To: Robert Severinghaus
From: Team 3 Wildfire Drone
Date: March 26th, 2021
Subject: Testing Results Report

This report outlines the testing phase of the engineering design process for our capstone project. System testing is essential to determine whether the product meets the requirements specifications of the project. Our design team conducted four tests. The first test is a unit test, matrix (UTM) which tests requirements 2.8 and 2.8.1. Requirement 2.8 states that the system should be able to function autonomously while requirement 2.8.1 further specifies that data transmission between the drone and base station should occur autonomously. The second test is a unit test, step by step (UTS) that determines whether requirement 2.9 is met. This requirement states that the system should integrate the fire classification, object detection and image segmentation algorithms produced by the Computer Science team on the drone subsystem. The third test is a second unit test, step by step (UTS) that tests requirement 2.7 determining whether the system includes a user friendly interface that interacts with the drone subsystem providing the user with sensor, GPS information as well as images of the fire from both the HD and thermal cameras. The final test is an integration test of requirement 2.6 and 2.6.1 through 2.6.4. These requirements state that the data must be transmitted from the drone microcomputer to a ground station computer. The data for transmission should include images, sensor data such as humidity, ambient temperature and GPS location. Data transmission will occur through single hop communication by means of Software Defined Radio (SDR) from the drone subsystem to the ground station and vice versa. Our design team invested approximately 12 hours to perform each of these tests. Most of our tests had to be simulated due to an incomplete integration of our entire system. Therefore, the results of the tests we conducted were of parts of the system rather than the system as a whole. Besides this fact, the tests that we could perform resulted in success through a visual representation of our expected results.

EE486C:Capstone Design
Team 3: Wildfire Drone
March 26th, 2021

Introduction to the System:

Client Background

Our project is sponsored by Dr Fatemeh Afghah who is an assistant professor at Northern Arizona University (NAU). She is leading this project along with her Ph.D. student, Alireza Shamshoara. Alireza Shamsoshoara is a Ph.D. student of Dr. Fatemeh Afghah at Northern Arizona University and he is overseeing the design of the project. He has a lot of experience with SDR communication and software. As a Senior IEEE member and the Director of Wireless Networking and Information Processing Laboratory, she will provide our design team with lab space and equipment necessary to complete the project.

The Problem Being Solved

With the rise in wildfires in recent years due to global warming, there is a demand for solutions to address this issue. Our design team intends to address this issue by creating a system that monitors wildfires by utilizing unmanned aerial vehicles (UAVs). This system aids in providing relief from the severity of the effects of wildfires. Our wildfire monitoring system provides live feed of the state of ongoing wildfires to give responders a plan of action.

An Overview of the Design

Our design team was given the opportunity to contribute to our client's ongoing effort to monitor wildfires using unmanned aircrafts. We were to design a wireless communication system that uses software defined radio (SDR) to transmit aerial data from a drone to a base station. The data from the SDR receiver will be processed at the ground station through an interface with a personal computer to display images of the fire in real time.

EE486C:Capstone Design
Team 3: Wildfire Drone
March 26th, 2021

System Architecture:

