift does just that. It is planned to 3D print the arms of the lift and use metal pins as the interfacing piece. A threaded rod assembly driven by a continuous servo will be used to control the scissor. Motors would be cheaper and often supply greater power, but can have major current spikes at startup and have no internal position feedback. The magnetic swap mechanism will be attached to the end of the scissor lift furthest from the battery drone.

When the scissor lift mechanism was designed, consistent stability and predicable control were kept in mind. By including springs on the ball joint assembly and a continuous servo in the lift assembly, both stability and control were achieved. Both solutions also give variable design parameters to customize each solution to produce the optimal result.

5.4 Essential Parts List

Qty.	Refdes	Part Num.	Description
2	B071YD56FM	3DR Pixhawk Mini Autopilot	flight control microcontroller
2	B00TROIMXW	T-MOTOR Air Gear 350 Motor, Prop, and ESC Combo Pack for Quadcopter	motors, ESCs, and props for worker UAV
3	B01MT4EA4D	Raspberry Pi 3 Barebones Kit - Includes Raspberry PI 3 - SanDisk 16GB Micro SD Card - Clear Case - Heatsink	Raspberry Pi 3s: 2 of them are flight controllers, the third is the simulator rig LED controllers
1	712096460711 …	PX4FLOW V1.3.1 Optical Flow Sensor Smart Camera for PX4 PIXHAWK Flight Control	Optical Flow sensor to determine x,y coordinates without gps
4	B004AFHP0Y	3DR Iris+ 3S 5100mAh 11.1V RC LiPo Drone Quadcopter Battery w/XT60 Plug by Venom	This is our battery. We need 4 of them so we have enough for both drones, recharging, and a swap to occur
1	B01I1ZB2SE	6 Piece N52 1.26" x 1/8" Most Powerful Disc Neodymium Magnets 50% Stronger Than N35 Magnets	Permanent Magnets needed for battery transfer

Table 46 – Essential Components Parts List [NR]

5.5 Essential Parts Budget

Qty.	Part Num	Description	Unit Cost	Total Cost
2	3DR Pixhawk Mini Autopilot	flight control microcontroller	$139.99	$279.98
2	T-MOTOR Air Gear 350 Motor, Prop, and ESC Combo Pack for Quadcopter	motors, ESCs, and props for worker UAV	99.95	199.90
3	Raspberry Pi 3 Barebones Kit - Includes Raspberry PI 3 - SanDisk 16GB Micro SD Card - Clear Case – Heatsink	Raspberry Pi 3s: 2 of them are flight controllers, the third is the simulator rig LED controllers	49.99	149.97
1	PX4FLOW V1.3.1 Optical Flow Sensor Smart Camera for PX4 PIXHAWK Flight Control	Optical Flow sensor to determine x,y coordinates without gps	56.99	56.99
4	3DR Iris+ 3S 5100mAh 11.1V RC LiPo Drone Quadcopter Battery w/XT60 Plug by Venom	This is our battery. We need 4 of them so we have enough for both drones, recharging, and a swap to occur	44.73	178.92
1	6 Piece N52 1.26" x 1/8" Most Powerful Disc Neodymium Magnets 50% Stronger Than N35 Magnets	Permanent Magnets needed for battery transfer	10.49	10.49

Table 47 – Essential Parts Budget Table [NR]

6.0 Network Infrastructure and Testing

6.1 Peer-to-Peer Communication

Even though the Raspberry Pi 3 Model B has WiFi, this device can fail to connect with large, public networks as is the case with the University of Akron's RooSecure network. In order to establish communication between the UAVs' Raspberry Pis and ground computers, an ad hoc network was created to enable peer to peer communication between laptops and the Raspberry Pis over WiFi bands. The following code was added to the /etc/network directory of the Raspberry Pis. This script assigned the Raspberry Pi an IP address and created a new network that broadcasted an SSID that was discoverable by WiFi enabled devices.

```
 interfaces(5) file used by ifup(8) and ifdown(8)


# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet dhcp

auto wlan0
iface wlan0 inet static
    address 192.168.54.1
    netmask 255.255.255.0
    wireless-channel 1
    wireless-essid PiAdHocNetwork
    wireless-mode ad-hoc
```

