**UAV Docking**

**TOI-1231b**

A picture containing calendar

Description automatically generated

**Team Members:**

Jayden Bautista, Javier Ventura Guerra, Talin Hambarsoumian, Brian Lee,

Jared Roe, Irving Lorenzano, Lance Pancha, Nathaniel Ruiz

Zhubeen Vakilzadeh, Anthony Woo, Christopher Young, Matthew Tootoonchi, Anthony Nursalim

**Faculty Advisor:** PI: Professor: Mohamed El-Hadedy

**Email:** mealy@cpp.edu

**California State Polytechnic University, Pomona**

**Department of Electrical and Computer Engineering, College of Engineering**

x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of the sponsoring faculty advisor verifying that he has read and reviewed the paper prior to submission to NASA.

# **Table of Contents**

[**I. INTRODUCTION**](#_g21ga6a8sr31)1

[**II. Project Management**](#_jbto73du2w27)2

[A. System Requirement Analysis](#_weposafudju1) 2

[B. Schedule of Work](#_wh6k0zwigaqx) 2

[C. Cost Budget](#_cfionfbuo1fw) 3

[**III. Systems Engineering**](#_vfg1xla5tdhw)4

[1. Concept of Operations](#_v2ka0v4jqjsp) 4

[A. Drone Build](#_t99fva9qz310) 4

[B. Landing Method](#_hw5m8c3b46q) 4

[C. Power Transfer/Charging Circuit](#_phsi5wcykca6) 6

[D. Data Transfer/Apache Flink/Kafka](#_w89jfirhdasb) 7

[2. System Hierarchy](#_44yfyjjb532b) 8

[3. Interfaces](#_4rxwtlf5kser) 8

[4.. Technical Performance and Verification](#_46x7q8t1bp4l) 8

[6. Trade Studies](#_w19ix2v9bkd0) 10

[7. Reliability and Safety](#_sywyturubjg3) 11

[**CONCLUSION**](#_b5peofuynv47)11

[**APPENDIX**](#_qh86epwiias)12

[**ACKNOWLEDGMENT**](#_l8tchosfrd9v)12

[**REFERENCES**](#_8hi36btu8taz)12

UAVs Docking

***Abstract*—** **Drone docking in mid-air is incredibly useful in multiple situations such as in refueling, package delivery, and cargo swapping. Most current systems capable of this functionality require experienced, trained pilots to navigate and aerially dock these entities together. Developing a system to do this automatically and efficiently decreases cost dramatically and takes much less time to complete tasks, creating a demand for this technology to be implemented. Drone mid-air docking has the fundamental challenge of securing a connection between the main and support entities. In this project, we introduce an implementation of a system where a support drone can dock onto another drone, charge its battery, then depart from the main drone and return using a second battery. Using wide angle cameras and ultrasound sensors, main and support drones are able to avoid collision and track each other mid-air. In addition, the docking mechanism is completely autonomous, meaning they should have the ability to fly and navigate itself. Furthermore, the current docking mechanism, which is introduced in the current project has the ability to allow the flying drones to work as an edge computing nodes that can work as a cluster using both Flink for streaming processing data and Kafka for data streaming between nodes, The current computing nodes are single computer (Raspberry PIs) boards. In addition, there are two different types of flight controllers lying on the top of each Raspberry Pi (One is NAVIO2 and the other is Pixhawk 2.4.8). The reason for using two different flight controllers is for testing and searching the different commercial controllers solutions and defining the best for drones’ docking in the mid-air scenario.**

***Index Terms*—**

**ESC – Electronic Speed Controllers,**

**GPS – Global Positioning System,**

**UAV – Unmanned Aerial Vehicle**

**LiPo – Lithium Polymer,**

# **I. INTRODUCTION**

In recent years, the applications and technological advancements of unmanned aerial vehicles (UAVs) have progressed tremendously. Specifically, quadcopters are multirotor drones that are powered by LiPo batteries.  However, the main limitation of these UAVs is the consumption of the onboard power source, which restricts drones from having extended flight times. Thus, human interference is required to replace the battery or charge it. Docking is a procedure for predicting the relative position of two moving objects to attach and detach them from each other. Historically, the US Air Force started their docking program for extending the range of the air jets while in operation. Moreover, NASA and the space private sector are using docking for attaching/detaching their payloads into the International Space Station (ISS). This implementation of a docking system between a charging and receiving drone will allow for an autonomous charging system and direct connection to transfer data within.

Systems engineering is invaluable to this project due to its innate ability to lay out complex designs and systems in an organized and understandable manner. In this project, there are many systems interacting with each other in elaborate ways, and in order to reach our goal, everything in this network must interact together without fault. Throughout the journey of this project, the process of systems engineering assisted in the design, integration, and management of each component, ensuring an organized and integrated rollout of the system as a whole.

Additionally in this paper, the IEEE (Institute of Electrical and Electronics Engineers) format was utilized to allow for another layer of organization and display for this project's separate sections.

This work was funded by NASA for participating in the NASA MINDS competition. Jayden Bautista (email: jlbautista@cpp.edu), Javier Ventura Guerra (email: javierguerra@cpp.edu), Talin Hambarsoumian (email: talinh@cpp.edu ), Brian Lee (email: bjlee@cpp.edu), Irving Lorenzano (email: ilorenzano@cpp.edu), Anthony Nursalim (email: [anursalim@cpp.edu](mailto:anursalim@cpp.edu)), Lance Pancha (email: lbpancha@cpp.edu), Jared Roe (email: jcroe@cpp.edu), Nathaniel Ruiz (email: njruiz@cpp.edu), Matthew Tootoonchi (email: mftootoonchi@cpp.edu), Zhubeen Vakilzadeh (email: zvakilzadeh@cpp.edu), Anthony Woo (email: anwoo@cpp.edu), and Christopher Young (email: cay@cpp.edu) are students under the Department of Electrical and Computer Engineering at the California State Polytechnic University, Pomona, CA, 91768 USA.

