Gesture Operated Drone

Anshul Devnani, CPE
Pranay Jay Patel, CPE
Bernardus Swets, CPE

Group 3

Project Description

- DIY Indoor Drone
- Remotely Controlled
- Operated by Hand Gestures
- User-Friendly GUI to show live drone info

Motivation

- Drones are not easily flown
  - Can cause safety concerns if incorrectly flown
- Remote Control is not directly intuitive, takes practice
- Allows for free hands while controlling the drone
- Puts together concepts learnt throughout our coursework that interests us. (Machine Learning, Wireless Comm, Embedded Systems)

Goals and Objectives

- Improve ease-of-use for drones
- Achieve quick response time to gesture commands
- Build GUI that is understandable to the untrained eye
- Maintain stability during drone flight
- Sustain connection to drone at usable distance

Software Requirements

- The drone will be able to convert the signal received over Bluetooth within 500 ms.
- The user interface will be able to recognize each of the 8 gestures.
- The feedback/reading pane will highlight the correct predicted gesture within 1 second.
- The Neural Network will produce an accuracy of a minimum of 95 percent.
- The user interface GUI will consist of a webcam pane, log pane, and miscellaneous pane.

Hardware Requirements

- The drone frame will be no larger than 150mm.
- The drone will not weigh more than 2 pounds.
- The drone will be powered by 3.7V lithium polymer batteries.
- The microcontroller will be powered by a 9v DC battery.
- The drone will utilize propellers of 3 inches or smaller.
- The drone will utilize 4 electronic speed controllers to help control the propellers.
- The drone will utilize 4 brushless motors with KV above 900.
- The drone will utilize an ATmega328P microcontroller

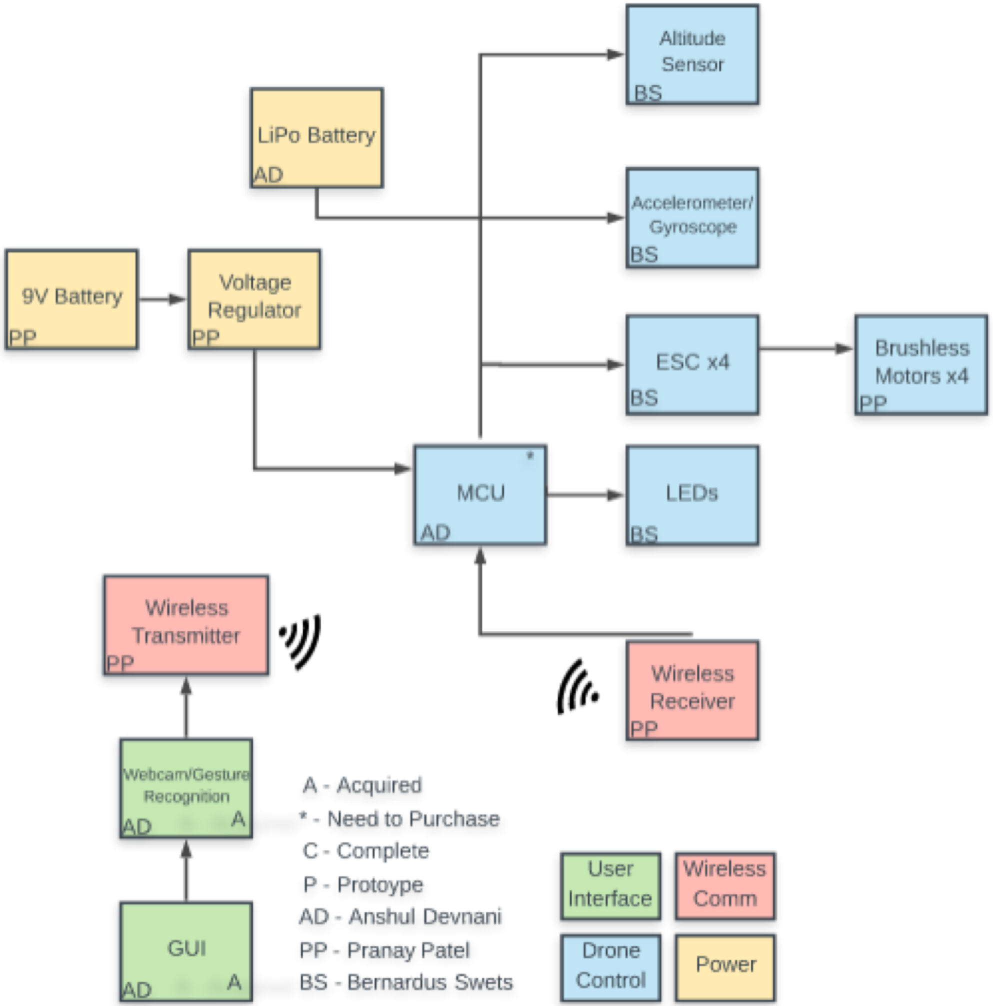
System Requirements

- The drone will be able to receive signals over Bluetooth communication from within a range of 20 feet.
- The drone will be able to react to commands within 1 second.
- The drone will be able to land, and motors will terminate within 5 seconds.
- The drone will be able to take off to 3 feet within 3 seconds.
- The Bluetooth signal will maintain connection within 15 feet.
- When the drone’s Bluetooth signal is lost, the drone will hover in place and land within 10 seconds.
- The time from the user doing the gesture to the drone reacting to it will be a maximum of 2 seconds.
- The drone will communicate its current altitude to the GUI with a maximum latency of 3 seconds.
- The drone will maintain its altitude when moving left, right, forwards, and backwards.
- When the drone accelerates in a specific direction, it will rotate less than 90 degrees to perform the given action, as to not tip the drone over.
- The drone’s altitude will be able to be read with a maximum 1 second delay on the miscellaneous pane of the GUI.
- The drone will be able to reach a height of 10 ft.

Constraints

Constraint	**Value**
Drone Laws	Flying Outdoors
Wireless range	Less than 30ft in either direction
Drone Frame Size	Less than 150mm
Drone Battery Runtime	20 Minutes
Drone Weight	Less than 2 pounds
Number of Gestures	At least 7 gestures, but limited, as similar gestures may be hard to differentiate by webcam
Budget	Affordability

System Design



Gesture Recognition

Gesture Recognition Goals

Two Main Goals:

1. Should be accurate regardless of skin color and background
2. Recognition needs to be done in real time

Ways To Achieve Recognition

Two most popular approaches - our pick Neural Networks

- More complex code, can lead to accuracy errors
- Not easily scalable
- Teaching the computer what to look for

