# Safe Construction Unmanned Aerial Vehicle

Alan Hernandez, Baian Elmazry, Nicola DaSilva, Veronica Love

Dept. of Electrical and Computer Engineering

University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* – Every year, construction workers are severely injured while working at construction sites. Falls from high places are one of the main reasons for construction related injuries and death. Since drones are becoming more popular within our society, they could be the solution for beter construction worker safety. Safe Construction Unmanned Aerial Vehicle aims to solve some of the hazardous situations that occur on and around construction sites. To achieve this goal, this project uses a series of algorithms, a claw mechanism, computer vision, and implements various safety measures.**

***Index Terms* - April-tag, Construction, Delivery System, Machine Vision, Safety, UAV**

## I. INTRODUCTION

Safe Construction Unmanned Aerial Vehicle (SCUAV) seeks to revolutionize the construction industry using autonomous drones capable of constructing structures within metropolitan areas. Our project addresses the issues of safety and cost in the modern construction industry. This project implements the use of a drone capable of picking up pre-cut Styrofoam blocks no larger than the drone itself from a construction material site. The drone assembles these Styrofoam blocks to create a three by three by three-foot structure at the build site. Used in our project is the Pixhawk flight controller which, through various sensors such as a gyroscope and GPS, provide stable flight for our vehicle. A small robotic claw attached to the bottom of the drone works in conjunction with a rangefinder to grasp and place Styrofoam blocks.

To automate the process of construction, an on-board computer and camera running computer vision and Artificial Intelligence algorithms would be required on the drone to recognize the materials that are needed to be pick up and assemble a structure. The microcontroller would act as a slave to the on-board computer of the drone issuing commands to the flight controller during the construction process.

Communication to the drone would be done over Wi-Fi or SSH from the on-board computer to the base station. The drone's information, such as its position, its current action being performed, and battery life, will be monitored at the base station. There will also be a way for manual override to take control if for any reason an issue occurs during operation. If the connection between the drone and the base station fails, the drone will land and safely drop the object that it is currently holding. If this project is successful, it can change the way engineers build structures and help create more innovative uses for drones.

Fig. 1. Flight Assembled Architecture by Gramazio Kohler Architects, Raffaello D'Andrea, and ETH Zürich [1]

## II. VEHICLE COMPONENTS

The vehicle is comprised of components that are essential for the operation of this project. This section provides a brief introduction to the major components used to achieve the final product.

### *A. Flight Controller*

The Pixhawk flight controller is vehicle's main component and the heart of the drone. This essential controller sends signals for flight functionality. Included in this device is an Inertial Measurement Unit (IMU) sensor, a GPS module, and a barometer. The processor has a 32-bit ARM Cortex M4 core with FPU, 168 MHz/256 KB/2 MB Flash, and a 32-bit failsafe co-processor. [2] Some other features that are important for this device's functionality are the safety switch, the low power battery buzzer, and the USB connector. Sections III and V provide more details about this component.

### *B. On-board Computer*

The on-board computer, Raspberry Pi 3 Model B, is the brain of the entire system. All the sensory information is transmitted to the device and the computer determines all the instructions that need to be carried out. This device has a 64-bit ARMv8 Quad Core processor, runs at 1.2 GHz and has 1GB of RAM. [3] A Wi-Fi module is necessary for the functionality of the Raspberry Pi integration. The operating system is Raspbian. The Bluetooth module in the Raspberry Pi will also be used in communication between a mobile device and the drone in regards of arming, disarming, taking off and emergency landing.

## *C. Vision*

The vision aspect is an essential part of the vehicle's functionality. The OpenMV M7 camera is used as the eyes of the UAV. This camera uses machine vision instead of the traditional computer vision aspect. Machine vision slightly differs from computer vision in the sense that the objects to be viewed are already known and almost all observed events are predictable. The camera uses an STM32F765VI ARM Cortex M7 processor that runs at 216 MHz/ 512 KB of RAM/ 2 MB of Flash. The OV7725 image sensor enables 640x480 grayscale and RGB 565 high definition images and videos. [4] The Hardware and Software Interface sections shows how this camera is used in this project.