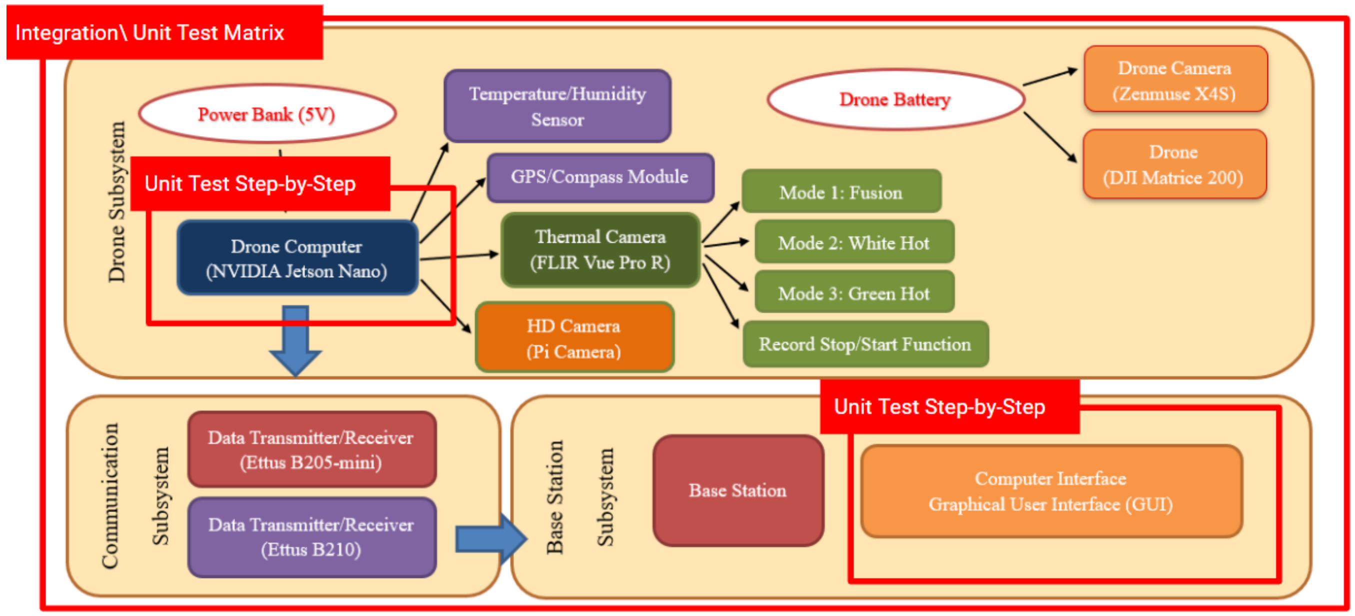

Integration\ Unit Test Matrix
Drone Subsystem
Power Bank (5V)
Unit Test Step-by-Step
Drone Computer (NVIDIA Jetson Nano)
Temperature/Humidity Sensor
GPS/Compass Module
Thermal Camera (FLIR Vue Pro R)
HD Camera (Pi Camera)
Mode 1: Fusion
Mode 2: White Hot
Mode 3: Green Hot
Record Stop/Start Function
Drone Battery
Drone Camera (Zenmuse X4S)
Drone (DJI Matrice 200)
Communication Subsystem
Data Transmitter/Receiver (Ettus B205-mini)
Data Transmitter/Receiver (Ettus B210)
Base Station Subsystem
Base Station
Unit Test Step-by-Step
Computer Interface Graphical User Interface (GUI)

Figure 1. Highlighted Tests on System Architecture

For our testing, we decided to conduct four types of tests on our system: one Unit Matrix Test (UTM), two Unit Step-by-Step Tests (UTS), and one Integration Test. In the figure above, it is outlined and defined where each test was conducted in our system architecture. Our matrix test and integration test involved utilizing the interaction of each subsystem with each other as a complete system. As for each unit step-by-step test, we chose to test one component of the drone subsystem and the base station subsystem. On the drone subsystem we wanted to test the drone computer, the Jetson Nano, and the CS algorithms integrated on the microcomputer. Meanwhile, on the base station we want to test the functions of the GUI our design team developed.

EE486C:Capstone Design
Team 3: Wildfire Drone
March 26th, 2021

Requirements, Status, Type of Test:

	Status	Req #	Requirement
		1	**Standards.**
	⬛	1.1	The frequency signal from the software defined radio (SDR) should be between 70MHz to 6GHz
			...
	⬛	1.3.2	Pin number: 4 Pins.(1 power+ 2, data+ 3, data- 4, power-)
		2	**Engineering Requirements.**
inspect	🟩	2.1	System must interface between the two cameras and Jetson nano microcomputer in order to capture aerial images and stream live videos
inspect	🟩	2.2	The two cameras must capture aerial images
	⬛	2.2.1	The drone camera: Pi camera
			...
	⬛	2.2.2.1	This camera resolution should be 720p
	N/A	2.3	Utilize the jetson nano to compress the captured images and video before transmission
inspect	🟩	2.4	System interface between ground computer and SDR receiver
	🟨	2.4.1	Processes the data from the SDR receiver and generates the images or video that will display to a graphical user interface (GUI).
inspect	🟥	2.5	System should utilize a Global Position System (GPS) on the drone microcomputer inorder to pinpoint the exact location of the fire
Integrate	🟨 *	2.6	The data from the drone microcomputer must be transmitted to a ground station computer
integrate	🟨	2.6.1	Data transmitted: images, sensor info, and GPS location
	⬛	2.6.2	Data transmission will occur through single-hop communication
			...
	⬛	2.6.4	Utilize the SDR Ettus B210 at the base station to receive data from the drone
UTS	🟨	2.7	System should include a user friendly interface to select between two viewing modes providing user with sensor and GPS information, or feed from either cameras
UTM	🟨 *	2.8	The system should be able to function autonomously
UTM	🟨 *	2.8.1	Data transmission between the drone and base station should occur autonomously
UTS	🟨 *	2.9	System must integrate the fire classification, object detection and image segmentation algorithms from the Computer Science team onto the drone microcomputer
inspect	N/A	2.1	Produce 3D prints on AutoCad of case mountings to house devices on the drone while being mindful of aerodynamics (optional)
		3	**Constraints.**
inspect	🟩	3.1	Utilize the SDR Ettus B205-mini for drone communication
			...
inspect	🟩	3.4.3	Thermal Camera: 4.8V - 6.0V, 2W

