Hey! This is from team 1001B-2021. We had a great season this year and we want to share and give insight on some of the more important things that we learned.

Keep in mind that this is NOT all of the information that you'll need to build your drone, just some things we think are important

Aubrey Vernon: Stay on top of the notebook because if you don't, you'll have to catch up on it the day before the tournament and that is just not convenient at all. Also when flying, don't have chairs set up near you, you may crash into them. In saying that, keep in mind that the drone is very touchy when you fly it.

Owen Vaneck: When flying the drone take into account that it will most likely vear to one direction more than it being centered, and always practice using only position and altitude mode. This is much better and easier to fly in than stabilized mode, it's very touchy and in order to be good at it all you need to do is practice.

Jacob Herrema: Sphero EDU is much better than Sphero Play for driving the RVR and the spheros. You can drive the RVR in reverse using the button in the bottom right corner of the app. Bring Dawg (the stuffed ladybug) on all the trips. P.S. Tegran is very ticklish.

Tegran Grigorian: The RVR is very stupid, it doesnt work, the microbit that runs its code neverworks. Everytime you try to download and run a program, restart the RVR, Microbit and the app controlling it. Once building micro bit code, MAKE SURE YOU PUT A SOUND SO YOU KNOW WHEN ITS RUNNING A FILE!!! The microbit will run code once it's activated; it is crucial you have an indication when it runs. Make sure the micro bit or any other rvr devices aren't touching carpet or anything that can get static(Insert svero breaking here). Next up is the RVR Frame, this year for earthquakes; we had an autonomous program, to obstacle avoid walls, we had these sensors to avoid walls by using the sensitivity. These sensors fell off the rvr on the national final; we lost because of that so my tip is to get the black, double-sided tape; and heat gun the adhesive into the sensor so it can't fall off. I have all of the code files and 3d printed objects on a flash drive and google drive just ask. Last tip is the most important, always read the gitbook.

Jack Snidancko: use correct motor screws. Solder correctly. Don't be scared. Solder is the answer to everything.

Tyler Peddie: **in Thrust we trust**

Logan Schwander: When making attachments for the drones, the more simple you can make it the better it will perform. Also make sure you DO NOT under any circumstances put an attachment directly under the propellers, it will affect the airflow affecting flight. Lastly, if you can

make sure to keep the center of mass as close to the middle as possible. Your pilot for the drone will thank you. Good luck :)

Chris Callahan

Notes for programming and general knowledge
Joshua Ma:
(616-634-3462)
I probably won't answer but leave a message and then I'll probably call back some time later

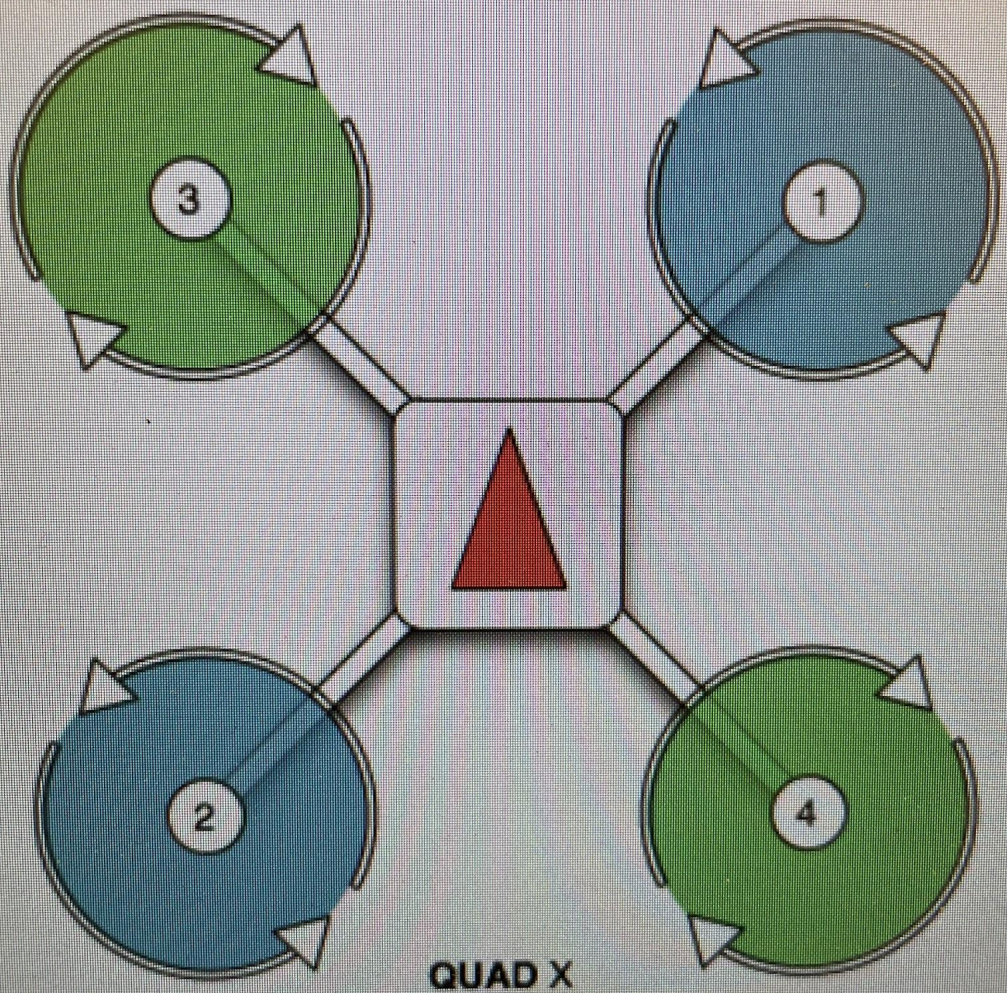
I will be going down the list of the 2021 drone gitbook with everything that I know about the sections that have to do with the computer/programming