## *D. Microcontroller (PCB)*

The microcontroller unit is housed on the Printed Circuit Board (PCB). Connected to the board are the claw and the rangefinder. This controller used is the Atmega328 chip. This chip is connected to a 16-megahertz crystal oscillator. The ATmega328 will drive the claw and rangefinder. The data wire of the servo motor connects to a PWM pin on the ATmega328 and the rangefinder's trigger and echo pins connect to the analog and digital pins of the microcontroller. The claw and the rangefinder both operate at 5 volts and are powered from the PCB. To establish communication with the Raspberry-pi, the USB to serial converter board is connected to the TX and RX pins of the microcontroller. And the SPI pins are used to program the ATmega chip. The hardware connections between the Atmega328 chip and it supporting devices are seen in Figure 2.

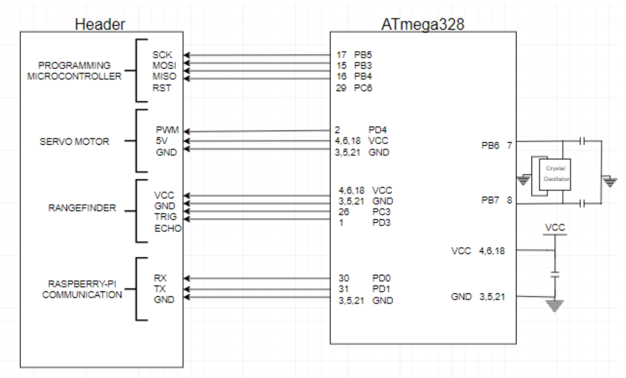

Fig. 2. PCB Overview

All these components are integrated together as well as the minor components such as the claw and the rangefinder. Figure 3 below briefly depicts how all the major components interact with each other. A more detailed explanation is described in the next sections below.

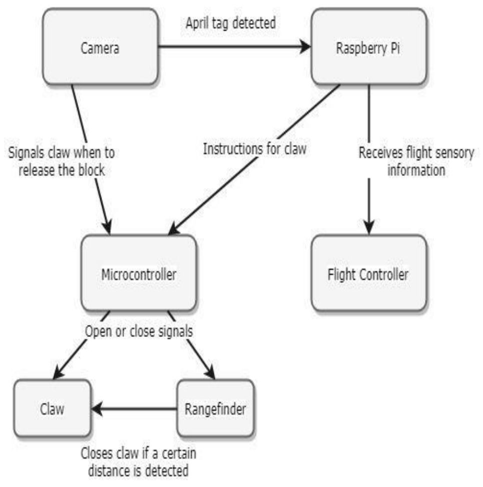

Fig. 3. System Interaction with Components

## III. HARDWARE INTERFACE

The selection of all the hardware components required much given thought for this project to be a success. From the size of the drone to the smallest device, all parts are essential. Section II, Vehicle Components, outlines the specification of each of the major components but Section III will dive into more details on why these components are important as well as minor details.

### *A. Minor Components*

The type of drone structure chosen was a hexacopter. This type of frame seems to be a powerful machine capable of performing the required mission. The electronic speed controllers (ESCs) and motors connected the flight controller are brushless. The motors are DC motors. These two devices are compatible with the controller and they work well when signals are received. The propellers are carbon fiber – meaning that they are light, durable, and seemed to be favored for powerful machines.

Lipo batteries are a safe option for this monstrous machine. They supply a voltage quality of 11V to the flight controller and all of the ESCs. The Raspberry Pi and the PCB will be powered via the RPi Powerpack V1.2. This light weight power supply fits snuggly onto the Pi and it is sufficient enough to power both devices. It lasts up to 9 hours and supplies a voltage of 5V.

The safety components that needed to be taken into consideration are the safety switch, the low battery buzzer, and the RC remote controller. The safety switch is crucial for arming and disarming the drone. It signals what stage the drone is currently in whether it be in the initializing stage, the enabling motors stage, or the drone is ready to be armed stage. The low battery buzzer signifies to the user when the battery on the drone is low. When the battery is low the drone will stop its current mission and land. Manual control is performed using the RC controller. The receiver is located on the drone and the remote is controlled by the user. Manual capabilities will come into play for if there is an emergency need to control the drone or if the user wishes to have SCUAV's mission be semi-autonomous.