# **II. Project Management**

## **A. System Requirement Analysis**

The system requirement review for this project is intended to provide a detailed explanation of the features and design. It will serve as a tool to communicate the design plan between the senior project teams to fully understand the specification requirements and design plan moving forward. To successfully build a complex and meaningful system, requirements must be understood and laid out for development. the requirements for this project are separated into four sections:

* Drone build
* Landing method
* Power transfer
* Data transfer

The first part of the project is to design and build two drones that satisfy the following requirements:

* One drone must be a charger drone (Biggy) and another must be a receiver drone (Tiny).
* Tiny must have object and barcode detection capabilities.
* Biggy must have power transfer capabilities.
* Both Tiny and Biggy must have the capability to fly autonomously.

The second aspect of the project is to design and build a landing method that satisfies the following requirements:

* Biggy must have a landing platform for Tiny to land on.
* Tiny must be able to detect barcodes placed on Biggy for accurate landing.
* The distance between the drones must be calculated to avoid collision.
* Tiny must be able to land on the charging platform accurately in order to charge.

The third part of the project is designing and building a charging system that would satisfy the following requirements:

* Both drones must have corresponding charging circuits for successful power transfer.
* Each drone must have metal contact rails to be used in transferring power between them.
* Safety measures must be in place to prevent possible shorting of the battery.

The final part of this project is to design a communication system between the drones with using Apache Flink in order to satisfy the following requirement:

* Data must be transferable between the two drones as well as a central command station.

The following section outlines how this work was broken up and accomplished.

## **B. Schedule of Work**

Due to restrictions caused by the COVID 19 Pandemic on campus meetings were delayed until February 14th, 2022.

| **Drone Building Team** | | | |
| --- | --- | --- | --- |
| **Description** | | **Start** | **End** |
| Drone Concepts (Design, layout, Modeling, Flight Controls) | | Sept 15, 2021 | Feb 1, 2022 |
| Drone Parts for Order | | Sept 15, 2021 | Feb 10, 2022 |
| Initial Building of Drones | | Feb 14, 2022 | Feb 21,  2022 |
| Flight Testing and Drone Modification | | Feb 21,  2022 | On Going |

| **Charging Team** | | | |
| --- | --- | --- | --- |
| **Description** | | **Start** | **End** |
| Charging Concepts | | Sept 15, 2021 | Feb 1, 2022 |
| Charging Parts for Order | | Sept 15, 2021 | Feb 14, 2022 |
| Building of Charging circuit | | Feb 21, 2022 | Feb 28, 2022 |
| Charging Testing | | Feb 28,  2022 | Mar 7, 2022 |
| Implementation of Charger to Drones | | Mar 7, 2022 | Mar 14, 2022 |

| **Flink Team** | | | |
| --- | --- | --- | --- |
| **Description** | | **Start** | **End** |
| Studying of the Concept of Flink | | Sept 15, 2021 | Feb 1, 2022 |
| Testing the Software applications | | Sept 15, 2021 | On Going |
| Setting up Test Environment | | Feb 14, 2022 | On Going |
| Installing Flink onto the Drones | | Feb 21,  2022 | Mar 18, 2022 |

## **C. Cost Budget**

The fundings’ for this project is provided by NASA Minds. The majority of the funding is dedicated to the parts needed to build the drones. The first estimation and cost analysis is done based on the initial ideas and methods. Throughout the project and testing different components, the initial idea was modified and the cost analysis was changed accordingly. the table below is the summary of the initial cost analysis:

| **Number** | **Part Name** | **Quantity** | **Price** |
| --- | --- | --- | --- |
| 1 | Motors | 12 | 150$ |
| 2 | Batteries | 5 | 220$ |
| 3 | Flight Controller-Navio2 | 1 | 200$ |
| 4 | Raspberry Pi 3 | 1 | 35$ |
| 5 | Flight Controller-Pixhawk | 1 | 120$ |
| 6 | Propellers | 2 pack | 22$ |
| 7 | Pixy2 | 2 | 70$ |
| 8 | GPS Module | 2 | 25$ |
| 9 | Battery Charger | 1 | 16$ |
| 10 | Wireless Charging | 1 | 130$ |
| 11 | Ultrasound Sensors | 8 | 30$ |
| 12 | Lidar | 2 | 210$ |
| 13 | Other small parts and shipping costs |  | 272$ |
| Total: | | | 1500$ |

During the project development, some of the initial ideas were modified such as the charging system and battery power requirements. Also, some of the parts initially purchased did not have the functionality desired. Therefore, considering all factors the final cost analysis is:

| 1 | Initial cost analysis | 1500$ |
| --- | --- | --- |
| 2 | Final cost analysis | 1630$ |
|  | Difference | 130$ |

Throughout the development of the project the goal is to keep the cost low and use free resources available such as 3D printing available at CPP.

# **III. Systems Engineering**

Fig 1. Proposed scheme for autonomous docking and charging mission

## 1. Concept of Operations

### *A. Drone Build* Diagram Description automatically generated

This project consists of two drones; a charger drone (Biggy), and a receiver drone (Tiny). drones consist of a flight controller, a Raspberry Pi 3, a GPS module, power management board, four multi-copter drone motors and propellers, electronic speed controllers (ESCs), and a LiPo battery.

Tiny is designed for high maneuverability because of the need to quickly align with and adapt to the movement of Biggy when landing. As shown in Figure 2, this is accomplished by custom designing and 3D printing the frame out of PETG to be light and compact. It is also designed with three blade propellers for more control and thrust in the air.



Fig 2. Design of Tiny/Receiver Drone

Biggy is designed for load bearing and in flight stability. This is because Tiny must be able to easily track and land on it while it is airborne . As shown by Figure 3, the frame of Biggy is noticeably larger and prebuilt by YoungRC. On top of the pre-build frame is a plate for Tiny to land on. Biggy also has larger, two blade propellers to provide the drone more stability in flight.