Figure 29 – Ad Hoc Code [NR]

6.2 IP Discovery

Because the Raspberry Pi is only a single board computer, it is difficult to determine the IP address of the Pi without plugging it into a monitor and checking it visually. In order to

increase testing efficiency when testing in the field or the lab, the following script was implemented on boot up to send a text message containing the IP address of the Raspberry Pi and the network name it is connected to.

```
import subprocess
import smtplib
import socket
from email.mime.text import MIMEText
import datetime
import time

time.sleep(10)
# Change to your own account information
to = '4402891995@vtext.com'
gmail_user = 'nmr33@zips.uakron.edu'
gmail_password = '********'
smtpserver = smtplib.SMTP('smtp.gmail.com', 587)
smtpserver.ehlo()
smtpserver.starttls()
smtpserver.ehlo
smtpserver.login(gmail_user, gmail_password)
today = datetime.date.today()
# Very Linux Specific
arg='ip route list'
p=subprocess.Popen(arg,shell=True,stdout=subprocess.PIPE)
data = p.communicate()
split_data = data[0].split()
ipaddr = split_data[split_data.index('src')+1]
my_ip = 'The adhoc IP is 169.254.137.201, Your Ip is %s' %  ipaddr
msg = MIMEText(my_ip)
msg['Subject'] = 'IP For RaspberryPi on %s' % today.strftime('%b %d %Y')
msg['From'] = gmail_user
msg['To'] = to
smtpserver.sendmail(gmail_user, [to], msg.as_string())
smtpserver.quit()
```

Figure 30 – IP Text Message Script [NR]

7.0 Gantt Chart

7.1 Fall, 2017

The Gantt chart below is a layout for Autonomous UAV Battery Swapping project for the Autumn of 2017. Each task was given a start and completion time with leveled breakdowns of tasks to create more reasonable jobs.

Task Finish: Tue 12/5/17
Duration: 102 days
% Complete: 0%

SDP1 fall2017			
Project Design			
Preliminary Design Report			
Problem Statement	1 day	Tue 8/29/17	Tue 8/29/17
Preliminary Design Gantt Chart	14 days	Tue 8/29/17	Mon 9/11/17
Block Diagrams Level 0, 1, ... w/ FR tables	**14 days**	Tue 9/5/17	Mon 9/18/17
Hardware modules (Creighton, Walker)	**14 days**	**Tue 9/5/17**	**Mon 9/18/17**
Level 0 (Creighton)	1 day	Tue 9/5/17	Tue 9/5/17
Power (Walker)	10 days	Tue 9/5/17	Thu 9/14/17
Control (Creighton)	6 days	Wed 9/6/17	Mon 9/11/17
Motors (Creighton)	3 days	Tue 9/12/17	Thu 9/14/17
Battery Swap (Creighton, Walker)	4 days	Thu 9/14/17	Sun 9/17/17
Software modules (Nathan, Niko, Reed)	**14 days**	**Tue 9/5/17**	**Mon 9/18/17**
Flight Control (Nathan)	10 days	Tue 9/5/17	Thu 9/14/17
Vision (Niko, Reed)	10 days	Tue 9/5/17	Thu 9/14/17
Battery Swap Control (Nathan, Reed)	4 days	Fri 9/15/17	Mon 9/18/17
Preliminary Design Presentation 9:55am	1 day	Tue 9/19/17	Tue 9/19/17
Midterm Report	**28 days**	**Tue 9/19/17**	**Mon 10/16/17**
Design Requirements Specification (Niko)	7 days	Tue 9/19/17	Mon 9/25/17
Midterm Design Gantt Chart (Reed)	28 days	Tue 9/19/17	Mon 10/16/17
Design Calculations	**7 days**	**Tue 9/19/17**	**Mon 9/25/17**
Electrical Calculations	**4 days**	**Tue 9/19/17**	**Fri 9/22/17**
Communication (Niko)	4 days	Tue 9/19/17	Fri 9/22/17
Computing (Nathan)	4 days	Tue 9/19/17	Fri 9/22/17
Control Systems	1 day	Tue 9/19/17	Tue 9/19/17
Power, Voltage, Current (Walker)	4 days	Tue 9/19/17	Fri 9/22/17
Mechanical Calculations	**7 days**	**Tue 9/19/17**	**Mon 9/25/17**
Structual Considerations (Reed)	7 days	Tue 9/19/17	Mon 9/25/17
System Dynamics (Reed)	7 days	Tue 9/19/17	Mon 9/25/17
Block Diagrams Level 2 w/ FR tables & ToO	**12 days**	**Sat 9/23/17**	**Wed 10/4/17**
Hardware modules (Creighton, Walker)	**12 days**	**Sat 9/23/17**	**Wed 10/4/17**
Battery (Walker)	1 day	Sat 9/23/17	Sat 9/23/17
Power Regulator (Walker)	9 days	Sun 9/24/17	Mon 10/2/17