Plugs for the Flight controller(FC):

DO THIS PART LATER

Connections for the motors(propellers)
Picture of the setup:

W direction.



QUAD X

Motors 1 & 2 = CCW rotation, Motors 3 & 4 = CW rotation

right-hand and left-hand threaded and marked underneath w
nts the diameter of the shaft in millimeters.

Going into the flight controller, the ports number 1-6 starting from the RIGHT, black wire goes up

We had some problems with the FC when trying to update/download firmware to it through Qgroundcontrol

If you can't update the firmware through qgroundcontrol, you might need to reset the firmware using the debugger

https://nxp.gitbook.io/hovergames/developerguide/program-software-using-debugger
Read up on this link to do this, although I doubt that you will need to.
Although most of the process from this link is fairly straightforward, there is one line
loadbin /complete/path/to/bootloader.bin 0x0
You don't copy this and paste in to J-link directly but I can't remember exactly what the bootloader.bin name is. Sorry. There is some file that you have to load and that has to go in the place of /complete/path/to/bootloader.bin something like

```
J-Link>loadbin C:\Temp\nxphlitev3_bl.bin 0x0
```

Don't forget airframe
For sensors and calibrating make sure to spin the same way that qgroundcontrol is telling you, we had better performance on one drone because of this

Later when you are able to SSH into the jetson and everything, you don't have to directly be connected to the FC which will be helpful for not having to have a wire be winded over your drone

If you are using the same computers as we did, only one computer can actually create and update the code as you can't build the code on the other. The one that does work has a protector for the camera that I didn't put there.

Serial means you have to be connected directly by wire to the jetson for it to be connected
Where as SSH you can be wirelessly connected
To SSH make sure you know what IP address the Jetson is connected to and then use that IP address in putty to login

```
1 nmcli radio wifi on
2 sudo nmcli dev wifi connect <wifi network name> password '<wifi network password>
```

This is needed to change the wifi the jetson is connected to.
Sometimes the jetson won't connect to a hotspot, it is very picky. You might need to run the command 'iwlist scan', this will update what the jetson can see as wifi connections, then you can connect to the hotspot(it's still very picky so it might not work)

As for the sandbox code(the autonomous code), we code in visual studio and in python

I am not the greatest programmer but I will try to make sense of the code for you.

For you to call function you need to have it in this format like on line 64

```
56          # whenever a message arrives on that topic.
57          self.topic_map: Dict[str, Callable[[dict], None]] = {
58              # This is what is known as a "f-string". This allows you to easily inject
59              # variables into a string without needing to combine lots of
60              # strings together. Scroll down farther to see what `self.show_velocity` is.
61              # https://realpython.com/python-f-strings/#f-strings-a-new-and-improved-way-to-format-
62              #f"{self.topic_prefix}/velocity": self.show_velocity,
63              #f"{self.topic_prefix}/#": self.aprilTagHeading,
64              f"{self.topic_prefix}/apriltags/raw": self.show_AprilTag,
65              #f"{self.topic_prefix}/apriltags/raw": self.show_aprilTags,
66              }
67
68
```

f”{self.topic_prefix}/(whatever topic you are in)”: self.(function name)
If you need to call multiple functions, add a comma at the end of the line

```
84      def on_connect(self, client: mqtt.Client, userdata: Any, rc: int, properties: mqtt.Properties
85          # Print the result code to the console for debugging purposes.
86          print(f"Connected with result code {str(rc)}")
87          # After the MQTT client has connected to the server, this line has the client
88          # connect to all topics that begin with our common prefix. The "#" character
89          # acts as a wildcard. If you only wanted to subscribe to certain topics,
90          # you would run this method multiple times with the exact topics you wanted
91          # each time, such as:
92          #client.subscribe(f"{self.topic_prefix}/velocity")
93          # client.subscribe(f"{self.topic_prefix}/location")
94          client.subscribe(f"{self.topic_prefix}/apriltags/raw")
95          self.open_servo(1)
96          self.close_servo(2)
97          self.open_servo(3)
```