`

Fig 3. Design of Biggy/Charging Drone

### 

### 

### 

### 

### *B. Landing Method*

A black cell phone

Description automatically generated with low confidence

Fig 4.1. Landing/Charging Platform Top View

A picture containing text, stationary, case

Description automatically generated

Fig 4.2. Landing/Charging Platform Bottom View

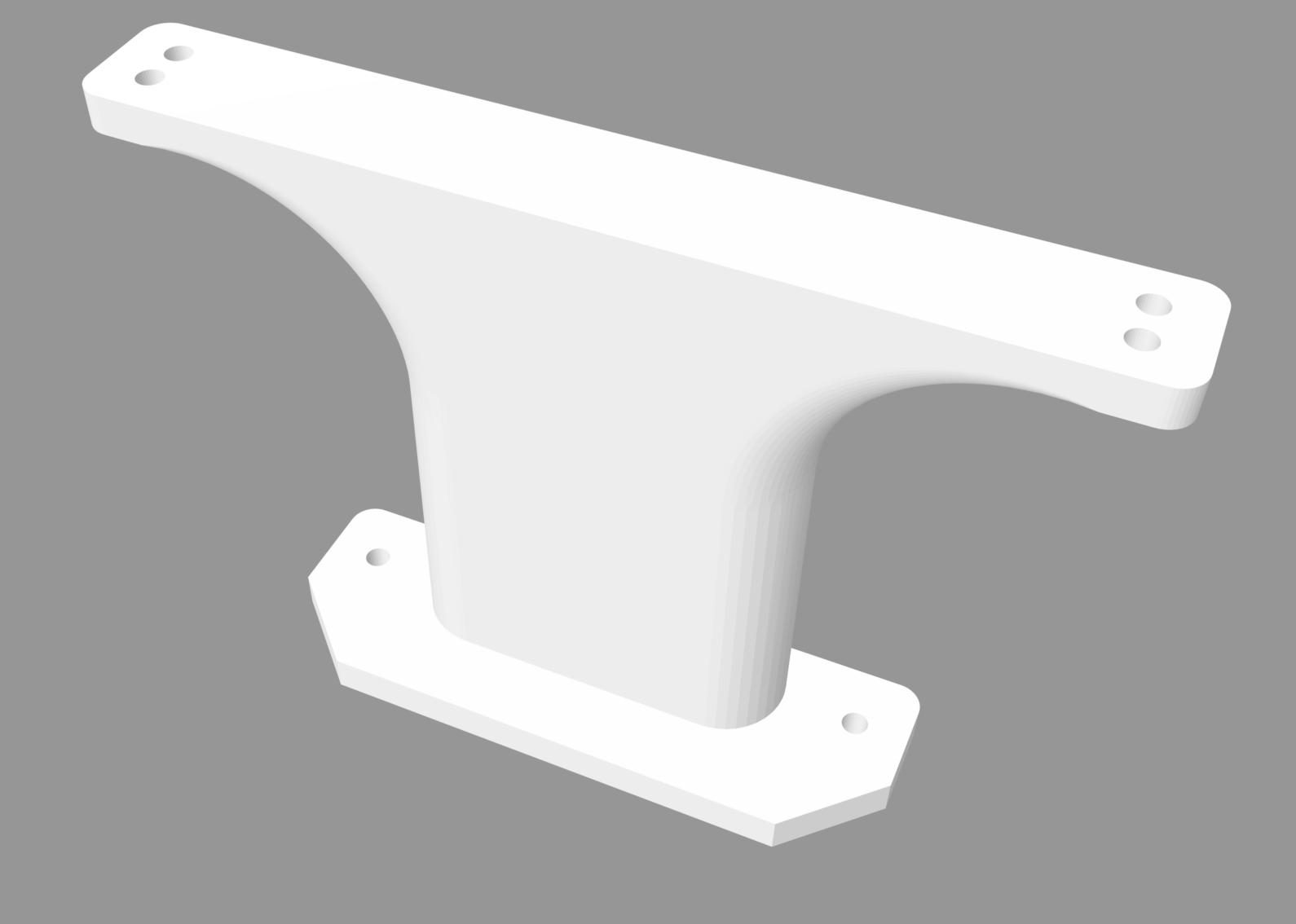


Fig 5. Landing Pad Standoff

The figures above show the landing platform and standoffs. We have custom designed these pieces to screw together and fit on our drone frame securely. Figure 4.1 and 4.2 are the top and bottom view of our landing pad, which has cutouts for our standoff pieces on the underside (Figure 4.2). We decided to add these slots to increase the stability, as we determined this design would keep the wobble from side to side to a minimum. The insets on the top (Figure 4.1) ensures we have a flat landing surface, and it provides a spot to screw our contact plates. The landing pad dimensions are 200mm x 200mm, and the standoffs (Figure 5) raise it up 70mm. These dimensions were ideal because it prevented the platform from interfering with the airflow of the propellers, while providing a large enough landing surface. At the same time, it was the perfect size to be printed on our Ender 3 Pro 3D printer, which has a printable area of 200mm x 200mm.

To ensure that the smaller receiver drone will land on the charging drone accurately, the system onboard has three main components: a GPS module, a Pixy2 camera, and ultrasound sensors. The main component for the landing method is the Pixy2, a microcontroller camera that can be used for object detection and barcode reading, more information about Pixy camera in appendices section B. The GPS module will be used to communicate coordinates between the drone and its user through Mission Planner [1]; that way, the user can know the drone’s location at all times. The ultrasound sensors will work as obstacle avoidance sensors to ensure the safest flight possible, more information about ultrasound sensors in appendices section C. For an accurate and safe landing, the receiver drone will use the ultrasound sensors to avoid obstacles and crashing while approaching the charging drone by using its GPS coordinates. The receiver drone will more precisely locate the landing pad on the charging drone shown in figure 4 and 5 by detecting a barcode that is on the pad using the Pixy2. It will then begin its descent and accurately land on the landing pad. The figure 6 below shows the general procedure to ensure safe and accurate landing.