EE486C:Capstone Design
Team 3: Wildfire Drone
March 26th, 2021

Most Important Requirements:

The requirements that are most important in our project are requirements 2.6, 2.8, 2.8.1 and 2.9. Requirement 2.6 states that data transmission must occur from the drone subsystem through the microcomputer to the ground station computer. The data that should be transmitted are images, sensor information and GPS location. Data transmission should be driven by Software Defined Radio (SDR). In effort to monitor wildfires, our client is interested in obtaining information about the environmental conditions of the area during the fire, the location and relevant images of the state of the fire. The sensor data will provide information about the current ambient conditions at the state of the fire such as the temperature, humidity and air pressure. Collectively these data provide the user with a complete understanding of the nature and severity of the fire. Retrieving accurate data and information about the state of a wildfire during the early stages of the fire is essential for removing the fuel from the area and stopping the expansion of the fire to prevent further harm or damage. Failure to meet this requirement impedes the user from obtaining a complete understanding of the nature of the fire.

Requirement 2.8 states that the system should be able to function autonomously. In addition, requirement 2.8.1 states that data transmission between the drone subsystem and base station system should occur autonomously. The automatic functionality of our system is essential to the overall purpose of the product. Our client plans to utilize UAVs as a safe and inexpensive method to capture aerial images and provide real time data feed from the location of the fire. This is realized through automation. If this requirement is not met then the system would not be able to operate as specified by our client.

Requirement 2.9 states that our system must integrate the fire classification, object detection and image segmentation algorithms of the Computer Science team. Fire classification identifies the images with fire that is detected through the camera feed and ensures that this image is sent to the user for inspection. Object detection identifies objects such as people, animals, cell phones etc that are within the vicinity of the fire. These algorithms filter the camera feed and provide the user with relevant information about the state of the fire. This is essential to our client's effort to monitor wildfires. Without successful completion of this requirement the user would be unable to receive only the relevant information about the state of an active fire.

Types of Tests:

Descriptions

Unit Test Matrix: Cataloging a list of repeatable tests on one axis—for example, different tasks to accomplish, listed in the first column; and a list of items to be tested on the other axis—operating systems on which the tasks are to be done, listed on the top row of the table.In our system this was performed to check the cameras (HD and thermal) turned on and capture images automatically and thermal cameras can be operated as different modes (mode 1: fusion, mode 2: white hot, mode 3: green hot). Meanwhile, GPS, compass, temperature module and humidity sensors can take readings automatically.

Unit Test Step-By-Step: Showing or explaining each stage in a process; happening or done in a series of steps or stages. It is like our whole capstone simulation, we only use GUI to check if the drone can receive command and send the correct file to the base station , then showing on the GUI.

Integration Test: The phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before validation testing. Our EE team and CS team are both working on this capstone project. They mostly get charged for the software part, our EE takes the hardware part. This test is to make sure there is no mistake happening when we combine the hardware and software into a whole system. The system integrates the drone, communication, and base station subsystems.

Inspection: An observation of the design requirement is noted to determine whether it has been met. With our system, there were many requirements where we only had to observe whether or not it appeared in the system such as using certain devices like a specified camera or device.

Major Tests:

For our system, we decided to perform several tests and of those tests, the most important ones we conducted involved components of the system and the system as a whole. These tests were necessary to ensure the operations of the system would be simple and would outline the changes to our project that would be needed.

On the drone subsystem of our project, we decided to perform a unit step-by-step test on the operations of the drone. In this test, the CS team had integrated their algorithms to our system and we wanted to ensure the functionality of those algorithms on the hardware setup. Therefore in this test, we had a process in which we could switch between the different tasks of the drone that were dependent on the CS algorithms such as object detection, segmentation, and classification. If we were able to initialize the programs by changing a json file and the drone began operations, then we considered the test to be successful.

Another major test we conducted was at the base station, where we wanted to conduct the practicality and operation of the Graphical User Interface. For this test, we used the unit step-by-step, this way we could iterate through the buttons of the GUI to see if it had produced the output we expected for a user. On the GUI there exist several buttons that would determine the function of the drone as well as what is to be displayed onto the base station. The process of this test required the base station of our system to be set up as well as the GUI running on the base station as well. Additionally to see if the output of the buttons on the GUI, we opened the json file that would be manipulated and later transmitted. To conduct the test we would click each button of the GUI and see if there was a change to the json file and if there was a change to the file we would pass the test with each button.