The claw apparatus can be seen as both a major and minor component of this design. It is used to pick and release the Styrofoam blocks when commanded to do so. The ultrasonic rangefinder sensor correlates with the claw. It signals how close an object is to the drone. Both of these devices are configured on the microcontroller located on the PCB.

## *B. Major Components*

The Raspberry Pi is the mastermind of SCUAV. It is the command central for all of the autonomous or semi-autonomous functionalities. The Pi's configuration is described in greater detail in the Software Interface section. Hardware-wise, the Pi takes in all the sensory information from the microcontroller, the flight controller, and the camera. All the other secondary components are also transmitted to this device. Instead of using Universal Asynchronous Receiver-Transmitter (UART), the flight controller, the PCB, the camera, and RPi Powerpack will be plugged into the Pi via USB. USB connections seemed to be the most optimal solution for interfacing with each device. A USB hub may be needed for this hardware setup. An additional requirement for the Pi would be to have a portable router available. A local network will be initialized for accessing the Pi via Virtual Network Computing (VNC). The Pi will connect to it from a laptop, which will start the Pi's program. More about why VNC is crucial will be discussed in the next section.

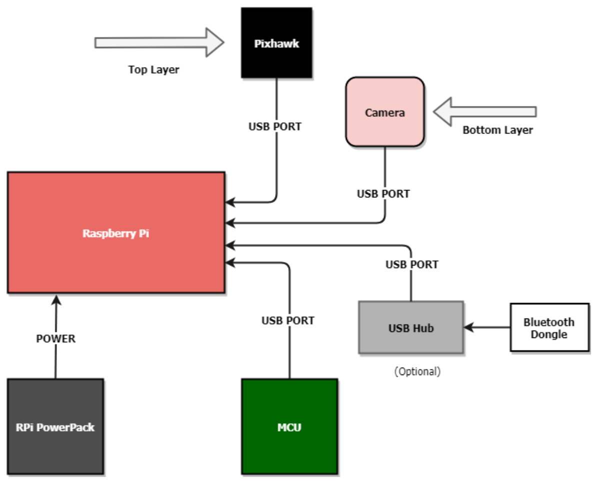

Fig. 4. Middle Frame: Raspberry Pi and PCB

The OpenMV Cam M7 camera will connect to the Raspberry Pi, as mentioned before, using one of the USB ports located on the Pi. Originally, the $I^2C$ communication was to be used for the camera to send data to the Pi. However, the OpenMV performs poorly as a slave because it would have to be in a system call to feed the hardware data for a master device to get anything other than zeros. [4] USB is a much faster and simpler protocol to use. The USB connection will also power the camera allowing it to perform its duties. The camera will be located at the bottom of the drone, facing downwards. It will be attached to the 3D printed frame. Its position is at the front of the drone. Figure 6 shows the location of the camera's position on the drone.

Pixhawk's flight controller device can be seen as the nerve and the muscle operator of the drone. When it receives its instructions from the Raspberry Pi, the controller sends signals to the motors via the ESCs on how to control the propellers. The IMU sensor in drone terminology comprises of a gyroscope, an accelerometer, and magnetometer. This sensor is embedded inside of the flight controller's schematic. The sensor keeps the drone stable during flight, signals the amount of speed needed to keep the drone in the air, report the rotational forces around the drone, and signify any changes that are needed to be made. The GPS module allows the flight controller to know the location of the drone. This information will allow SCUAV to navigate along its path. Interestingly, once the flight controller is powered on, the GPS locations help to signify when the drone is ready to fly since it will be flashing green.

The safety switch is wired directly onto the flight controller. It allows for the drone to be ARMED – ready for flight or DISARM – deactivate the drone. As the name implies, it guarantees when it is safe to use the drone. The RC receiver is also connected directly into the flight controller. A binding process needs to take place in order for the transmitter to properly communicate with the receiver. There is also a small power distribution board connected to the controller. The board evenly distributes power to all the devices plugged into the flight controller. Figure 5 below depicts how the top layer of the drone is laid out.