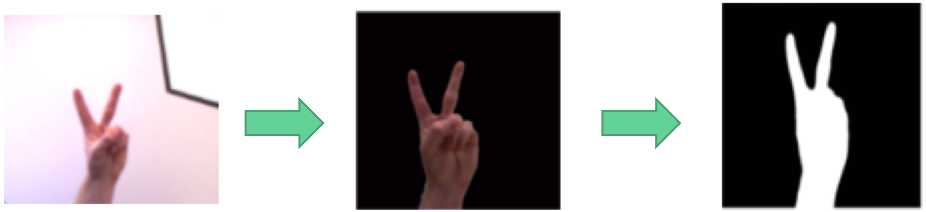
- Minimal code complexity
- Easily scalable
- Computer learns what to look for, let the computer do the hard work

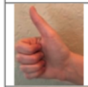
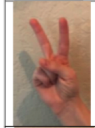
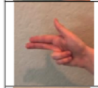
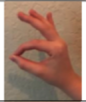
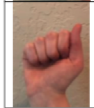
Convolutional Neural Network

- There are many different types of neural networks, choice is based on the problem at hand
- Most popular machine learning technique for image recognition is using a Convolutional Neural Network
- CNN’s use images from a dataset (hand gestures) and learns what makes each subset different from each other (palm vs thumbs up) by extracting features from each image (through convolutional operations) - Known as the training phase
- CNN’s are then tested on images it has not seen to verify how well the model trained - Known as the testing phase

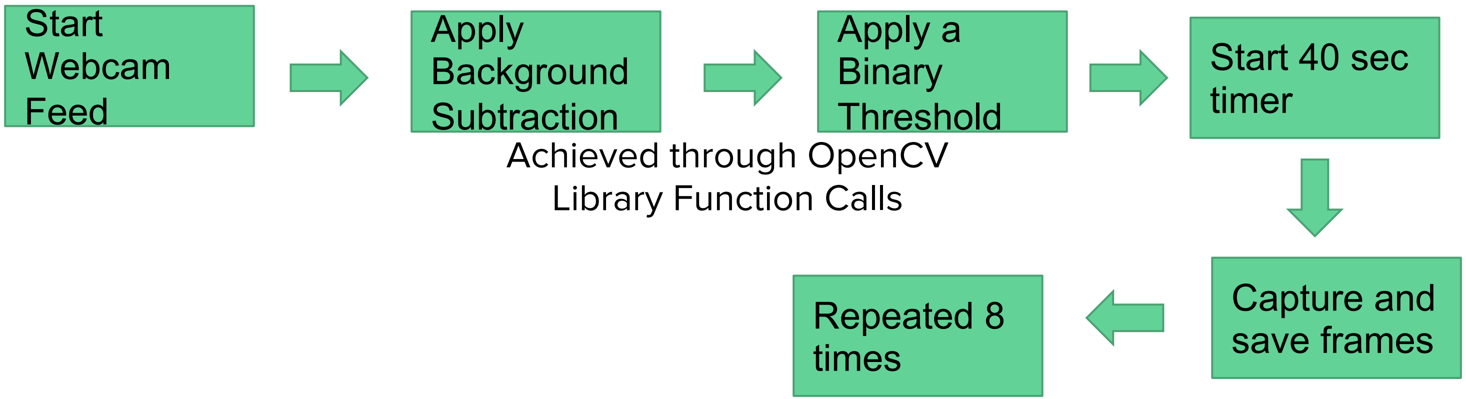
Building the Gesture Recognition Application

- 1st step is to build our own dataset
- Characteristics of a good dataset
    - Provide good coverage
        - 1000+ images per hand gesture
        - Different angles and focuses
    - Independent of skin color and background
        - Background Subtraction -> Binary Threshold



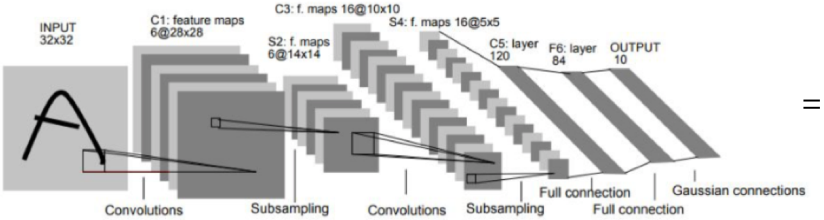
User Action	Result
No Gesture	Hover in place/autolevel
	Thrust Upwards
	Drone flies forwards
	Drone flies to the left
	Drone flies to the right
	Drone lands in current position
	Drone flies backwards
	Thrust down

Python Utility To Build Dataset



Building the Gesture Recognition Application Cont.

- 2nd Step is to build the CNN architecture/model
- Two Options: Create your own vs. Use existing architecture*
- LeNet-5 Architecture
  - Used for handwritten digit recognition and classification
  - Shallow - Only 5 layers - Good for real time recognition
  - Research shows that this architecture achieves good accuracy rates
- How to code?
  - Use Keras + TensorFlow frameworks to build in Python

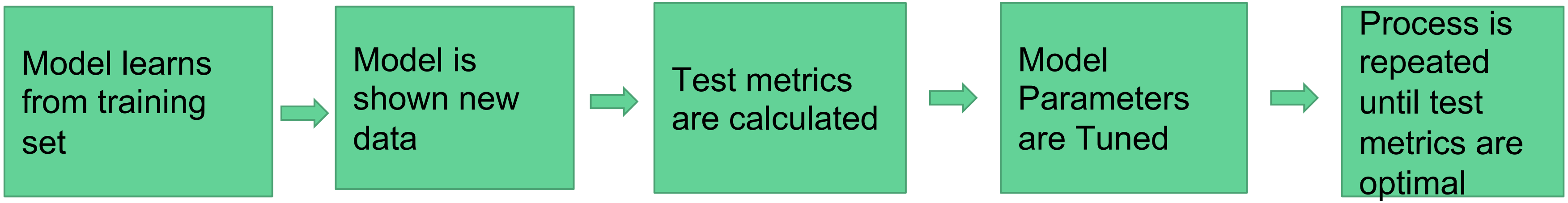


Original Image published in [LeCun et al., 1998]

= 10 lines of code using Keras

Building the Gesture Recognition Application Cont.

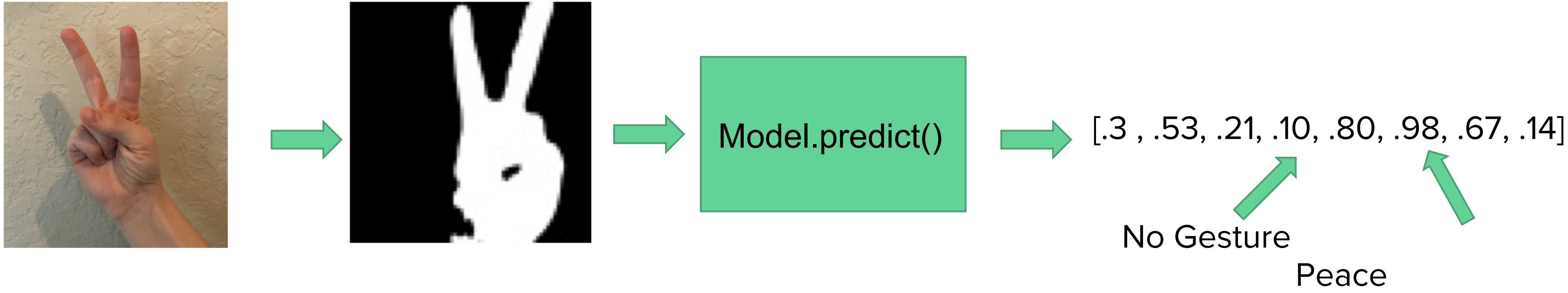
- 3rd step is to train and test the CNN model
- Training is done by splitting the dataset into two subsets. Training set and Validation set
  - 80 - 20 training to validation split