We had also performed a unit matrix test on the entirety of our system, which was split up into sub tests this way we could divide certain operations to be tested. For this test we wanted to see the autonomous operation of the system. In this test, we tested the cameras and sensors of the drone computer, as expressed in Appendix C and Appendix D, as well as the communication between the drone subsystem and the base station. For the cameras, both the HD Camera and thermal camera, we wanted to see that when their task was run that we would receive an image captured by either camera that would be ready for transmission. If when the json file was changed to capture an image from either camera and the drone would perform the operation and the image was present in a designated file, we would consider the test to be a pass. For the sensors on the drone, we want to see that the sensors would begin reading and writing to a json file when the program of the drone runs. We determine the success of this portion of the test if the values of the json file display measurements and continuously change those values as the program runs. The last section of this test involved the communication of the drone and base station. In this section, we would transmit text from the base station to the drone and then we would transmit text or images from the drone to the base station. To conduct this test, we produced a text file on the drone system to be sent to the base station, then we used an image to be sent to the base station and finally we produced a text file at the base station to be sent to the drone. The test was successful if the same text or image was received at the designated receiver.

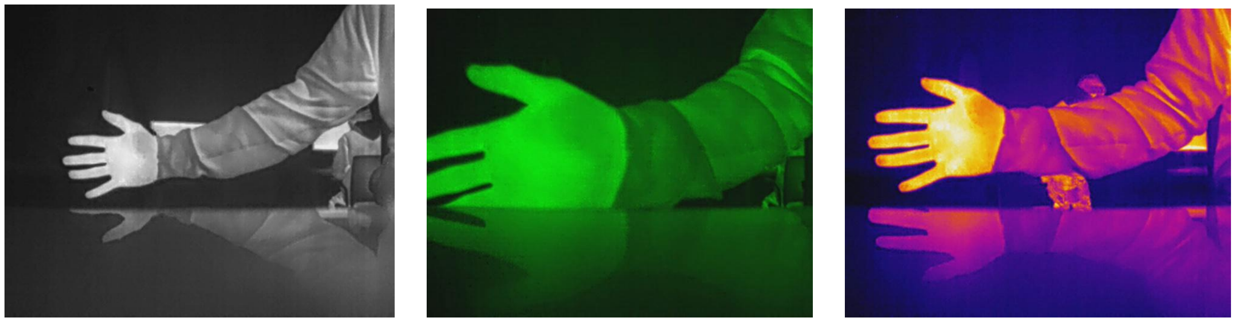
Analysis of Results:

Unit Test, Step by Step (UTS)

The first unit test, step by step (UTS) examines whether our system integrates the fire classification, object detection and image segmentation algorithms of the Computer Science team. Our design team was unable to integrate the image segmentation algorithm since the computer science team is currently working on completing it. Hence, this unit test only encompasses the fire classification and object detection algorithms. Under this test condition, the project was successful. These algorithms were executed upon user request. When fire classification was requested, the fire classification model began to execute enabling the camera feed which identified whether a fire was in view or not and indicated the percent accuracy of that prediction. In addition, the request of object detection by the user resulted in the execution of the object detection model and enabled the camera feed. The camera feed identified the objects in view with bounding boxes and a descriptive label of that object as well as the percent accuracy of that prediction. The integration of the fire classification model was successful 80% of the time while the object detection algorithm was successful 100% of the time. The result was unexpected for the fire classification model. However, this observation was clarified by the computer Science team who expressed that the client has requested that these algorithms be executed after detecting a fire. Hence, the camera feed would not turn on but these algorithms would be expressed in the captured image. Currently, the Computer Science team is modifying their algorithm to incorporate this change. Hence, there still remains bugs and errors that they need to fix.

Unit Test Matrix (UTM)

The unit test matrix (UTM) examines whether the system is able to function autonomously. This tests requirement 2.8. It consists of four parts. The first part of the test examines whether the two cameras are able to turn on and capture images automatically. Our client has requested that we utilize thermal and HD cameras to capture aerial images. The test was successful under this test condition. This was an expected result. The second part of the test examines whether the thermal camera is able to switch between three viewing modes. For the modes, we wanted to see the thermal camera receive commands from the Jetson Nano to switch between the color palettes of whitehot, greenhot and fusion. In order to conduct this test we would manually change the file that would manipulate the tasks of the drone. For this test, we expected to see the result of an image with the specified color palette in a path directory specified in our code. As we iterated through each color, we received the expected image in the path directory for thermal images on the drone, which outputted the images in *Figure 2.* Therefore, the results of this test were as expected.