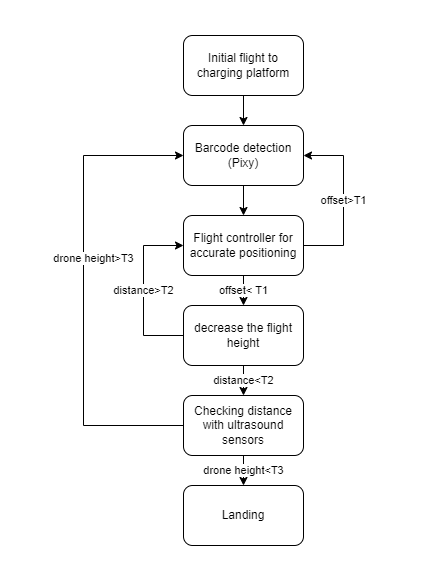


Fig 6. General Procedure Diagram for Accurate Landing

Each positioning cycle takes a certain amount of time since the drone needs to be positioned a few times to approach the target. When the barcode is detected an offset will determine if the positioning is accurate for landing and if the offset is not within acceptable range, this process will be repeated. Otherwise, the distance between the two drones will be determined using an ultrasound sensor for accurate and safe landing.

### *C. Power Transfer/Charging Circuit*

To start the process of power transfer from Biggy to Tiny, a charging circuit is built on both drones. The objective is to transfer power with copper contact plates. The power comes from a 4S LiPo battery and transfers into another 4S LiPo battery. A receiver circuit is built on Tiny using a 4S balance board to safely distribute power to each cell, and a 1N4007 rectifier diode[6] to limit the current to a max of 1A to prevent possible shorting of the battery. The transmitter circuit is found on Biggy and also requires a balance board like the receiver to protect the safety of the battery. As shown in figure 7, the transmitter side uses a LM7182, a linear voltage regulator[7] to cut off voltage when the battery is at 15V. This regulator will supply steady 12V to the MT3608, a step-up boost converter[8] which will convert the voltage to 18V for power transfer. This will allow us to distribute voltage to each cell in the LiPo battery through positive and negative terminals on aluminum contact pads found on top of the Landing platform of Biggy.