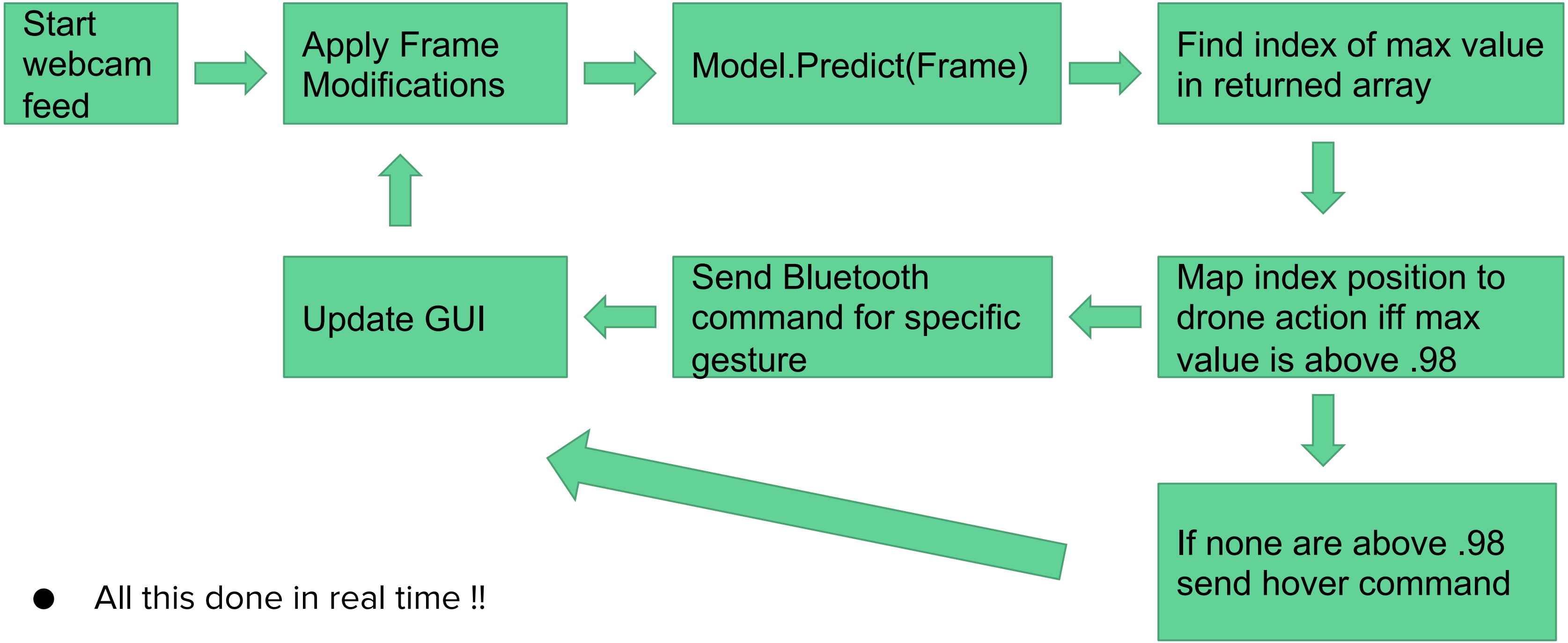
- Model parameters are saved

Building the Gesture Recognition Application Cont.