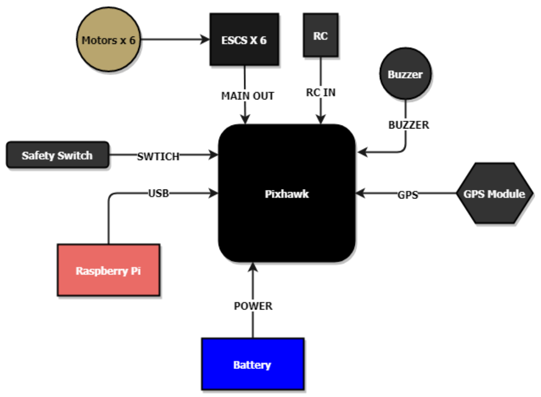

Fig. 5. Top Frame: Flight Controller Connection

The microcontroller drives all of the components connected to the PCB which can be seen Figure 2. The absolute maximum current that can be supplied from the ATmega328's IO pins is 200 milliamps [5]. The rangefinder draws 15 milliamps and the servo motor may draw between 90 to 190 milliamps. The voltage regulator on the Arduino Uno limits the current that can be drawn between VCC and ground. The current drawn from the IO pins of the ATmega328 PCB does not exceed 200 milliamps, and the power supply to the ATmega microcontroller from the battery will have enough current to support the current draw of the rangefinder together with the servomotor.

The UART communication between the Raspberry-pi and the microcontroller requires the use of a level shifter. The RX and TX lines of the Raspberry-pi and the ATmega328 operate at different voltages, the former operates at 3.3 volts and the later at 5 volts. Level shifting is performed through a FTDI chip external to the PCB. The FTDI chip converts USB from the Raspberry-Pi to UART operating at 5V, which can be directly connected to the RX and TX lines of the ATmega chip. The RX line from the FTDI connects to the TX GPIO of the microcontroller and the TX line from the FTDI connects to the RX GPIO of the microcontroller. Along with this, the grounds between the Raspberry-pi and the ATmega328 needs to be connected.

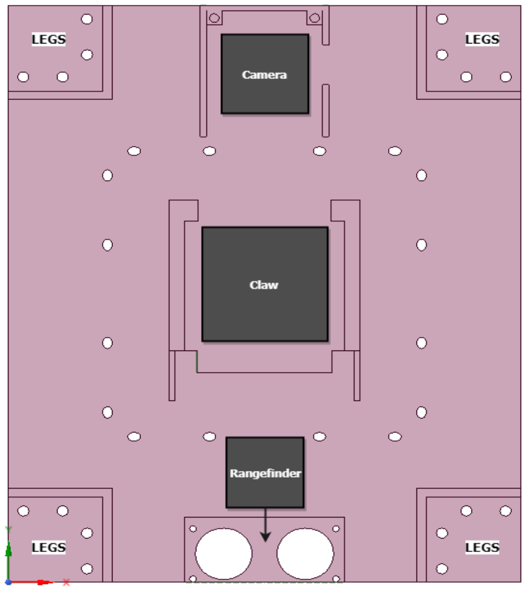

Fig. 6. Bottom Frame: 3D Print Frame

## IV. Software Interface

The software interface is an essential part of the drone's functionality. The programming language mostly used for this project is Python, but some of Arduino's language – which is C/C++, was used to program the microcontroller on the PCB. The section is split into primary and secondary programs. The autonomous functionalities and computer vision will use OpenMV and DroneKit libraries.

### *A. Primary Program*

The main program for this system begins with the Raspberry Pi. With the use of DroneKit API, a series of parameters are initialized when the program first begins. Some of these parameters include a set altitude, an adjustable speed, flight modes, and emergency steps.