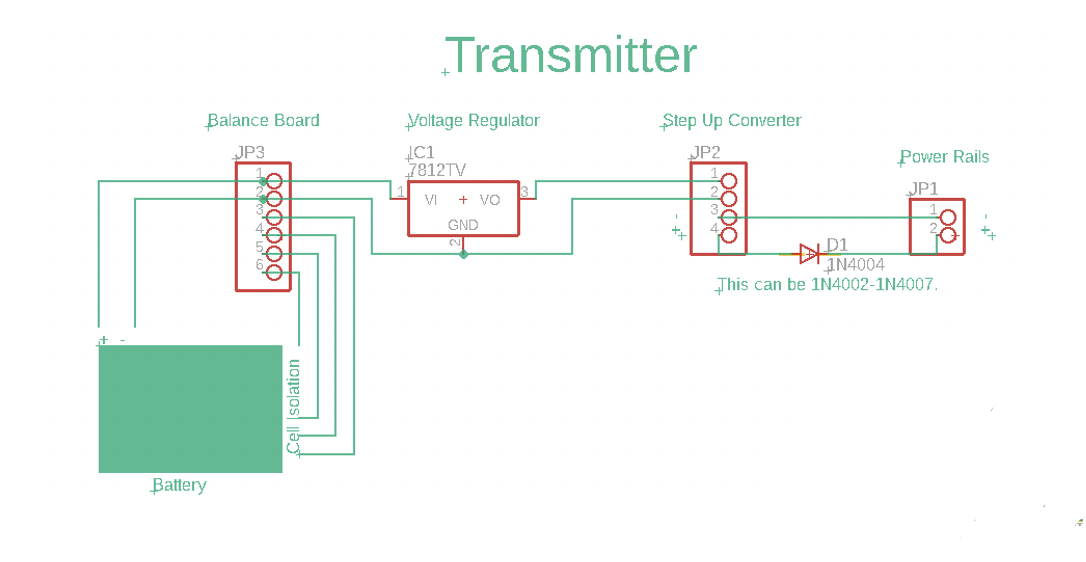


Fig 7. Schematic of Transmitter Circuit for Power Transfer

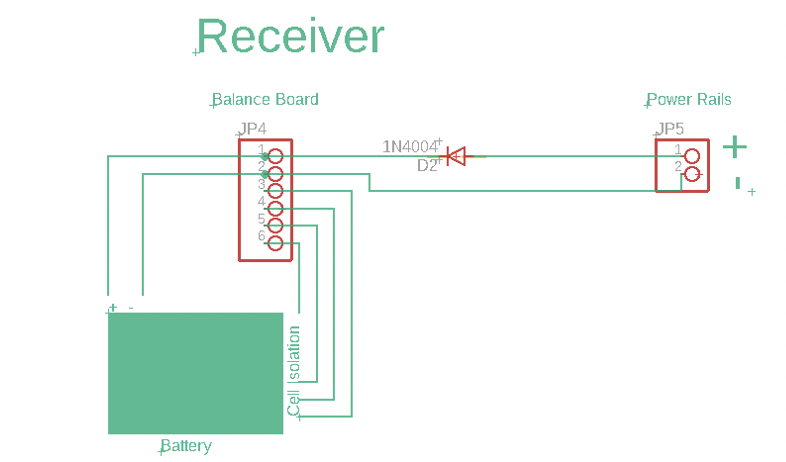


Fig 8. Schematic of Receiver Circuit for Power Transfer

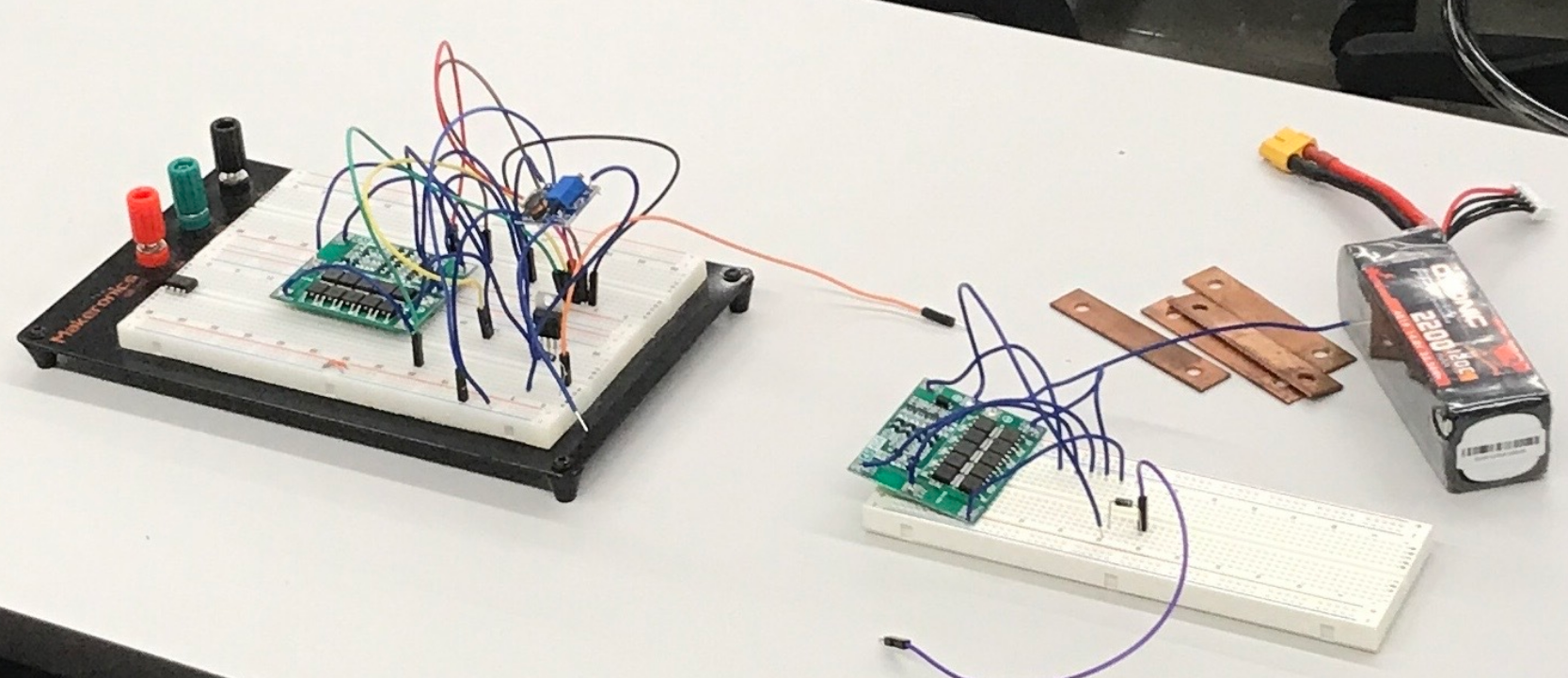
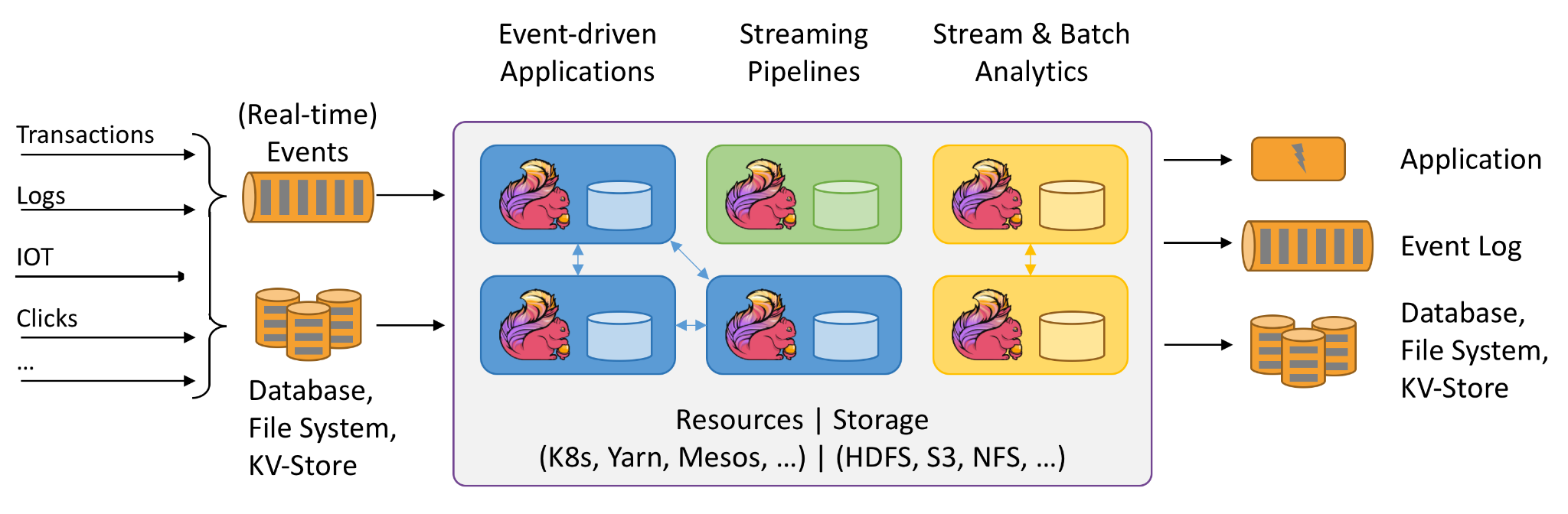


Fig 8.1. Power Transfer Circuit on Breadboard

### *D. Data Transfer/Apache Flink/Kafka*

To transfer data between Biggy / Tiny and a command center, there are various combinations of data streaming and processing software that could be used. For this project’s purpose, the ideal combination of software shall have enough speed on the transferring side to transfer a relatively small amount of data quickly, and the speed to quickly process the data being given. A low latency and stability is essential to ensure that the drone consistently receives accurate data of its relative position, else it risks crashing. By referencing benchmarks of Apache Flink, Spark, and Storm for processing software, and Kafka and Redis for data streaming software, the results of Apache Flink and Kafka line up with our necessities the most. Apache Flink as a processing software has low latency compared to other processing software, and has good stability when attempting to process larger pools of data. Kafka natively supports Flink, which eases the process of streaming data using Kafka[2].