- Final step is to use the saved model to predict hand gestures in real time



Gesture Recognition Application Flowchart

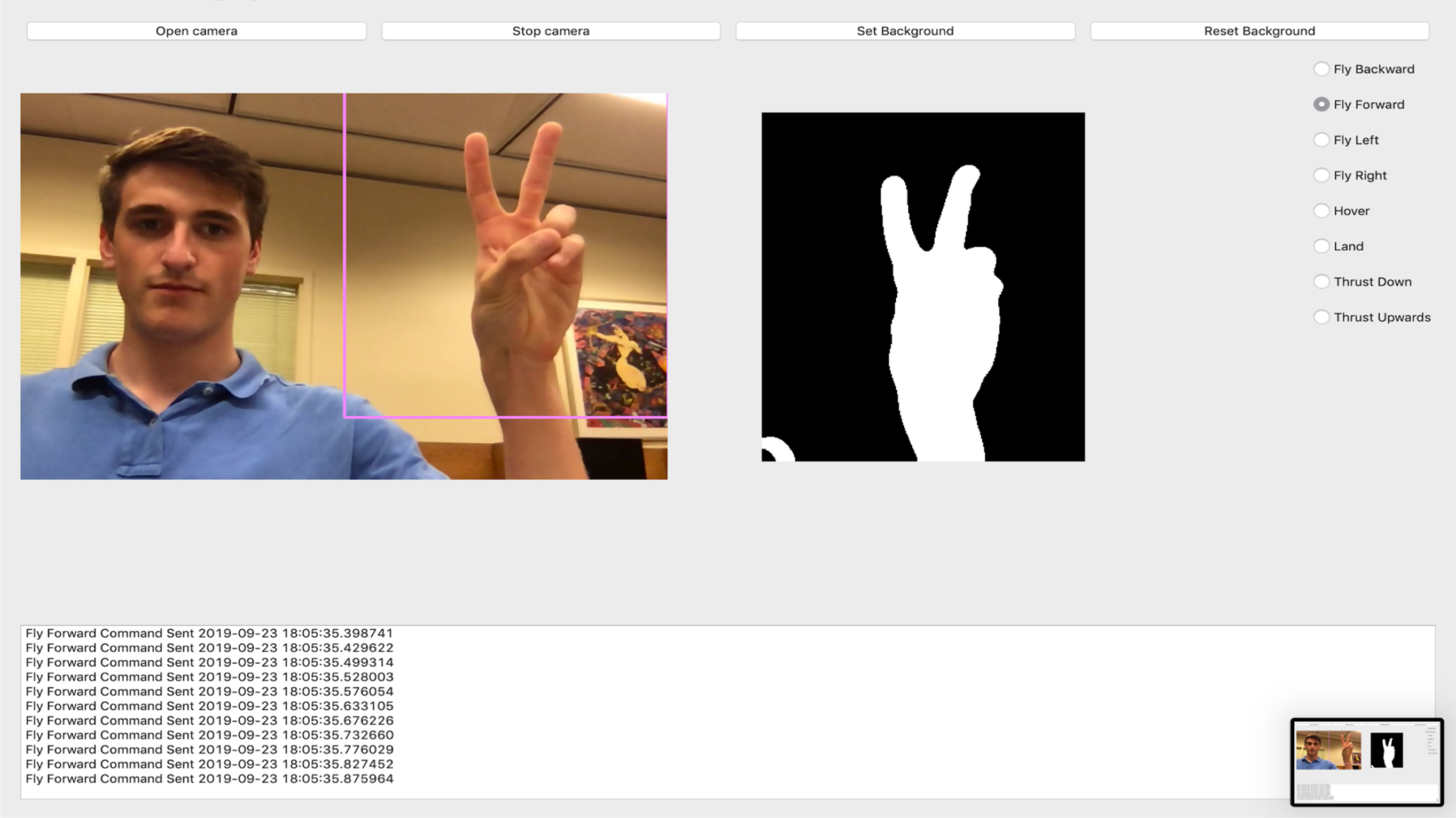


- All this done in real time !!

GUI

- Increase organization and usability
- Used PyQT for designing layout

GUI Prototype



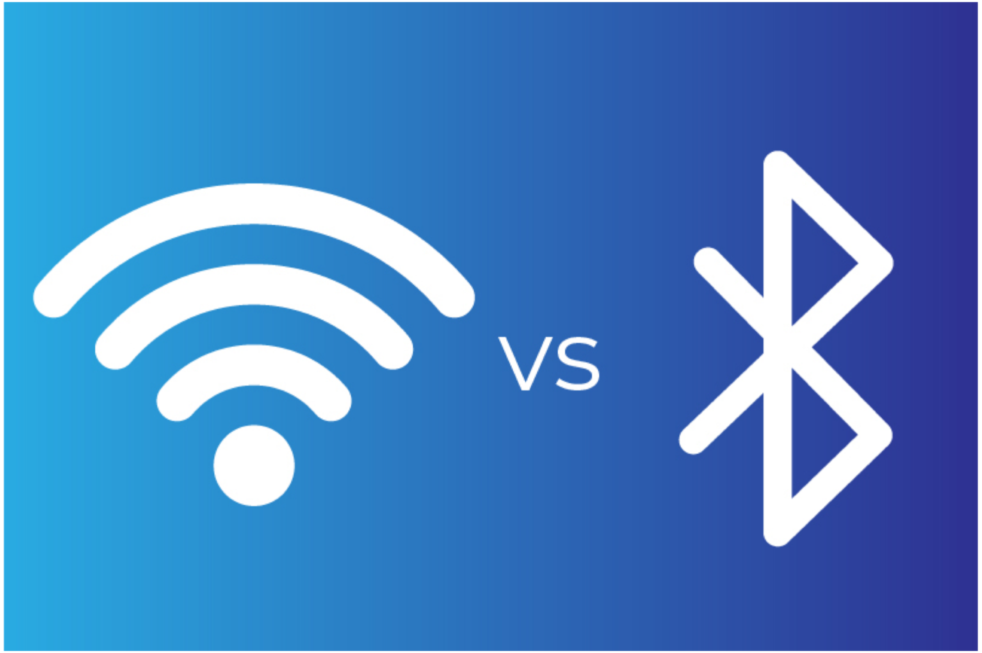
Wireless Communication

Wireless Communication - Overview

- Which type of communication should we use?
- Why did we choose that type?
- What parts are we using to do this?
- How will it be configured?
- What are its limitations?

Which type of Wireless Communication?

- WiFi
  - Industry leading
  - Allows for longer range
  - Expensive
- Bluetooth
  - Ease-of-use
  - Familiar
  - Cheap
  - Shorter range

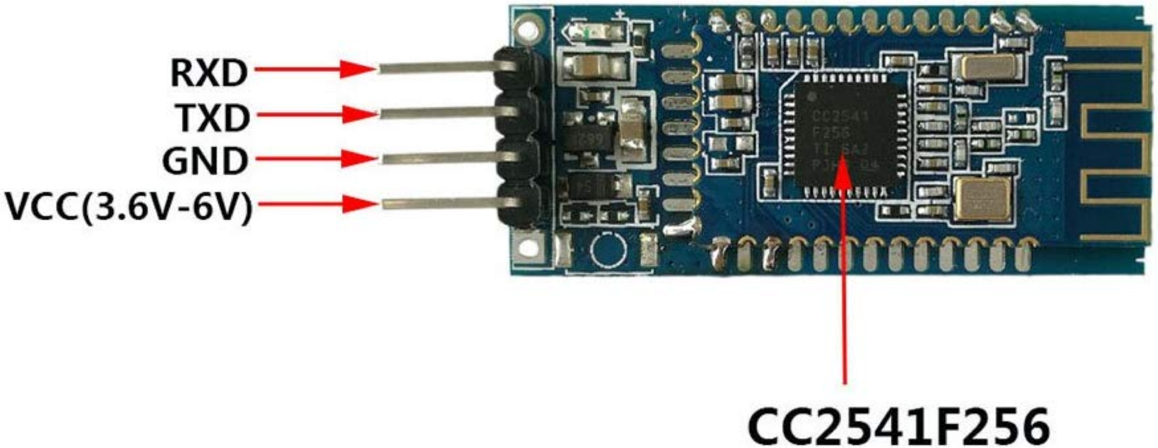


Why Bluetooth?

- IEEE Standard
- Widely used
- User-friendly
- Cheaper
- Continually updated
- Backwards Compatibility

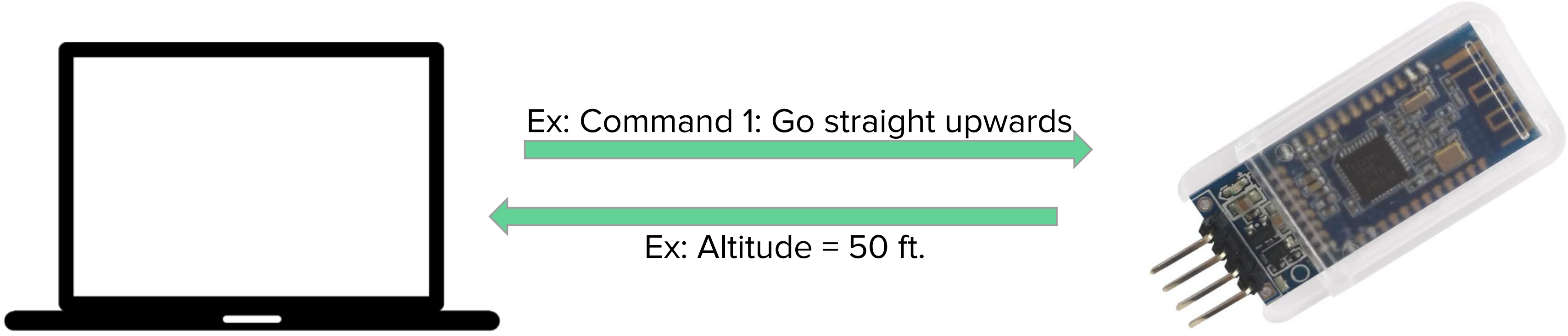
What Parts Are We Using?