Hardware modules (Creighton, Walker)	**12 days**	**Sat 9/23/17**	**Wed 10/4/17**
Battery (Walker)	1 day	Sat 9/23/17	Sat 9/23/17
Power Regulator (Walker)	9 days	Sun 9/24/17	Mon 10/2/17
Flight Controller and Sensors (Creighton)	3 days	Tue 9/19/17	Thu 9/21/17
Single Board Computer and Sensors (Creighton)	4 days	Fri 9/22/17	Mon 9/25/17
ESC & Motors (Creighton)	3 days	Tue 9/26/17	Thu 9/28/17
Battery Swap (Creighton, Walker)	2 days	Tue 10/3/17	Wed 10/4/17
Software modules (Nathan, Reed)	**12 days**	**Sat 9/23/17**	**Wed 10/4/17**
Get Vision Sensor Information	12 days	Sat 9/23/17	Wed 10/4/17
Battery Drone State Handler	1 day	Sat 9/23/17	Sat 9/23/17
MavLink Message Factory	10 days	Sun 9/24/17	Tue 10/3/17
Worker Drone State Handler	1 day	Wed 10/4/17	Wed 10/4/17
Block Diagrams Level 3 w/ FR tables & ToO	**12 days**	**Thu 10/5/17**	**Mon 10/16/17**
Hardware modules (Creighton, Walker)	**12 days**	**Thu 10/5/17**	**Mon 10/16/17**
Battery (Walker)	1 day	Thu 10/5/17	Thu 10/5/17
Power Regulator (Walker)	9 days	Fri 10/6/17	Sat 10/14/17
Flight Controller and Sensors (Creighton)	1 day	Thu 10/5/17	Thu 10/5/17
Single Board Computer (Creighton)	1 day	Fri 10/6/17	Fri 10/6/17
Optical Sensor (Creighton)	6 days	Sat 10/7/17	Thu 10/12/17
ESC & Motors (Creighton)	1 day	Fri 10/13/17	Fri 10/13/17
Tether Extension/Retraction (Creighton)	3 days	Sat 10/14/17	Mon 10/16/17
Battery Retention (Walker)	2 days	Sun 10/15/17	Mon 10/16/17
Software modules (Nathan, Reed)	**12 days**	**Thu 10/5/17**	**Mon 10/16/17**
Comptuer Vision Module (Reed)	12 days	Thu 10/5/17	Mon 10/16/17
Battery State Controller (Nathan)	3 days	Thu 10/5/17	Sat 10/7/17
Communication Module (Nathan)	3 days	Sun 10/8/17	Tue 10/10/17
Flight Module (Nathan)	3 days	Wed 10/11/1	Fri 10/13/17
Worker State Controller (Nathan)	3 days	Sat 10/14/17	Mon 10/16/17
Midterm Design Presentations 9:55-11:35am Part 1	1 day	Tue 10/17/17	Tue 10/17/17
Midterm Design Presentations 9:55-11:35am Part 2	1 day	Tue 10/24/17	Tue 10/24/17
Project Poster	14 days	Tue 10/31/17	Mon 11/13/17
Final Design Report	**42 days**	**Tue 10/17/17**	**Mon 11/27/17**
Abstract (Niko)	10 days	Tue 10/17/17	Thu 10/26/17