Apache Flink aims to achieve its goal as a data processing software by automatically allocating “jobs” to different “Job Managers” on a Flink network, which in turn has the “Job Manager” allocate the work to an available “Task Manager”. This essentially is splitting the workload of a larger task to separate processing nodes. This approach makes it so that the main constraint of data processing is not the speed of the processors, but rather, the speed at which data can be transferred between the different nodes of the network. As a streaming software, Kafka can efficiently deliver data between the nodes, and because it natively supports Flink, there are less complications with getting data ready to be moved through streaming software.   
  
Fig 9. Apache Flink Architecture [13]

Apache Flink 1.14.3 requires the use of Java 8 or Java 11[10]. Through testing of the different libraries which are more compatible with the Raspberry Pi 3s used for the drones, it was decided that Java 8 would be used. Java 8 provided the least amount of compatibility issues when running jobs through Apache Flink[10]. In this current rendition of Apache Flink on the drones, Java is the base language when transferring the data and running jobs on the Task Managers. Kafka is able to use the most recent version, which is currently 3.1. Kafka is the software used with Apache Flink in order to stream data between the two drones. No compatibility issues were found when using Kafka 3.1. PyFlink was found to have a lot of compatibility issues. PyFlink currently requires the system to have Python3.6, Python3.7, and Python3.8[9]. When testing the installation of PyFlink, Python3.7 had certain libraries that weren’t compatible with the PyFlink libraries. Python3.6 and Python3.8 were found to have no installation issues. An issue with PyFlink being installed on our drones is the Raspberry Pi 3s are running on the current Buster OS. The Buster OS Python3 default is currently 3.7 and Python3.8 is fully supported on the Buster OS. Due to this issue, it was determined to not install PyFlink on the current drones until a successful installation can happen on a Raspberry Pi. PyFlink is a feature considered an upgrade to the Flink Drone system as many of the current codes for the drones use Python2.7. PyFlink would allow the drone system to run through Flink in the same language, reducing the latency time of data transfer.

## 

## 2. System Hierarchy

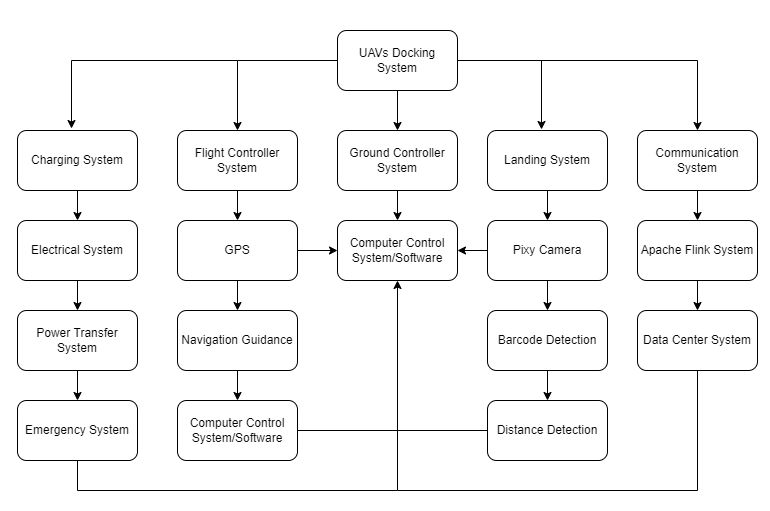


Fig 10. UAV Docking System Hierarchy

The system hierarchy is developed based on the key system requirements. This hierarchy is meant to be a representation of the most essential elements in the system and the relationship between the elements. The computer control system is the central unit that would communicate between different elements of the system and check various aspects of the operation as shown in figure 10.

## 

## 3. Interfaces

There are six main interfaces used to control and interact with the drones in flight or on the ground.

*A. Transmitter*

A six channel Flysky fs-16x transmitter is used to manually fly each drone. It also allows for programming switches on the transmitter itself which gives more control over the drone as well as the ability to switch between flight modes mid-flight.

*B. Mission Planner*

Both drones are able to interface with an open source ground control station application called Mission Planner. Mission Planner allows for the configuration, and calibration of the drones before flight to ensure that they are safe and ready to fly. It also makes it possible to give the drone simple autonomous missions through the ArduPilot firmware with little to no code writing allowing for testing the feasibility of the drones flying autonomously.

*C. Drone Kit*

Drone kit is an open source python library that allows for the drones to run autonomous missions through high level commands. It was used for flying Tiny in order to align with and land on a bar code in collaboration with the barcode reading library for the Pixy2 camera.

*D. Mav-Link*

Mav-Link is an open source vehicle communication protocol. It is used for linking together the Drone kit software which allows writing commands in python on the drone and the Ardupilot firmware which connects to the physical controlling of the drone itself.

*E. Putty*

Putty is an open source terminal emulator that is used on windows to SSH to the Raspberry Pi’s on the drone. This allows for connection to the drones in order to run commands through Drone-Kit or connect to the drones through Mav-Link.

*F. Flink*

Flink is used to send and receive commands and instructions between the drones in the form of jobs. It also allows us to communicate with drones from the ground.

## 4.. Technical Performance and Verification

*A. Initial Drone Test Setup*

Our initial drone testing was done with a pilot by having both Biggy and Tiny take off, do a simple flight and then land. This progressed to having the pilot manually have the drones take off then hold their positions in the air for a few seconds before coming back down and landing in the same place as they took off. This test was followed by giving the drones simple missions to autonomously take off, move to a location, hover for a few seconds and then return to where it launched from first using mission planner, and then drone kit. From here we moved onto testing our landing accuracy.