- JBtek HC-05 Bluetooth Module
  - Reason for Choosing: Widely Used, Cheap, Reliable
  - Cost: $10
- Apple MacBook Pro - 2017
  - Built-in Bluetooth
  - Confirmed Compatibility



Configuration

- MacBook as Master
- Bluetooth Module as Slave
- Two-way communication over UART
- Bluetooth module serves altitude data to MacBook
- MacBook serves gesture commands to Bluetooth module



Limitations - Data Complexity

- Bluetooth is not built for passing large amounts of data quickly
- Due to this, we opted to make the data communicated minimal
  - Alternative to finding the Bluetooth module that had the fastest data throughput
- The data we are passing will be a few bytes of data
- We intend to send it as often as we can within reason
  - Ties to power consumption
- With this solution, we circumvent Bluetooth’s constraint of data throughput

Limitations - Distance

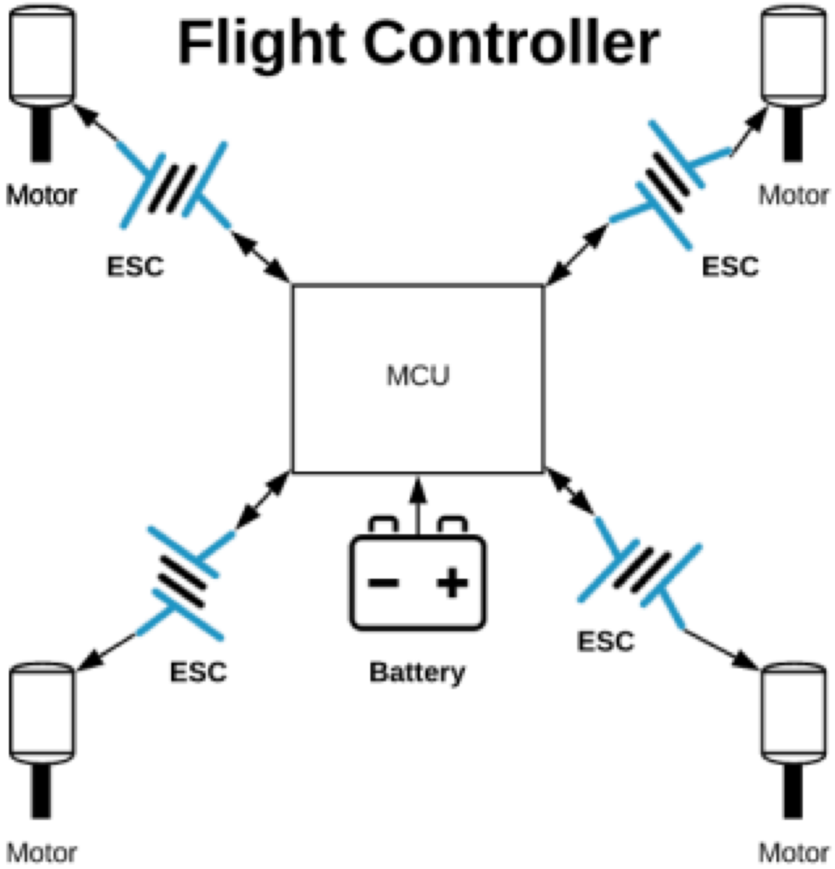
- Recall:
  - Objective of project is to sustain connection at usable distance
- Due to this, connection range was of primary concern
- Based on this, we searched to find a module that had a usable range
  - With the caveat of being affordable

Limitations - Power Drain

- Recall:
  - Objective of project is to have a reasonable flight time
- Due to this, we plan to reduce the power consumption of the Bluetooth module
- To overcome this, we will use various power-saving modes of Bluetooth
- We will also reduce transmission repetition as much as we can

Drone Design

Flight Controller



Drone Frame

Key Components

- Fiber Reinforced plastic, opposed to carbon fiber
  - Affordability and no RF interference
- 450 mm
- Four Arm Drone
- Integrated PCB connections
- Colored Drone Arms to eliminate the need for directional LEDS

Drone Frame Specifications

Width	450mm
Height	55mm
Weight	280g
Manufacturer	YoungRC
Price	$18.99

Drone Motors

Key Components

- As light as possible, ideally less than 2 oz each
- Brushless motors for increased efficiency, & outrunner motors for increased torque
- Around 1000 KV of voltage is ideal
- Compatible with 8-10 inch propellers
  - Stator size greater than 2200
- At least four motors are required and a lower cost will help lower our project budget

Drone Motor Specifications

Weight	1.5 oz
Stator Size	2212
Max Efficiency	80%
Current Capacity	12 A
Type	Brushless
KV	1000
Dimensions	45x24x11 mm
Manufacturer	Hobbypower
Price	$40.00

Electronic Speed Controller

Key Components

- Compatibility with our motors
  - Max voltage shouldn't be greater than the motor
  - Extra wire for brushless motors
- Lesser discharge current than battery for a more efficient design
- As lightweight and small as possible
- Greater than or equal to amp rating of 20A
- Affordable cost as four are needed for the quad copter

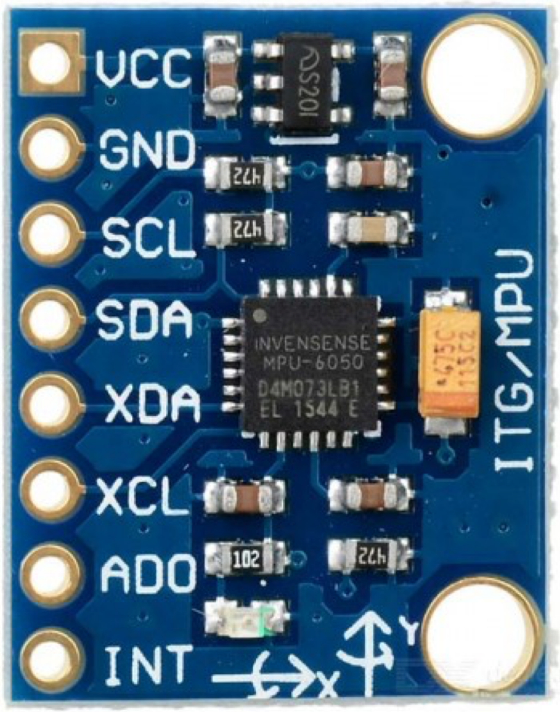
Electronic Speed Controller

Weight	4.5 oz
Amp Rating	30A
Dimensions	2.1x1.0x0.5 in
Manufacturer	Hobbypower
Price	$16.00