a. White Hot b. Green Hot c. Fusion

Figure 2. Thermal Images Output from Test

The fourth test was to determine if text can be transmitted and received from the base station to the drone subsystem and vice versa automatically. In addition, it examined whether images can be transmitted from the drone subsystem and received at the base station. Since our client only lets us use one device so we have to only send text or pictures for the drone to the base station (can only use one channel to transmit), we generate a command file to determine which type of file we are going to receive on the base station. Under this test condition, automatic transmission of text was successful 100% of the time when transmitting from the base station to the drone. Our initial test trial failed because of difficulties in setting up the hardware for optimal transmission. Automatic transmission of images was successful 100% of the time for images produced from the thermal camera. However, the images captured through the HD camera were often distorted and resulted in data loss where only a fraction of the image is sent. The difference between these two images is that the thermal camera images are small within 100 KB or lower while the HD camera images tend to be larger than 200 KB. Our design has limitations in the size of the image that it can transmit. It exhibits distortions and loss of data when the images are above 200 KB. These results were expected since we understood and were working towards improving the design of our transmission process. However, the raspberry Pi camera was supposed to be used in our design but since this camera was damaged we could not manipulate the size of the images that were produced. Our design team plans to resize the image captured by the raspberry Pi camera in order to ensure optimal transmission through the SDR and eliminate the issue of data distortion and loss in the process.

Unit Test, Step-by-Step (UTS)

Another test was conducted on the operations and functions of the GUI at the base station. At the time we were testing the GUI the interface was not fully complete so we could not test every function we had intended. Therefore, we could only test the available functions on the GUI, in this case it was the classification, object detection, and thermal camera with a white hot view. Additionally because our system was not quite complete with each subsystem fully integrated with one another we had to simulate most of the operations at the base station. With this test we determined that the GUI was fully functional and produced the results that we expected to an extent. For instance, when we clicked on classification, it manipulated the json file that was then sent to the drone subsystem and then the drone began the classification protocol, but we are unable to send the classification image from the drone to the base station meaning we are unable to display it to the GUI. This is the case for the other buttons of the GUI as well. Therefore this test we considered as only being partially passed.

Integration Test

The integration testing was successful for the majority in task testing and sensor information. The success was shown when the base station would create a task file and that file would be sent to the drone in order to perform the task. We were able to determine the operations of the drone through the terminal output on the drone which would print the current task given. We were able to witness this result with each of the tasks we defined. Our system was not fully integrated therefore we had issues operating the system as a whole which could not be tested. Additionally in this test we also tested whether the sensor information on the drone would be sent to the base station and displayed on the GUI. However, when we transmit the sensor information from the drone to the base station, there is a 37% chance that the file sent could be altered, this was discovered after testing this transmission 50 times.

As an integration GUI and base station, we mostly finish all the tasks. Currently, we can power up our GUI on the GUI and show the data or picture base on a specified folder. We are able to send the command by text file from the base station to the drone. The command does not change after transmission. However, since we have not integrated all the work with the CS team. We are unable to tell the drone to correctly execute all the tasks files such as segmentation and gather GPS data.

Our testing process revealed that the most testing requirements were indeed met with the exception of the GPS sensor module not being implemented.

Lessons Learned:

The testing process was difficult since the HD camera code and sensor code was not integrated into the system. This is a task that the Computer Science team had to perform since only one person is familiar with the structure of this particular python script. Due to the timing sensitivity of our testing paper, our design team was not able to wait until the computer science team had integrated our code. As a result our design team conducted simulations to facilitate specific functions in each test. For example to test the operation of the HD camera our design team executed the python script which enabled the HD camera and captured an image. This script will be integrated into the overall system which enables the HD camera to turn on upon user request, capture an image and transmit the image to the base station using the SDR. Without complete integration of certain components of our system such as the sensors and the HD camera, we were not able to completely test the operations of the system to the extent that we had intended initially.

During testing our design team was able to modify part two of the unit test matrix (UTM) which switches between the three modes of the thermal camera. Our design team removed a delay in execution of the camera feed thereby enabling improving performance and enabling the user to access the various thermal camera modes at a faster rate. When testing the switching of the modes on the thermal camera, we did not have any difficulties or failures. However, we did run into some problems with other tests where we made adjustments as we continued, to applicable parts of the system. With many of our tests, we were able to discover ways that would improve not only the functionality of the system but improve the performance as well. For instance, as mentioned before we discovered the limitations to the size for which we are able to transfer images without outputting some distortion on the receiving end.

In addition, during the first UTM test to determine whether the system functions autonomously, our team simulated the automatic transmission of text and images through the SDR. This was done to meet the deadline of this testing report. The python script for the software defined radio in both the drone and base station subsystems [Appendix A and B] is currently pending to be integrated by the Computer Science team into the entire system. Automatic transmission is driven by a command text file that must be empty to enable transmission of text files and it must be written to to enable image transmission. Our design team uses two different ways to transmit text and picture, we try to open two different channels to receive each type of file separately with only one pair of devices. However, due to our hardware limitation, we can not set two different frequencies for two channels. Therefore, we only set up one common frequency for both channels in order to receive both types of files.