*B. Accurate Landing Test*

After the initial drone testing, landing tests were performed using a cardboard landing pad as shown in Figure 11 instead of the actual landing platform in order to avoid damages and collisions with Biggy. Barcodes were attached to the cardboard in order to simulate the landing pad and a Pixy2 camera was mounted on the drone in order to scan them. The purpose of this test was to ensure that Tiny could successfully land on a desired landing position when located above the landing pad using the Pixy2 camera to find the barcodes and land on them. After this testing phase was complete we moved on to testing our charging circuit.

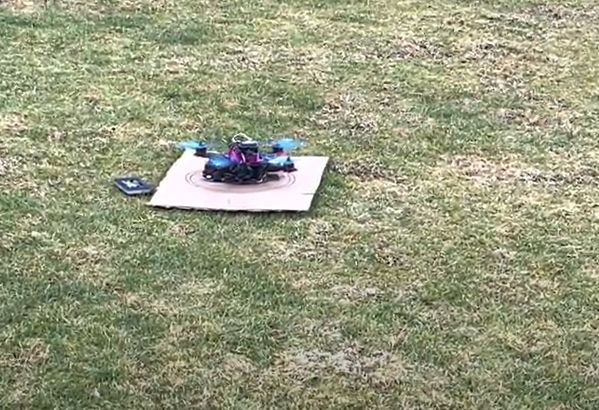


Fig 11. Landing testing

*C. Charging Platform Test*

The charging platform was also tested in stages. The first stage after building the circuit was to physically place all the copper rails on each other in their corresponding positive and negative polarities. The next step was to design a way to attach them to the drones so that the drones could still fly with the circuits on board. A final test was to physically place Tiny on Biggy to make sure that if the landing happened correctly then Biggy would begin to charge Tiny.  *D. Flink Test*

An initial test of the Apache Flink cluster system was set up on a virtual machine setup. This is due to initial concerns of possible issues on the Pi’s for the drones if Flink caused issues. Due to the limited quantity of Pi’s available, we didn’t want to risk testing the Apache Flink capabilities on the Drones. The test shown below captures what a Job in Apache Flink can do in our cluster, as well as seeing data being transferred between two machines. Apache Flink also depicts the speeds at which the Jobs complete.

Graphical user interface, text, application, email

Description automatically generated

Fig 12. Flink Test Code

Text

Description automatically generated

Fig 13. Submitting Job in Flink

Fig 14. Flink DashboardA screenshot of a computer

Description automatically generated

Graphical user interface, application

Description automatically generated

Fig 15. Results/Speed of submitted Jobs

Fig 16. Results of the data Transfer A screenshot of a computer

Description automatically generated

## 6. Trade Studies

*A. Charging Methods*

A few methods of charging were considered throughout the course of the development of this project. The methods discussed below were discarded after studying and careful consideration.

The first method is a simple tether cable. Similar to mid-flight refueling of jets, both drones connect via a cable and transfer power, once the docking was successful. Magnets would be utilized to secure the connection between the two drones and create a magnetic field. However, the efficiency of this proposed method would require accuracy that may be hindered by laws of aerodynamics. Factors such as the shape and size of the object, velocity and inclination to flow, and the properties of the air such as mass, compressibility, and viscosity would influence the motion of the UAVs while in air.

The second possible charging method is inductive charging. This is a method that is found in wireless charging systems to charge short range, low power devices such as phones, toothbrushes, electronic tools, and medical devices. This method seemed to be an ideal solution for the charging method in this project, however distance was a factor that would affect the efficiency of power transfer. Because of this the alignment of the transmitter and receiver coil would affect the outcome meaning that the receiver drone with the coil would have to land on the charging drone successfully and align with minimal misalignment to transfer power efficiently. The further the coils are, the less efficient the transfer is[3].

*B. Navio2 vs Pixhawk*

In this project two different flight controllers are used to design the drones, a Navio2 and a Pixhawk. Below is the analysis of both flight controllers for the purpose of future drone builds to create a more efficient robotic environment. The table below summarizes and compares some of the useful futures available on both flight controllers.

| **Feature** | **Pixhawk** | **Navio2** |
| --- | --- | --- |
| Affordability | 120$ | 200$ |
| Weight | 60 grams | 80 grams |
| I/O ports | Plenty of I/O ports | Limited |
| Open source firmware | Yes(Ardupilot) | Yes(Ardupilot) |
| Autonomous functionality | Yes | Yes |
| Linux | No | Yes |
| Processor | 32 bit | 64 bit |

Table 1. Comparison Between Navio2 and Pixhawk

The first noticeable difference is the price, Pixhawk is much more affordable than Navio2 and Navio2 requires a Raspberry pi as well to function. The second difference is the weight, Navio2 is heavier than the Pixhawk and can affect the flight time and power consumption. After mounting the Navio2 on Raspberry pi there are only few I/O ports left for other functionalities and this can be limiting if other components are required to be added in the project. One of the advantages of Navio2 is that the firmware runs on a Linux environment unlike Pixhawk. This gives the option of easy setup and interaction such as ssh during flight[4, 5]. After reviewing this analysis the decision was to use Navio2 on the Biggy (the charger drone) and Pixhawk on the Tiny (the receiving drone). Since affordability and increased flight time is one of the main focuses of this project, for the future builds the Pixhawk is the better option.

## 7. Reliability and Safety

Dealing with systems as dangerous as drones, plenty of safety precautions were put in place to ensure the welfare of the team as well as surrounding life. Two types of protection were taken into account when building and testing the drones: digital and physical. Digital safety includes systems not directly controlled by people; whereas physical safety would be the contrary.

Digitally, the drone has many on-board measurement systems to keep track of itself, including a GPS, an onboard compass, an external compass, a gyroscope, a battery sensor, and barometer. Built into Mission Planner are fail safes based on the readout of these on-board systems. These data outputs are crucial in providing safety due to their ability to be used in creating fences which can limit height, yaw, pitch, roll, thrust, and distance. These limits are used to create limited variability, thus making the drones more predictable and safer. For example in the event of a crash, the outputs from the gyroscope, compass, and GPS would flag as being at critical levels and have the ability to turn the motors off to prevent further harm to the drone or any surrounding people. The automated portion of our testing has utilized these digital resources heavily to create a digital safety net; one that would keep these values in a more narrow desirable range.