Gyroscope Accelerometer

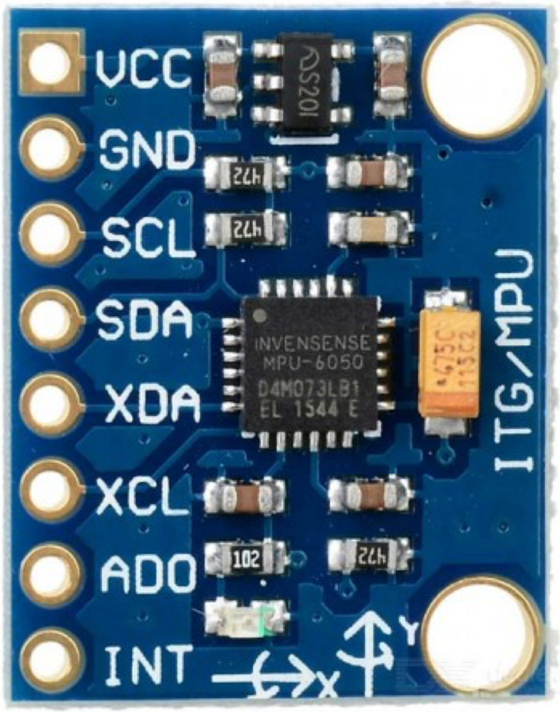
Key Components

- I2C communication protocol
- At least 6-axis as extra 3-axis compass is not required
- Small and lightweight as possible
- Low cost and reliable source
- Operating voltage 3.3v



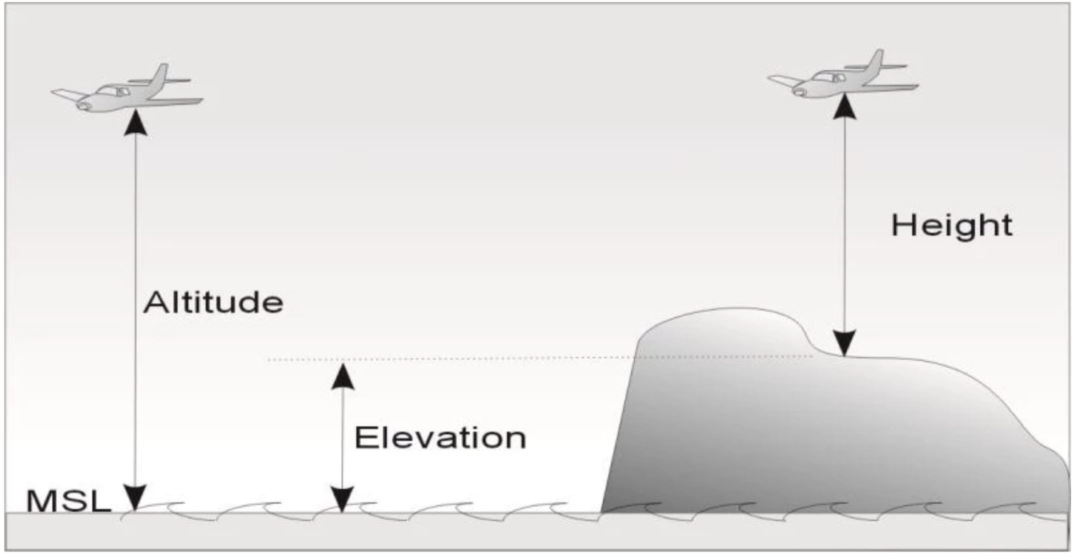
MPU-6050 Specifications

Operating Voltage	3.3V
Comm. Mode	I2C
Dimensions	21.2x16.4x3.3 mm
Axes	6-axis
Weight	2.1 g
Manufacturer	InvenSense
Price	$8.31



Altitude Sensor

- Purpose
- Choice of parts
- Implementation
- Price



Altitude Sensor - For What?

- Provide real-time info about the drone to the GUI
- Allows us to monitor the drone’s location better while controlling it
- Confirm drone is flying within:
    - Recommended altitude constraints
    - Lawful drone heights
- Can be expanded upon to provide temperature info

Altitude Sensor - Which One?

- JBtek BMP180 Barometric Pressure, Temperature, and Altitude Sensor
  - Reason for choosing: Pre-built library to convert data into human-readable altitude
  - Made for Arduino
  - Cost: $6
  - Operating Voltage 3.3V
- Sparkfun library is already made to convert the data
- This type of sensor is the most commonly used for this purp

Altitude Sensor - Software Implementation

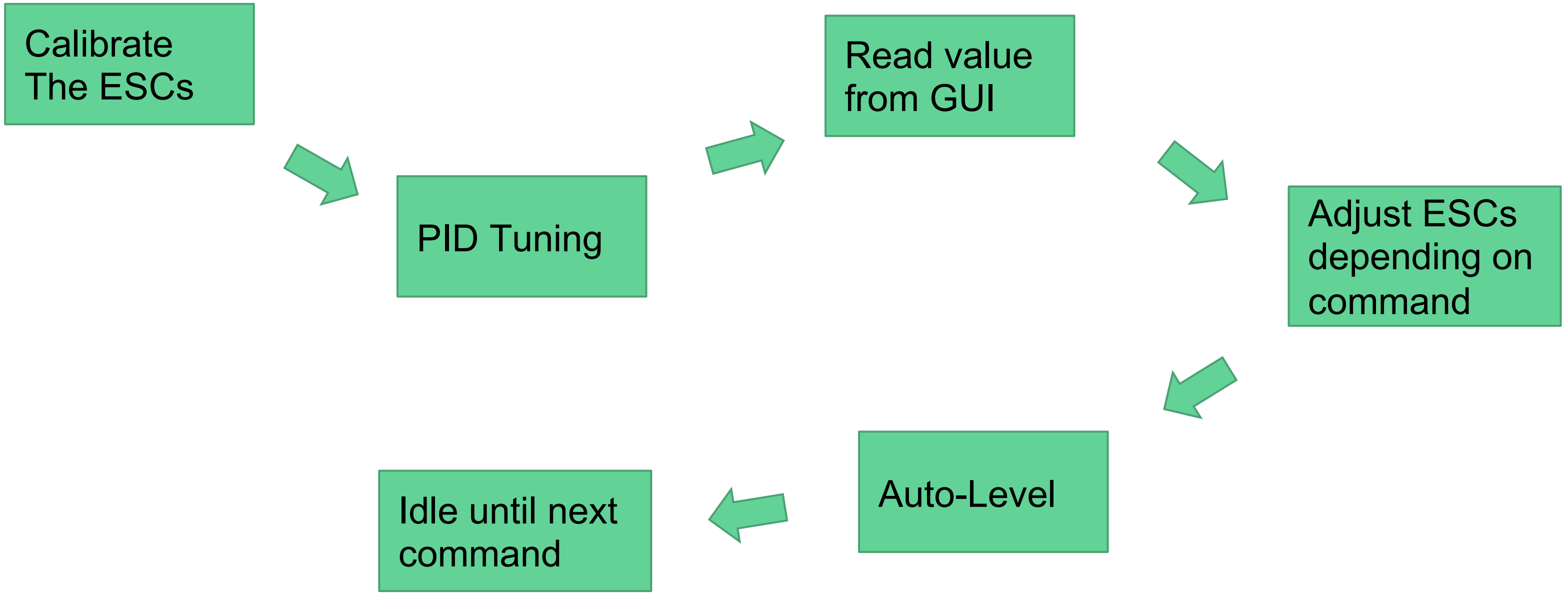
- SparkFun Library already built to convert temperature/pressure data to altitude
- We will simply zero out the altitude before the launch process has begun
- Use library to convert to floating point, send floating point via Bluetooth