A connection between the Pi and the flight controller is established once the battery is plugged in and the program is initiated. Also, a network must be established in order for the Raspberry-Pi to execute any commands. A big issue encountered during testing is UCF's IP address is private and does not allow users to access it, which makes wi-fi very difficult to obtain for the Raspberry-Pi. We can establish a network by enabling a mobile hotspot with our phone. Using GPS signals, the home location or base station position is saved as a programming requirement. Preflight checks are performed and reports a status back to the Raspberry Pi. If the status is positive, the drone is ready to be ARMED. If the status is negative, the issue is printed to the screen and requests the user to fix the highlighted issue. The drone is now ready for take-off once a positive status is issued. A Bluetooth connection also needs to be established for accessing any requests made from the phone via the Bluetooth Application installed.

A series of steps are now flowed to ensure a successful mission. These steps are mentioned below:

1) The Take-Off command is launched.
2) Reach target altitude and set the vehicle mode to Loiter.
3) Search for April-tag 1 using the camera's algorithm.
4) Once the Materials Site (April-tag 1) has been spotted, navigate towards it using GPS coordinates.
5) Find the first block tag required and head towards it by using the Euclidean distance of the block and drone's location.
6) When the drone is directly above the block, begin the claw and rangefinder sequence.
7) Once the block is grabbed, increase the altitude to the desirable flying height and search for April-tag 2 (Build Site).
8) Locate the designated tag spot for the block being currently held and position drone for the placing sequence.
9) After block is placed, fly back to Tag 1 and grab another block and place it at Tag 2. Continue this sequence until all the blocks are gone from the Materials Site.
10) When all of the blocks are accounted for, return back to the home base. This will be labeled with April-tag 3.

The motors are spun at the required speed to lift the drone off the floor once Step 1 is completed. In Step 2, the drone must reach its required altitude in order for the camera to search for the correct April-tag. An April-tag as shown in Figure 7 is a visual fiducial system similar to QR codes that can be used for a variety of functions including robotics, camera calibration and more.

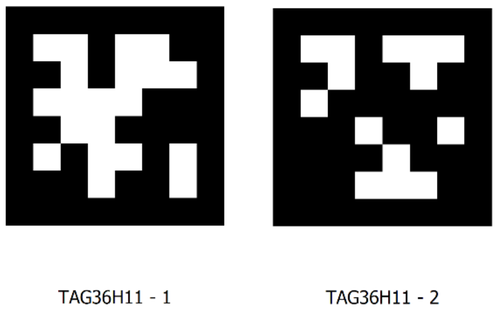

Fig. 7. Examples of April-tags [4]

April-tags are robust to lighting conditions and view angle, which makes it simpler and efficient to complete the building process. The April-tag should be located a few meters away from the home base. The camera should not be able to see the home base's April-tag, for it should be placed behind the drone's view. Each April-tag belongs to a tag family and encodes an ID number. With the ID number, the camera can differentiate which April-tag to be used in the building process. April-tag 1 is the Materials Site. Once the drone is at the Materials Site, it will hover until the camera spots the first block tag. When the correct block is found, the drone will navigate towards it by determining the Euclidean distance using the Pythagorean formula since GPS will lack accuracy.

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}. \quad (1)$$

The distance must reach a value of 0 and this will signify that the drone is directly above the block. Once it is centered, the drone will finally be able to descend to reach for the block. It is similar to using visual servoing.

The claw and rangefinder sequence consists of receiving instructions from the Pi. Figure 11 gives a quick overview of how this sequence works and it is described in detail in the Secondary Program section. Parametric conditions will be given and once they are met, it will signify that the drone has indeed collected a block within the claw's teeth. Now that the block is being held, the drone will increase its altitude again and search for the Build Site. Once the Build Site has been spotted the drone will navigate towards it but using the GPS coordinate system. At the Build Site, the drone will be set to the Loiter mode again and the camera will search for the correct location to place the block on its corresponding block position. Adjustments to the drone's flight will be made so that the dropping sequence can be initiated. When the block is released, the drone will navigate back to the Materials Site. There it will restart the picking up block sequence and fly back to the Build Site to perform the placing sequence. This will continue until every block is picked up and placed in the desirable location. Once all the blocks are placed, the drone will fly back to Home Base (April-tag 3). The mission is now complete. Figure 8 shows a layout of how the mission is to be performed.