For the improvement of our overall system, there are aspects in our design that could be approached to improve the performance of our system. In the communication aspect of our system, we could improve the design by modifying the gain values which would increase the transmission power of the system. Additionally, with larger antennas we would be able to receive weaker signals. We did not perform a regression testing.

APPENDIXES

Appendix A. Base Station Software Defined Radio (SDR) Python Script

We have several functions for SDR transmissions for the base station, they are named as tb1, tb2. tb1 is receiving function and tb2 is transmitting function, Transmission function will be triggered by the size which file will be transmitted. Once it is not zero byte, the file will be sent to Drone. And our code is constantly receiving the data from the drone.

```
{ def main(top_block_cls_1=Basestation_3,top_block_cls_2=Basestation_1,options=None):
    open('/home/ziming/Documents/Happy_1.txt','a').close()
    open('/home/ziming/Documents/Happy.txt','a').close()
    open('/home/ziming/Documents/Happy.jpg','a').close()
    qapp = Qt.QApplication(sys.argv)
    tb1 = top_block_cls_1()
    tb1.start()
    Jpg_count_number = 1
    Txt_count_number = 1
    filesize = 0
    filesize_1 = 0
    filesize_2 = 0
    while 1:
       Deter_mode = os.path.getsize('/home/ziming/Documents/mod.txt')
       filesize = os.path.getsize('/home/ziming/Documents/Happy_1.txt')
       if filesize != 0:
              tb2 = top_block_cls_2()
              tb2.start()
              time.sleep(1)
              tb2.stop()


              os.remove('/home/ziming/Documents/Happy_1.txt')
              open('/home/ziming/Documents/Happy_1.txt','a').close()
              print('commond sent')
       filesize_1 = os.path.getsize('/home/ziming/Documents/Happy.txt')
       if filesize_1 != 0:
              if Deter_mode == 0:
                     if Txt_count_number > 1 :
                            os.remove('/home/ziming/Documents/Overwirte.txt')
                     open('/home/ziming/Documents/Overwirte.txt','a').close()
                     textFile = '/home/ziming/Documents/TxT_Receiver_%d.txt' %
Txt_count_number
                     shutil.move('/home/ziming/Documents/Happy.txt', textFile)
                     time.sleep(1)
                     with open('/home/ziming/Documents/TxT_Receiver_%d.txt' %
Txt_count_number) as firstfile, open('/home/ziming/Documents/Overwirte.txt','w') as
secondfile:
                            for line in firstfile:
                                   secondfile.write(line)
                            secondfile.close()
                     Txt_count_number += 1
```

```
                    open('/home/ziming/Documents/Happy.txt','a').close()
                    print ('get txt')
                    tb1.stop()
                    tb1 = top_block_cls_1()
                    tb1.start()
             else :
                    textFile = '/home/ziming/Documents/JPG_Receiver_%d.jpg' %
Jpg_count_number
                    shutil.move('/home/ziming/Documents/Happy.txt', textFile)
                    Jpg_count_number += 1
                    open('/home/ziming/Documents/Happy.txt','a').close()
                    print ('get jpg')
                    time.sleep(10)
                    tb1.stop()
                    tb1 = top_block_cls_1()
                    tb1.start()
    tb1.stop()
 }
```

Appendix B. Drone Software Defined Radio (SDR) Python Script

We have several functions for SDR transmissions for the base station, they are named as tb1, tb2, tb3. tb1 is receiving function and tb2, tb3 are transmitting functions, Transmission function will be triggered by the size which file will be transmitted. tb2 is for test transmission and tb3 is for picture transmission. Once it is not zero byte, the file will be sent to Drone. And our code is constantly receiving the data from the Base station.