Worker State Controller (Nathan)	3 days	Sat 10/14/17	Mon 10/16/17
Midterm Design Presentations 9:55-11:35am Part 1	1 day	Tue 10/17/17	Tue 10/17/17
Midterm Design Presentations 9:55-11:35am Part 2	1 day	Tue 10/24/17	Tue 10/24/17
Project Poster	14 days	Tue 10/31/17	Mon 11/13/17
Final Design Report	**42 days**	**Tue 10/17/17**	**Mon 11/27/17**
Abstract (Niko)	10 days	Tue 10/17/17	Thu 10/26/17
Software Design	**42 days**	**Tue 10/17/17**	**Mon 11/27/17**
Battery Drone (Nathan, Reed, Niko)	**32 days**	**Tue 10/17/17**	**Fri 11/17/17**
FR Tables (Nathan, Reed)	4 days	Tue 10/17/17	Fri 10/20/17
Flow Chart (Nathan)	4 days	Sat 10/21/17	Tue 10/24/17
Psuedo Code (Nathan)	24 days	Wed 10/25/17	Fri 11/17/17
IR Camera Code (Reed)	28 days	Sat 10/21/17	Fri 11/17/17
Worker Drone (Nathan, Reed,Niko)	**10 days**	**Sat 11/18/17**	**Mon 11/27/17**
Flow Chart (Nathan)	5 days	Sat 11/18/17	Wed 11/22/17
Psuedo Code (Nathan)	5 days	Thu 11/23/17	Mon 11/27/17
Hardware Design	**42 days**	**Tue 10/17/17**	**Mon 11/27/17**
Battery Drone (Creighton, Walker)	**24 days**	**Tue 10/17/17**	**Thu 11/9/17**
Schematics (Creighton, Walker)	24 days	Tue 10/17/17	Thu 11/9/17
Worker Drone (Creighton, Walker)	**18 days**	**Thu 11/9/17**	**Sun 11/26/17**
Schematics (Creighton, Walker)	18 days	Thu 11/9/17	Sun 11/26/17
Parts Request Form (Niko)	10 days	Fri 10/27/17	Sun 11/5/17
Budget (Estimated) (Niko)	10 days	Mon 11/6/17	Wed 11/15/17
Implementation Gantt Chart (Reed)	5 days	Thu 11/23/17	Mon 11/27/17
Conclusions and Recommendations (Niko)	12 days	Thu 11/16/17	Mon 11/27/17
Final Design Presentations 9:55-11:35am Part 1	1 day	Tue 11/28/17	Tue 11/28/17
Final Design Presentations 9:55-11:35am Part 2	1 day	Tue 12/5/17	Tue 12/5/17

Figure 31 – Fall Semester 2017 Gantt Chart [RJ]

7.2 Spring, 2018

	Task Mode	Task Name	Duration	Start	Finish	
1		**SDPII Implementation 2017**	**105 days**	**Tue 1/16/18**	**Mon 4/30/18**	
2		Revise Gantt Chart	14 days	Tue 1/16/18	Mon 1/29/18	
3		**Implement Project Design**	**97 days**	**Tue 1/16/18**	**Sun 4/22/18**	
4		**Hardware Implementation**	**56 days**	**Tue 1/16/18**	**Mon 3/12/18**	
5		Breadboard Components	13 days	Tue 1/16/18	Sun 1/28/18	
6		Layout and Generate PCB(s)	14 days	Mon 1/29/18	Sun 2/11/18	5
7		Assemble Hardware	7 days	Mon 2/12/18	Sun 2/18/18	6
8		Test Hardware	14 days	Mon 2/19/18	Sun 3/4/18	7
9		Revise Hardware	14 days	Mon 2/19/18	Sun 3/4/18	7
10		*MIDTERM: Demonstrate Hardware*	5 days	Mon 3/5/18	Fri 3/9/18	8
11		SDC & FA Hardware Approval	0 days	Mon 3/12/18	Mon 3/12/18	
12		**Software Implementation**	**56 days**	**Tue 1/16/18**	**Mon 3/12/18**	
13		Develop Software	27 days	Tue 1/16/18	Sun 2/11/18	
14		Test Software	21 days	Mon 2/12/18	Sun 3/4/18	13
15		Revise Software	21 days	Mon 2/12/18	Sun 3/4/18	13
16		*MIDTERM: Demonstrate Software*	5 days	Mon 3/5/18	Fri 3/9/18	15
17		SDC & FA Software Approval	0 days	Mon 3/12/18	Mon 3/12/18	
18		**System Integration**	**42 days**	**Mon 3/12/18**	**Sun 4/22/18**	
19		Assemble Complete System	14 days	Mon 3/12/18	Sun 3/25/18	
20		Test Complete System	21 days	Mon 3/26/18	Sun 4/15/18	
21		Revise Complete System	21 days	Mon 3/26/18	Sun 4/15/18	
22		*Demonstration of Complete System*	7 days	Mon 4/16/18	Sun 4/22/18	21
23		**Develop Final Report**	**98 days**	**Tue 1/16/18**	**Mon 4/23/18**	
24		Write Final Report	98 days	Tue 1/16/18	Mon 4/23/18	
25		Submit Final Report	0 days	Mon 4/23/18	Mon 4/23/18	24
26		***Project Demonstration and Presentation***	0 days	Mon 4/23/18	Mon 4/23/18	