This is where you “subscribe” to topics, how you can get the information that the camera gets when seeing an april tag, in our code I have only subscribed to apriltags/raw, here is the link to all of the topics and what information each topic has
https://github.com/bellflight/VRC-2021-Phase-II#mqtt-topics

The things after the client subscribes, are functions that I made for opening and closing servos. I wanted the drone to close all of it’s servos just in case the servos weren’t closed at the start.

```
144
145     def show_AprilTag(self, data: dict) -> None:
146         id = data[0]["id"]
147         posX = data[0]["pos"]["x"]
148         posY = data[0]["pos"]["y"]
149         posZ = data[0]["pos"]["z"]
150         #print(self.heading)
151
152         #GUI 2nd servo close = drop
153         if (id == 1 and posZ <= .65 and posX >= -.15 and posX <= -0.02 and posY >= -.076 and posY
154             self.close_servo(1)
155             print(posX)
156             print(posY)
157             print(posZ)
158         #GUI 3rd servo open = drop
159         if (id == 2 and posZ <= .5):#and posX >= -1 and posX <= 0 and posY >= 0 and posY <= 1):#ri
160             self.open_servo(2)
161
162         #GUI 4th servo close = drop
163         if (id == 3 and posZ <= .4):# and posX >= -.11 and posX <= .5 and posY >= -.02 and posY <=
164             self.close_servo(3)
165
166         if (id == 5):
167             self.LEDcolor("Red")
168         if (id == 6):
169             self.LEDcolor("Green")
170         if (id == 7):
171             self.LEDcolor("Blue")
```

This is the main definition of the code, when the camera sees an april tag it will run this and do its work accordingly. For example, when it sees april tag with id 1, it will “close” the 1st servo(this opens the servos because of the direction we had to set up the servo)
For the id 5 through 7, it runs another function to change the color of the LED

```
197     def open_servo(self, servoNumber: int) -> None:
198         # First, we construct a dictionary payload per the documentation.
199         data = {"servo": servoNumber, "action": "open"}
200         # This creates it all in one line, however, you could also create it in multiple
201         # lines as shown below.
202         # data = {}
203         # data["servo"] = 0
204         # data["action"] = "open"
205
206         # Now, we convert the dictionary to a JSON encoded string that we can publish.
207         payload = json.dumps(data)
208
209         # Finally, we publish the payload to the topic, once again using f-strings to
210         # re-use our common prefix.
211         self.mqtt_client.publish(topic=f"{self.topic_prefix}/pcc/set_servo_open_close", payload=payload)
```

For opening the servo, you need to get the data of what servo and what action, I created a parameter that allows you to pick the servo number(data on what the servo code is like is in the GUI/app.py) The payload is something I don’t know what it means but it is necessary. The last line 211, the only part you need to change is what is in the self.topic section. You need to change it in a similar way to when you were calling functions earlier.

```
        Set a servo state
        """
        self.publish_message(
            "vrc/pcc/set_servo_open_close", {"servo": number, "action": action}
        )
```

(This is in the app.py)
The previous code was copied directly from this because this is how the GUI changes the state of the servo, vrc is self.topic_prefix and then the rest is the same.

```
    def LEDcolor(self, color: str) -> None:
        print(f"AprilTag Number: {id}")
        Red = {"wrgb": [255, 255, 0, 0]}
        Green =  {"wrgb": [255, 0, 255, 0]}
        Blue = {"wrgb": [255, 0, 0, 255]}
        if color == "Red":
            data = Red
        if color == "Green":
            data = Green
        if color == "Blue":
            data = Blue

        payload = json.dumps(data)
        self.mqtt_client.publish(topic=f"{self.topic_prefix}/pcc/set_base_color", payload=payload)
```

This is for the LEDcolor changing, I have a couple variables set to the wRGB values to that of Red, Green, and Blue. This also has the same format when publishing to the PCC, which just means telling the pcc to do something. The only difference is that we are using a different topic which is pcc/set_base_color.

I hope this is all that you need but please ask someone if you need help on something, you don't get anywhere just sitting around so ask for help, and hopefully you don't have to rage too hard when building your drone.