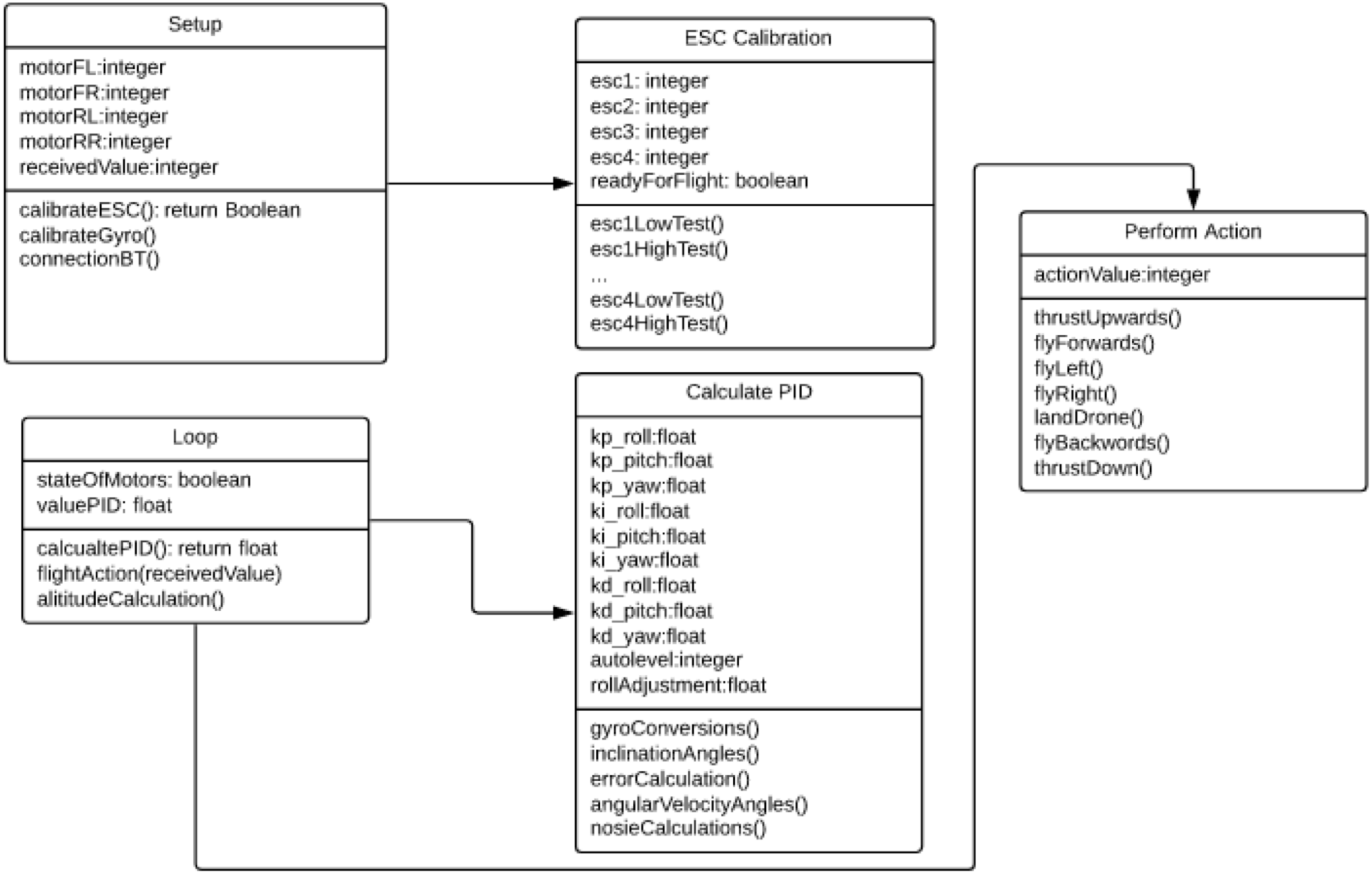
Flight Controller

- We are designing our own flight controller through the Atmega328p microcontroller. Can be tested using the Arduino Uno board
- Microcontroller must be able to handle the workload

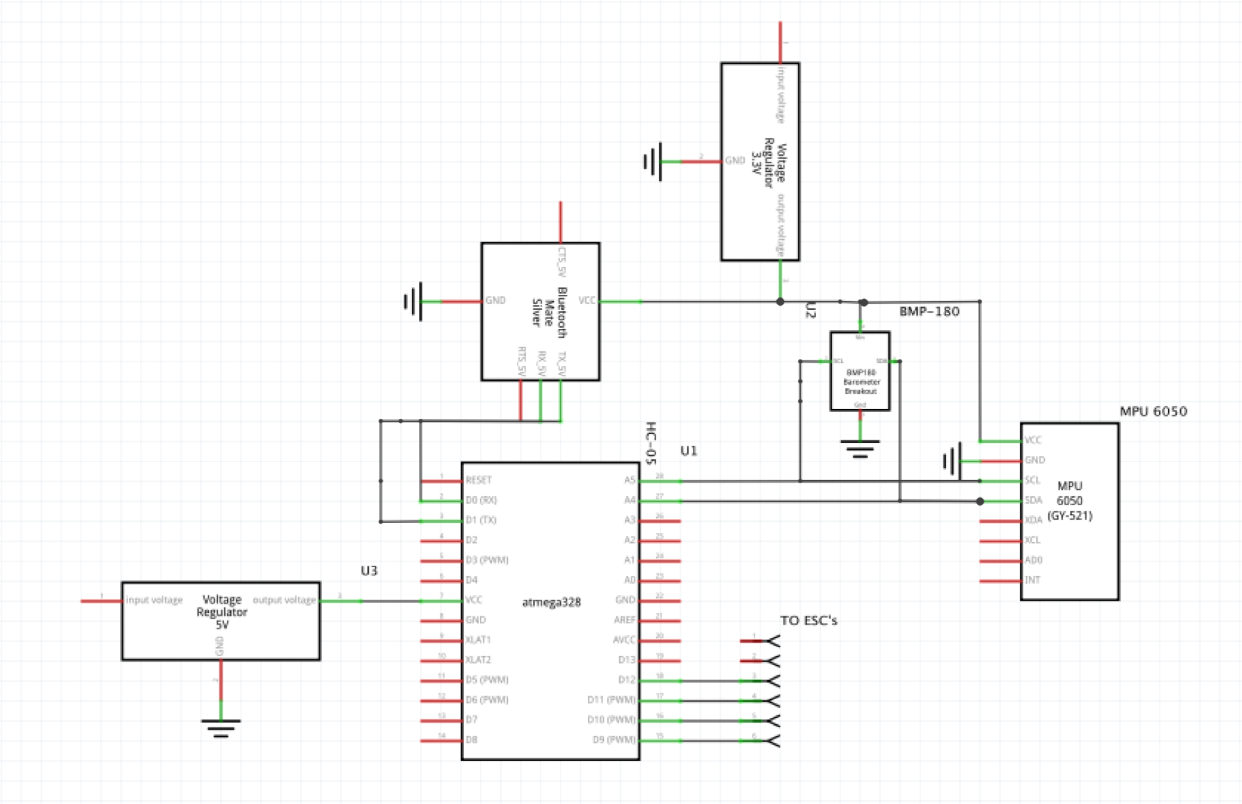
Clock Rate	16 MHz
RAM	2 KB
Flash	32 KB
Voltage Range	1.8-5.5 V
Price	$2
Manufacturer	Atmel