As for the physical considerations, numerous precautions were taken to establish a safe environment. When testing automated code, there is an increased chance of the drone flying in an arbitrary direction with no control of it. In order to account for this, a light, yet strong, rope was attached to the drone being tested to establish a backup manual form of control. Additionally, members of the team that test fly the drones received an aeronautical knowledge and safety test certification from the Federal Aviation Administration (FAA). This test is called “The Recreational UAS Safety Test” also known as TRUST. In passing this test, these pilots have proven they fully understand important safety and regulatory information regarding drone flight.

In the event of a system component failure during automated or manual flight, the controller in the hands of the pilot was set up with a disarm switch, with the ability to shut down the drone at any point. This precaution is extremely important as it may be the last resort if any other failsafes malfunction. The last physical safeguard employed by the team was environmental safety. This was achieved by flying in open non-restricted fields, keeping safe distances from others, and taking into account any wind or other aircraft. This project kept safety its highest priority throughout its span, resulting in no injuries.

# **CONCLUSION**

This project explores an integrated system that enables a drone to land on a charging platform for autonomous charging and data communication. The autonomous charging improves the flight time and faster data communication adds efficiency to the drone system.

The charging system includes a charging platform with barcodes installed on the platform. The landing procedure utilizes the barcode detection, position estimation and at the same time to improve the landing accuracy by checking the distance using an ultrasound sensor. For the charging procedure, copper contact plates are used to transfer powers between the two drones. Data communication is done through Apache Flink which will create a data center for all the interactions between the drones.

# **APPENDIX**

*A. Future Work*

The future work for this project is to create a collaborative robotic environment that can be implemented in deep space. The plan is to extend this project and include rovers in the system and implement the same concept of docking with rovers. The vision is to have the receiver drone to dock on a rover for charging and data communication. To secure the communication in this environment a post quantum level encryption will be used. This will allow the drones to create a secure cluster that can be used in more vulnerable enterprise level or military level positions.

*B. Pixy 2 Camera*

The Pixy2 is a small camera designed for object recognition, line tracking, and simple barcode reading. The camera is capable of recognizing seven distinct objects based upon their shape and color. Each of these objects is assigned a unique “signature”. The camera also has algorithms for line following. The camera is also compatible with raspberry pi which is an important feature for this project[11].

In addition to having all of the features of the original Pixy Cam the Pixy2 has these additional functions:

* Detects lines, intersections and simple barcodes.
* Has a framerate of 60 frames per second.
* Has an integrated light source consisting of two white LEDs.
* New tracking algorithms that allow you to track objects.

*C. Ultrasonic Sensor*

Ultrasonic sensors are commonly used for application such as measuring distance and noncontact presence. These devices typically transmit a short burst of ultrasonic sound toward a target, and the sound is reflected back to the sensor. The system then measures the time for the echo to return to the sensor and computes the distance to the target using the speed of sound in the medium. In an echo ranging system, the elapsed time between the emission of the ultrasonic pulse and its return to the receiver is measured. The range distance to the target is then computed using the speed of sound in the transmission medium, which is usually air. The accuracy of the target distance measurement is directly proportional to the accuracy of the speed of sound used in the calculation[12].

# ACKNOWLEDGMENT

This project is sponsored by NASA MINDS, The Reconfigurable Spacing Computing Laboratory at CPP, RECOIOT, The Coordinated Science Laboratory at University of Illinois, Air Force Research Laboratory (AFRL) and our faculty advisor Professor Mohamed El-Hadedy (Aly). A very special thanks to Mr. Christopher J. Watson for guiding us with our circuit designs and our as well as the flight stability of our drones.

# **REFERENCES**

[1] *“Mission Planner Overview”* Accessed on: February 21, 2022. [Online]. Available: https://ardupilot.org/planner/docs/mission-planner-overview.html

[2] “*Apache Flink-Stateful Computation Over Data Streaming*,” Accessed on: March. 15, 2022. [Online]. Available: https://flink.apache.org/

[3] Robert Clarke. “Autonomous multi-rotor aerial vehicle with landing and charging system”, May 18 2017. US Patent App. 14/858,795.

[4] “*The Open Standard for Drone Hardware*,” Accessed on: Feb. 3, 2022. [Online]. Available: <https://pixhawk.org/>

[5] “*Autopilot Hat for Raspberry Pi Powered by ArduPilot and ROS*,” Accessed on: Feb. 3, 2022. [Online]. Available: <https://navio2.emlid.com/>

[6] “*1N4007 General Purpose Rectifiers Data Sheet*,” Accessed on: Feb. 14, 2022. [Online]. Available: <https://www.mouser.com/datasheet/2/149/1N4007-888322.pdf>

[7] “*LM7812 Datasheet*,” Accessed on Feb. 14, 2022. [Online]. Available: <https://www.mouser.com/datasheet/2/149/LM7812-461970.pdf>

[8] “*MT3608 Datasheet*,” Accessed on: Feb. 14, 2022. [Online]. Available: <https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf>

[9] “*Installation*,” Installation | Apache Flink. [Online]. Available: https://nightlies.apache.org/flink/flink-docs-release-1.14/docs/dev/python/installation/

[10] “*First steps*,” First steps | Apache Flink. [Online]. Available: <https://nightlies.apache.org/flink/flink-docs-release-1.14/docs/try-flink/local_installation/>

[11] “*Pixy2 Camera-Object recognition with Arduino & Raspberry Pi*,” Accessed on: Feb. 14, 2022. [Online]. Available: <https://dronebotworkshop.com/pixy2-camera/>

[12] Massa p. Donald. “*Sensors Insight*,” Accessed on: Feb. 14, 2022. [Online]. Available: <https://www.fierceelectronics.com/components/choosing-ultrasonic-sensor-for-proximity-or-distance-measurement-part-1-acoustic-0>

[13] “Stateful computations over data streams,” *Apache Flink*. [Online]. Available: https://flink.apache.org/