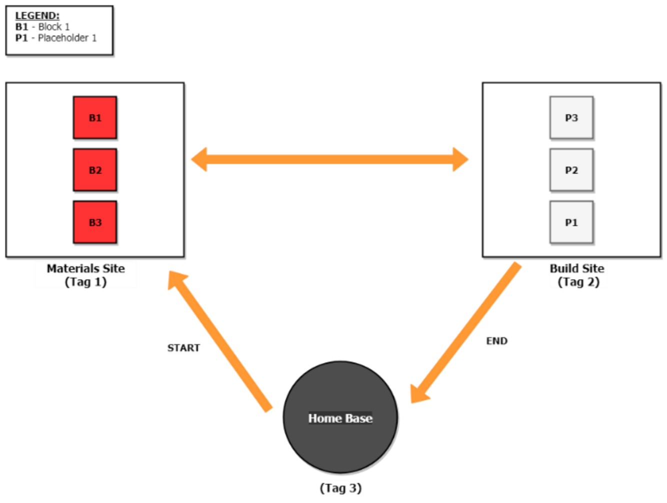

Fig. 8. SCUAV's Mission Flow

*B. Secondary Program*

The ATmega328 microcontroller code consists of commands for the claw and rangefinder operations. UART communication is also established with the microcontroller and the Raspberry-pi.

The rangefinder used in our project is the HC-SR04. This device emits ultrasound waves at 40 kHz. Ultrasound waves are those above 20 kilohertz and inaudible to humans. The rangefinder consists of a power, ground, echo, and trigger pin. The trigger pin emits ultrasonic sound wave in eight 40 kilohertz bursts. To initiate the bursts, the trigger pin must be held high for 10 microseconds. Once the waves are emitted, the timer begins counting. When the sound wave is interrupted by an object, it will bounce back and be received by the echo pin. The timer is then stopped, and the echo pin will be held high for the same amount of time that the sound wave traveled [6]. The timing for the rangefinder can be seen in Figure 9.

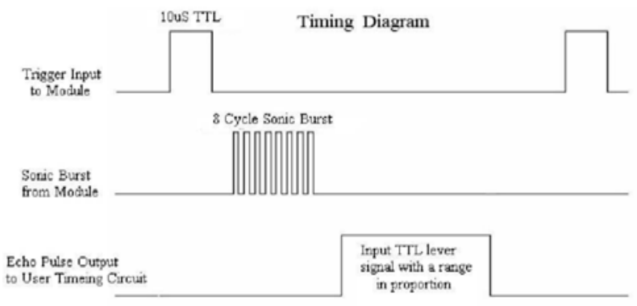
10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timeing Circuit
Input TTL lever signal with a range in proportion

Fig. 9. Rangefinder Timing Diagram [micropik]

Based on the timing for the rangefinder, correctly obtaining object distances involved the following commands:

```
digitalWrite (trigPin, HIGH);
digitalWrite (trigPin, LOW);
pulseIn (echoPin, HIGH);
```

Fig. 10. Rangefinder Commands

The distance an object is away from the rangefinder can be calculated from the total time a sound wave travels. This is the time from when the trigger pin emits the sounds waves to when they are received back by the echo pin. The distance away an object is can be calculated from the following equation:

$$d = (t * 0.034)/2 \tag{2}$$

Where $d$ is the distance away an object is away in centimeters and $t$ is the total time the sound wave traveled in microseconds. 343.5 meters per second is the speed of sound waves. The speed of sound converted to centimeters per microsecond is 0.034. The distance traveled by the sound wave is divided in half to obtain the distance to the object.

A standard servo motor drives the claw to open and close the claw. To initiate the commands for the claw, a pin for the servo data line. An angle variable is used to set the angle of the claw opening and closing. The val variable is used to set what command the Pi signals. The Pi signals when to open and close the claw. Close_claw function sets the angle to 100 degrees and writes this angle to the servo's data pin. Open_claw function sets the angle 15 degrees and write the angle to the servo when the command is issued.