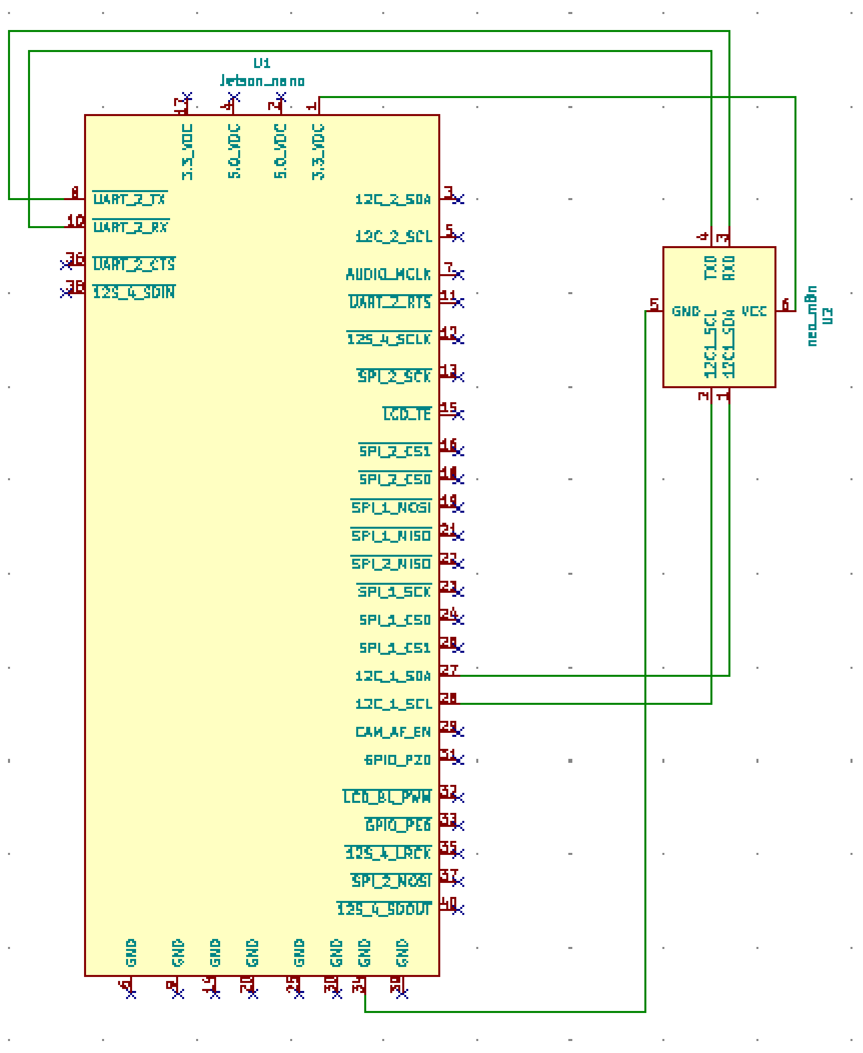
```
{ def main(top_block_cls_1=Drone_1 , top_block_cls_2=Drone_2 , top_block_cls_4=Drone_4 ,
options=None):
    open('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_rec.txt','a').close()
    open('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_sent.txt','a').close()

open('/home/spring2021/Desktop/Capstone/fire_scout_system/drone_station/test_images/ThermalFr
ame_0.jpg','a').close()
    tb1 = top_block_cls_1()
    tb1.Start()
    count_number = 1
    filesize = 0
    filesize_1 = 0
    filesize_2 = 0
    while 1:
        time.sleep(1)
        command = os.path.getsize('/home/spring2021/Desktop/Gnu radio
code/backup2/Command.txt')
        filesize = os.path.getsize('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_sent.txt')
        if filesize != 0:   ## have no blank txt
            if command == 0: ## send the txt
                tb2 = top_block_cls_2()
                tb2.start()
                time.sleep(1)
                tb2.stop()
                os.remove('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_sent.txt')
                open('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_sent.txt','a').close()
                print('  :  txt sent')
        filesize_1 =
os.path.getsize('/home/spring2021/Desktop/Capstone/fire_scout_system/drone_station/test_image
s/ThermalFrame_0.jpg')
        if filesize_1 !=0 :   ## have no blank jpg
            if command != 0: ## send the jpg
                tb3 = top_block_cls_4()
                tb3.start()
                time.sleep(10)
                tb3.stop()
```

```
os.remove('/home/spring2021/Desktop/Capstone/fire_scout_system/drone_station/test_images/Ther
malFrame_0.jpg')

open('/home/spring2021/Desktop/Capstone/fire_scout_system/drone_station/test_images/ThermalFr
ame_0.jpg','a').close()
                print('  :  pic sent')

        filesize_2 = os.path.getsize('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_rec.txt')
        if filesize_2 != 0:
            if count_number > 1 :
                os.remove('/home/spring2021/Desktop/Gnu radio
code/backup2/Receive_Overwrite.txt')
            open('/home/spring2021/Desktop/Gnu radio
code/backup2/Receive_Overwrite.txt','a').close()
            textFile = '/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Received_all/Receiver_%s.txt' % count_number
            shutil.move('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_rec.txt', textFile)
            open('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Txt_rec.txt','a').close()
            time.sleep(1)
            with open('/home/spring2021/Desktop/Gnu radio
code/backup2/File_recevie_send/Received_all/Receiver_%s.txt' % count_number,'r') as
firstfile,
open('/home/spring2021/Desktop/Capstone/fire_scout_system/drone_station/drone_ops.json','w')
as secondfile:
                # read content from first file
                    for line in firstfile:

                        in_json = False

                        for char in line:

                            # The SDR writes extra chars
                            if char == "{":
                                in_json = True

                            if char == "}":
                                secondfile.write(char)
                                in_json = False

                            if in_json:
                                # write content to second file
                                secondfile.write(char)

            print ('  :  get txt')
            tb1.stop()
            tb1 = top_block_cls_1()
            tb1.start()
            count_number += 1
```

```
   tb1.stop()
 }
```