Figure 32 – Spring, 2018 Gantt Chart [RJ]

8.0 Works Cited

[1] Bodin, William Kress, Redman, Jesse, Thorson, Derral Charles. Navigating UAVS with an On-Board Digital Camera. US: Patent 20100004802A1. 01-07-2010.

[2] Gentry, Nicholas Kristofer. On-Board Redundant Power System for Unmanned Aerial Vehicles. US: Patent 9376208B1. 06-28-2016.

[3] Kantor, Igor, Srivastava, Ashok N., Pasko, Douglas M., BATLA, Hani, Ubhi, Gurpreet. Unmanned Aerial Vehicle Network-Based Recharging. US: Patent 20150336669A1. 11-26-2015.

[4] Lutke, Kevin Reed, & Kutzmann, Aaron Jonathan. Unmanned Aerial Vehicle Base Station. US: Patent 8511606B1. 08-20-2013.

[5] Wang, Mingxi. Systems and Methods for UAV Battery Exchange. US: Patent 9440545B2. 09-13-2016.

[6] K. Fujii, K. Higuchi and J. Rekimoto, "Endless Flyer: A Continuous Flying Drone with Automatic Battery Replacement," 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, Vietri sul Mere, 2013, pp. 216-223.

[7] J. Kim, J. Lee, J. Jeong, H. Kim, J. S. Park and T. Kim, "SAN: Self-Adaptive Navigation for Drone Battery Charging in Wireless Drone Networks," 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Crans-Montana, 2016, pp. 248-251.

[8] F. Schenkelberg, "How reliable does a delivery drone have to be?," 2016 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, 2016, pp. 1-5.

[9] D. Schneider, "Air traffic control for delivery drones [Top Tech 2017]," in IEEE Spectrum, vol. 54, no. 1, pp. 32-33, January 2017.

[10] T. Ishikawa, T. Tsuji, S. Hashimoto and N. Kurita, "A simple equivalent circuit for efficiency calculation of brushless DC motors," *2013 International Conference on Electrical Machines and Systems (ICEMS)*, Busan, 2013, pp. 1133-1138.

[11] C. Vincenzo Angelino, V. R. Baraniello and L. Cicala, "High altitude UAV navigation using IMU, GPS and camera," *Proceedings of the 16th International Conference on Information Fusion*, Istanbul, 2013, pp. 647-654.

[12] F. Morbidi, R. Cano and D. Lara, "Minimum-energy path generation for a quadrotor UAV," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, 2016, pp. 1492-1498.

[13] M. Ong *et al*., "Intelligent agent based power management for UAV systems," *6th IET International Conference on Power Electronics, Machines and Drives (PEMD 2012)*, Bristol, 2012, pp. 1-6.

[14] P. Kakvand, M. Jaberzadeh, M. M. Inallou and Y. Alborz, "Smart on-board UAV system: Using computer vision system to find a movable and stationary target," *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, Tehran, 2015, pp. 694-699.

[15] Toksoz, Tuna, Joshua Redding, Matthew Michini, Bernard Michini, and Jonathan P. How. *Automated Battery Swap and Recharge to Enable Persistant UAV Missions*. Tech. St.Louis, Missouri: Massachusetts Institute of Technology, 2011. *ARC*. Web. 28 Feb. 2017.