The Raspberry-pi determines when to close and open the claw based on input information received from the camera and rangefinder. The camera will detect an April tag that indicates the drone is at the correct location to pick up a block. This is the first requirement for the Raspberry-pi to initiate the closing of the claw. The second requirement ensures the drone is the claw is correct vertical distance from the block. This information is obtained from the rangefinder. Figure 11 illustrates a flow chart of how the synchronization of the claw, rangefinder, and Raspberry Pi works.

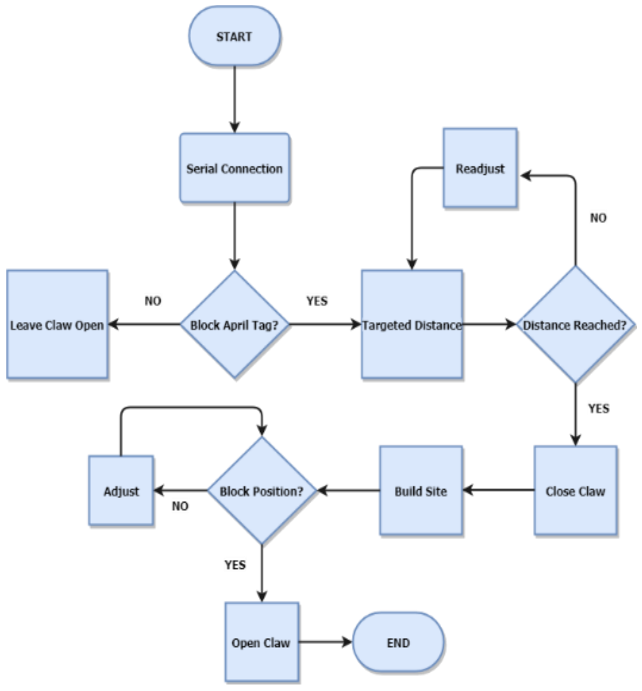

Fig. 11. Claw, Rangefinder, and Raspberry Pi Flowchart

The code is uploaded onto the microcontroller by using the Arduino Uno as an In-circuit Serial Programmer, or ISP. After configuring the Arduino Uno as an ISP, the code to be uploaded onto the PCB's ATmega328 chip is programmed onto the Arduino Uno. The program is then transferred from the Arduino Uno to the PCB by connecting the SPI pins and grounds. After the PCB is programmed and before it is connected to the drone's battery power supply, there is a break in power. The Atmega328 is a non-volatile device and will retain its program despite a break in power.

### C. Bluetooth Interface

Originally, users would be to control SCUAV's process through a web application designed using React framework in JavaScript. However, based on time constraints and the complexity of the project, it was simpler to use the built-in Bluetooth module from the Raspberry-Pi to communicate with a mobile device. The user will then be able to control the drone from his or her phone. Our team downloaded the Bluetooth Electronics applications from the Google Play Store. [8] The application allows users to connect their phone to another device and control it with any button you can choose from. A Python script is then created to connect our phone to the Raspberry-Pi. The entire program is halted until the user is connected to Bluetooth. Once connected, the Bluetooth dispatcher is executed and integrates the drone's functionalities depending on which button is pressed. The green button will arm the drone, the blue button will disarm the drone, the up-arrow button will take off, the down-arrow button will land the drone, and finally the red button will end the program and the disconnect from Bluetooth.

### D. Emergency Software Plan

Fly drones can be a very dangerous ordeal, especially in the surroundings in which this project will be demonstrated. Some of the cases to consider are low battery, claw did not grab block, and if the drone decides to malfunction. Safety is SCUAV's number one priority.

Case 1, Low Battery, should be consider because it will help to prevent unnecessary damage to the drone and limit the chance of a flying object situation. If the case were to occur, the drone will signal when the low battery percentage has been achieved. The user will be able to see this, and a beeping noise should be indicated. Once this signal is issued, the vehicle mode will be set to "RTL" - which stands for Return to Home and if a block is being currently held, it will be released immediately before the drone returns to Home Base.

Case 2, Missed Block, is considered because if drone misses the block, it will think that it is currently carrying a block and proceed to continue its path. This may cause a break in the algorithm and crash the drone in an unsafe way. To prevent this from happening, a flag will be set to see if the angle of the claw is reached for holding an object.