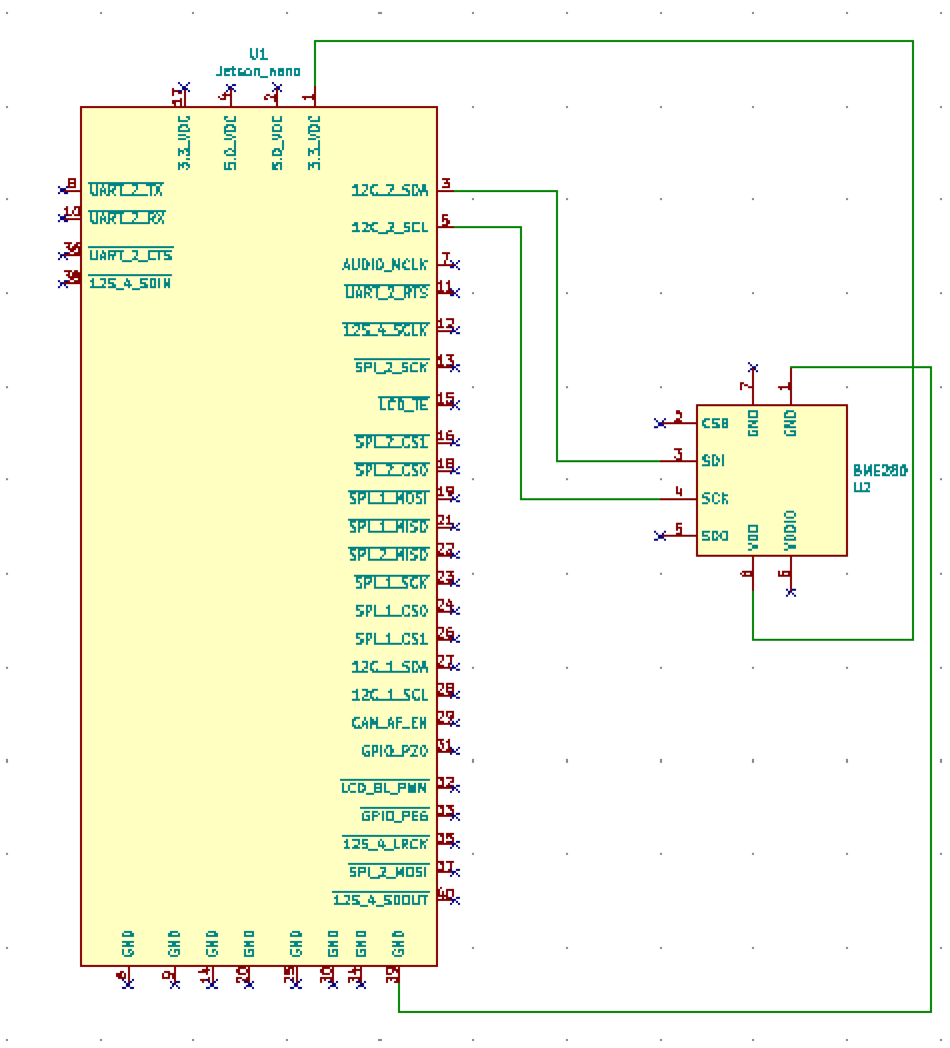
Appendix C. Global Position System (GPS) Design

The schematic of the implementation of the GPS and compass module (NEO M8N BDS) interfaced with the Nvidia Jetson Nano



Appendix D. Temperature and Humidity Sensor Design

The schematic of the implementation of the BM280 temperature and humidity sensor interfaced with the Nvidia Jetson Nano


U1
Jetson_nano
3.3_VDC
5.0_VDC
5.0_VDC
3.3_VDC
UART_2_TX
UART_2_RX
UART_2_CTS
I2S_4_SDIN
I2C_2_SDA
I2C_2_SCL
AUDIO_MCLK
UART_2_RTS
I2S_4_SCLK
SPI_2_SCK
LCD_TE
SPI_2_CS1
SPI_2_CS0
SPI_1_MOSI
SPI_1_MISO
SPI_2_MISO
SPI_1_SCK
SPI_1_CS0
SPI_1_CS1
I2C_1_SDA
I2C_1_SCL
CAM_AF_EN
GPIO_PZ0
LCD_BL_PWM
GPIO_PE6
I2S_4_LRCK
SPI_2_MOSI
I2S_4_SDOUT
GND
CSB
SDI
SCK
SDO
VDD
VDDIO
BME280
U2