[16] Mansor, Maszatul M., Ioannis Giagkiozis, Derek Wall, Andrew R. Mills, Robin C. Purshouse, and Peter J. Fleming. *Real-Time Improved Power Management for Autonomous Systems*. Rep. 3rd ed. Vol. 47. Cape Town, South Africa: International Federation of Automatic Control, 2014. Pages 2634-2639. Web. 28 Feb. 2017.

[17] Park, Sangyoung, Licong Zhang, and Samarjit Chakraborty. *Design Space Exploration of*

Drone Infrastructure for Large-Scale Delivery Services. Publication. New York, NY: ACM,

2016. *ACM*. Web. 28 Feb. 2017.

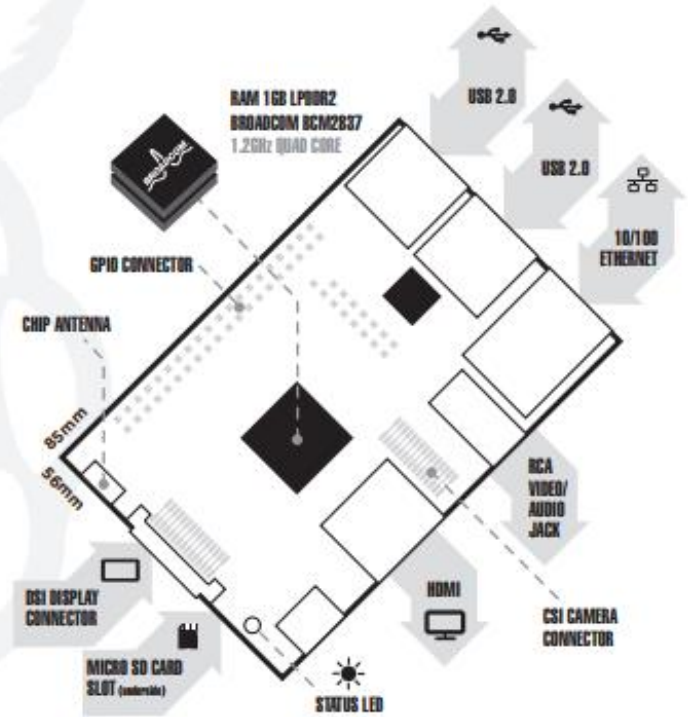
9.0 Appendix:

9.1 Raspberry Pi Data Sheet:

Raspberry Pi

Raspberry Pi 3 Model B

Product Name	**Raspberry Pi 3**
Product Description	The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processor, 10x faster than the first generation Raspberry Pi. Additionally it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs.
RS Part Number	**896-8660**



RS

www.rs-components.com/raspberrypi

Raspberry Pi

Raspberry Pi 3 Model B

Specifications

Processor	Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
GPU	Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V1, 2.5A

Connectors:

Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4 Composite RCA (PAL and NTSC)
Audio Output	Audio Output 3.5mm jack, HDMI USB 4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Push/pull Micro SDIO

Key Benefits

- Low cost
- Consistent board format
- 10x faster processing
- Added connectivity

Key Applications

- Low cost PC/tablet/laptop
- IoT applications
- Media centre
- Robotics
- Industrial/Home automation
- Server/cloud server
- Print server
- Security monitoring
- Web camera
- Gaming
- Wireless access point
- Environmental sensing/monitoring (e.g. weather station)

RS

9.2 Pixhawk Data Sheet:

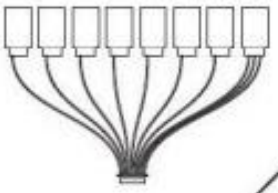
Pixhawk Mini

Advanced Autonomous
Vehicle Control

QUICK START GUIDE

3DR

PARTS

Pixhawk Mini Autopilot	
GPS module	
Quad Power Distribution Board	
8 channel PWM breakout board	
4 pin cable	
RC-in cable	
6 to 6 pin and 4 pin "Y"cable	
6 pin cable (2)	
6 pin JST to DF13	
Safety switch	
8 channel PWM breakout cable	