Case 3, Manual Override, is extremely crucial because as mentioned before safety is our number one priority. If the drone does not follow the algorithm and goes off the path, the user will be alerted immediately. The override flag will be issued, and drone will hover in its current position until the user has control of the drone using the remote controller.

## V. CONCLUSION

SCUAV has potential in revolutionizing the construction and drone industry, as well as enhancing safety procedures when working in hazardous areas. To achieve our project goal, challenges involving weight distribution, system communication, and compatibility were overcome. The Concerns involving the vehicles weight distribution arose because of the numerous components necessary for our drone. These issues were resolved to ensure steady vehicle takeoff and flight. The SCUAV project relies heavily on software and computer vision. The causes for some of the software communication issues involved various components sending data to one device and private networks. Our group also overcame the compatibility issues between the types of batteries, flight controllers and propellers used for the vehicle. The drone has been tested autonomously and manually in an outdoor environment.

The Senior Design course has been both challenging and enriching. The knowledge and skills gained in the classroom have equipped for achieving our project goals. Both technical and interpersonal abilities were developed during the course.

## ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance of Dr. Gita Sukthankar, Mr. Saif Ghassan, and Mr. Fred Cass.

## REFERENCES

[1] B. Hobson, "Building Architecture with Drones," Dezeen and MINI Frontiers, (03-Mar-2015). [Online]. Available: https://www.dezeen.com/2015/03/03/movie-drones-building-architecture-ammar-mirjan-gramazio-kohler/ [Accessed: 13-Apr-2018].

[2] ArduPilot Dev Team, "Pixhawk Overview," Pixhawk Overview - Copter documentation, (2016). [Online]. Available at: http://ardupilot.org/copter/docs/common-pixhawk-overview.html. [Accessed: 13-Apr-2018].

[3] Raspberry Pi Foundation, "Raspberry Pi 3 Model B," Raspberry Pi. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ . [Accessed: 13-Apr-2018].

[4] K. W. Agyeman and I. Abdalkader , "OpenMV Cam M7," OpenMV, (2017). [Online]. Available: https://openmv.io/products/openmv-cam-m7.[Accessed: 13-Apr-2018].

[5] Ww1.microchip.com. (2018). [online] Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf [Accessed 13 Apr. 2018].

[6] HowToMechatronics. *Ultrasonic Sensor HC-SR04 and Arduino Tutorial*. (2018). [online] Available at: https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/ [Accessed 13 Apr. 2018].

[7] Micropik.com. (2018). [online] Available at: http://www.micropik.com/PDF/HCSR04.pdf [Accessed 13 Apr. 2018].

[8] Keuwl.com. *Bluetooth Electronics*. (2018). [online] Available at: http://www.keuwl.com/apps/bluetoothelectronics/ [Accessed 13 Apr. 2018].

## ENGINEERS

**Alan Hernandez** is a 24-year-old graduating Computer Engineering student. He is responsible for the calibration of drone, manual flight testing, initial research into DroneKit, and assembling devices and other components to the project. Alan recently interned at Droplit.io. He will be pursuing a potential career in the Software Engineering field.

**Baian Elmazry** is a 22-year-old graduating Computer Engineering student. He is responsible for handling the computer vision integration of the drone as well as developing the drone's pathfinding algorithms. He recently interned at a small startup sofitU and will pursue a career in Robotics and Software Engineering. Baian eventually will pursue a graduate degree, focusing on Machine Learning.

**Nicola DaSilva** is a 23-year-old graduating Computer Engineering student. She was responsible for designing the 3D printed frame, some of the software integration, and hardware layout of drone components. She is pursuing a career in the Software Engineering field. Nicola also plans to obtain her Master's in the field of Robotics or Cyber Security, in the near future.

**Veronica Love** is a 21-year-old graduating Electrical Engineering student. She managed the distribution of power for the project and the design of the PCB. She has been a participant of the College Work Experience Program with Lockheed Martin since June 2016. After graduating she will be working for Direct Beam Incorporated. She will also be pursuing her Master's in Electrical Engineering with a focus in Electromagnetics and Optics.