Flight Controller’s Program




Setup
motorFL:integer
motorFR:integer
motorRL:integer
motorRR:integer
receivedValue:integer
calibrateESC(): return Boolean
calibrateGyro()
connectionBT()
ESC Calibration
esc1: integer
esc2: integer
esc3: integer
esc4: integer
readyForFlight: boolean
esc1LowTest()
esc1HighTest()
...
esc4LowTest()
esc4HighTest()
Perform Action
actionValue:integer
thrustUpwards()
flyForwards()
flyLeft()
flyRight()
landDrone()
flyBackwords()
thrustDown()
Loop
stateOfMotors: boolean
valuePID: float
calcualtePID(): return float
flightAction(receivedValue)
alititudeCalculation()
Calculate PID
kp_roll:float
kp_pitch:float
kp_yaw:float
ki_roll:float
ki_pitch:float
ki_yaw:float
kd_roll:float
kd_pitch:float
kd_yaw:float
autolevel:integer
rollAdjustment:float
gyroConversions()
inclinationAngles()
errorCalculation()
angularVelocityAngles()
nosieCalculations()

Schematic Design



Administrative Content

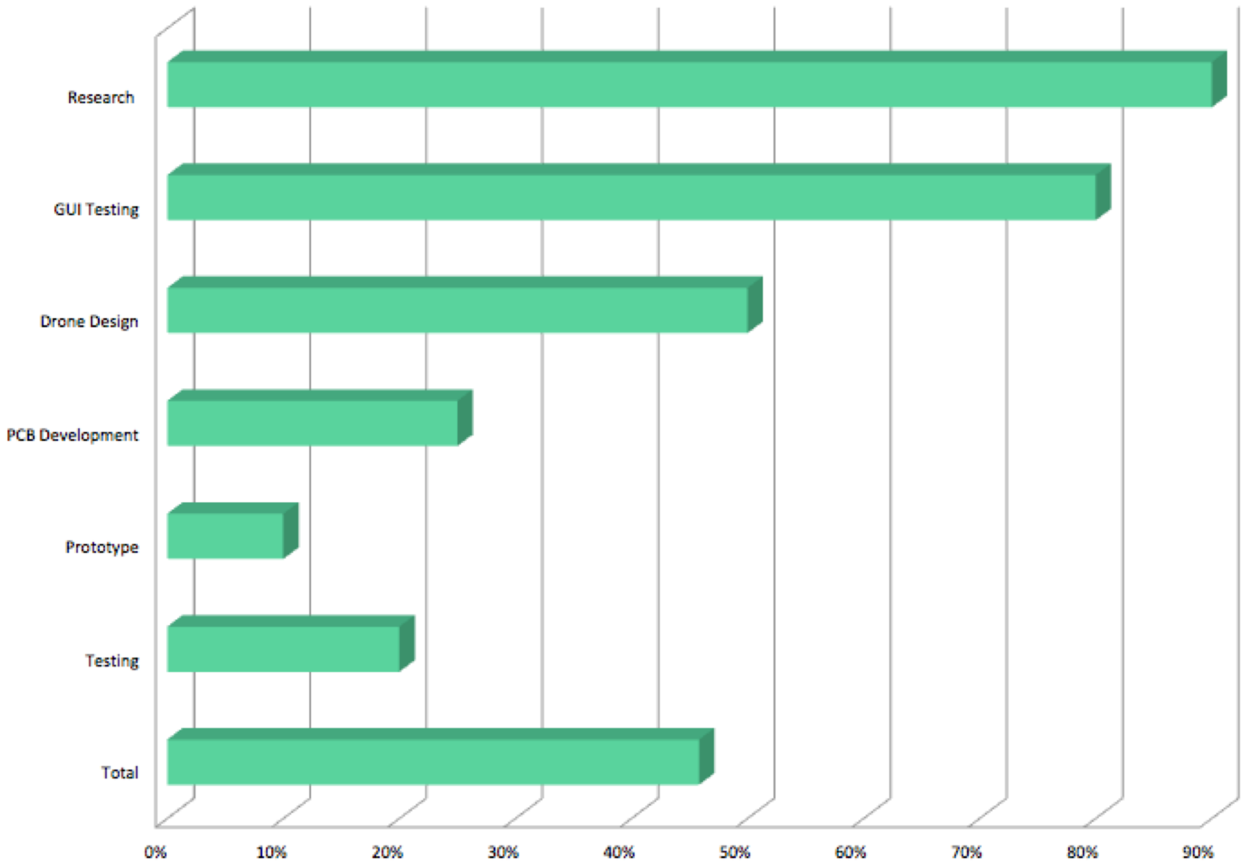
Proposed Budget

Component	Estimated Cost
Development Equipment	$100.00
Bluetooth Module	$10.00
Motors, ESCs, Propellers	$210.00
Drone Frame (150mm)	$20.00
Voltage Regulator	$5.00
Batteries	$40.00
Sensors	$40.00
PCB Printing	$30.00
Miscellaneous Components	$200
Total:	**$655**

Work Distribution

	User Interface	Drone Design	Power	Computer Vision	Wireless Comm.	Flight Controller
Anshul						
Pranay						
Bernardus						

Current Progress



Proposed Schedule

Date	Milestone
September 24, 2019	CDR Review
September 29, 2019	Finish building the prototype
October 4, 2019	Test Prototype
October 5, 2019	Use following time to redesign and rebuild
November 19, 2019	Finalize drone for final project
November 25, 2019	Final presentation

Future Improvements

- Our project can be expanded upon to be optimized for the market in many ways
- Range can be increased
- Response time can be faster
- Flight time can be increased

Questions?