Pixhawk Mini Autopilot

GPS module

Quad Power Distribution Board

(A) 8 channel PWM breakout board

(B) 4 pin cable

(C) RC-in cable

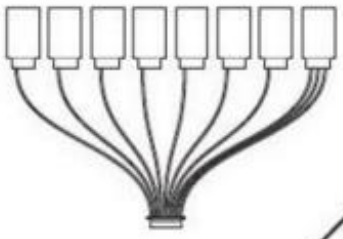
(D) 6 to 6 pin and 4 pin "Y"cable

(E) 6 pin cable (2)

(F) 6 pin JST to DF13

(G) Safety switch

(H) 8 channel PWM breakout cable



GETTING STARTED

With the help of PX4 firmware, Pixhawk mini turns any RC plane, copter, or rover into a full-featured personal drone. Once you have a fully assembled vehicle, follow this guide to install Pixhawk mini.

MOUNT

Use the provided foam pads to mount Pixhawk mini as close as possible to your vehicle's center of gravity. Make sure to orient the board with the arrow pointing forward.

VEHICLE FRONT

CONNECT RADIO CONTROL

For PPM Receivers

For Spektrum DSM Receivers

CONNECT MOTOR OUTPUT

SPECIFICATIONS

Main Processor: STM32F427 Rev 3
IO Processor: STM32F103
Sensors
Accel/Gyro/Mag: MPU9250
Accel/Gyro: ICM20608
Barometer: MS5611
Voltage Ratings:
Power module output: 4.1~5.5V
Max input voltage: 45V (10S LiPo)
Max current sensing: 90A
USB Power Input: 4.1~5.5V
Servo Rail Input: 0~10V
Dimensions: 38x43x12mm
Weight: 15.8g
GPS Module: GNSS receiver: ublox Neo-M8N; compass HMC5983
Weight: 22.4g
Dimensions: 37x37x12mm
Interface:
1 x UART Serial Port
Spektrum DSM/DSM2/DSM-X® Satellite Compatible
Futaba S BUS® Compatible
PPM Sum Signal Input
I2C
CAN
ADC
Internal Micro USB Port

OPTIONAL ACCESSORIES

Digital Airspeed sensor + Pitot tube (MS525DO)
Standard Telemetry (433MHz and 915MHz)
WiFi Telemetry (2.4GHz WiFi Radio)

PIN OUTS

Custom installations may require custom made cables.
Here's a handy description of all Pixhawk Mini's connectors and what they do.
Just in case...

TELEM PORT

1 (red)	VCC	+5V
2 (blk)	TX1 (OUT)	+3.3V
3 (blk)	RX1 (IN)	+3.3V
4 (blk)	GND	GND

CAN PORT

1 (red)	VCC	+5V
2 (blk)	CAN-H	+3.3V
3 (blk)	CAN-L	+3.3V
4 (blk)	GND	GND

SAFETY SWITCH PORT

1 (red)	VCC	+5V
2 (blk)	IO_LED_SAFETY	GND
3 (blk)	SAFETY	GND

GPS & I2C PORT

1 (red)	SCL	+3.3V
2 (blk)	SDA	+3.3V
3 (blk)	VCC	+5V
4 (blk)	TX3	+3.3V
5 (blk)	RX3	+3.3V
6 (blk)	GND	GND

POWER INPUT PORT

1 (red)	SCL	+3.3V
2 (blk)	SDA	+3.3V
3 (blk)	VCC	+5V
4 (blk)	TX3	+3.3V
5 (blk)	RX3	+3.3V
6 (blk)	GND	GND

CHANNEL PIN OUTS

PIN	Multirotos	4 Channel Planes	Rovers
Pin 1	Motor 1	Aileron	-
Pin 2	Motor 2	Elevator	-
Pin 3	Motor 3	Throttle	Throttle
Pin 4	Motor 4	Rudder	Steering
Pin 5	Motor 5	-	-
Pin 6	Motor 6	-	-
Pin 7	Motor 7	-	-
Pin 8	Motor 8	-	-

For plances with configurations other than 4 channels, see px4.io for more information.

ADDITIONAL INFORMATION

Be sure to visit http://px4.io/ for further information including tutorials, configurations, and community support.