Phoenix Drone: Final Documentation

Contributors:

Jimmy Banh		jtb382@nau.edu
Ziming Li		zl239@nau.edu
Suzan Nasona		sbn26@nau.edu

EE486C: Capstone Design
Team 3: Wildfire Drone
April 16th, 2021

Table of Contents

EE486C: Capstone Design
Team 3: Wildfire Drone
April 16th, 2021

1. Introduction

Our project is sponsored by Dr. Fatemeh Afghah who is an assistant professor at Northern Arizona University (NAU). She is our main client for this project along with her Ph.D. student, Alireza Shamshoara. As a Senior IEEE member and the Director of Wireless Networking and Information Processing Laboratory, she has provided our design team with lab space and equipment necessary to complete the project. Their main focus of work is in wireless communications, signal processing, and unmanned aerial vehicles.

The recent increase in wildfires globally has highlighted its effects in producing environmental pollution, damage or loss of property, crops, resources, animals, and people. Wildfires destroy many private and public infrastructures every year. In recent years, the effects of global warming has resulted in increased wildfires globally. The United States has experienced devastating wildfires in many areas in the west coast. 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. To address this issue, our design team was able to contribute to our client's ongoing effort to monitor wildfires using Unmanned Aerial Vehicles (UAVs). We were to design a wireless communication system that uses Software Defined Radio (SDR) to transmit aerial data such as images, Global Positioning System (GPS) location, and information about the environment surrounding the fire from a drone to a base station. The data from the SDR receiver would be processed at the ground station through an interface with a personal computer to display images of the fire in real-time. This project is a collaboration with the Computer Science team, who were to produce the classification, object detection, and segmentation algorithms for assessing and locating the fire. To satisfy the requirements, we created a system architecture modeling the behavior of the three subsystems of our wildfire monitoring system and identified its main functions. We then constructed the final product through prototyping, building each subsystem, and integrating the system. Lastly, we tested the system to ensure that it meets requirements before delivering the final product to our client. We were able to get a majority of the system functioning how we had outlined in our requirements where the system functions autonomously, transmitting images and text files with sensor information such as temperature, humidity, and altitude based on the fire's location to the ground station upon user request.

2. Design Process

Drone Subsystem

In the air, the drone will be manually deployed to monitor the fire. The drone subsystem will capture aerial images using a Raspberry Pi high definition (HD) camera and FLIR Vue Pro R thermal camera. This will be accomplished through an interface between the two cameras and NVIDIA Jetson Nano microcomputer. A GPS will be utilized to provide the location of the fire.

Additionally, the design of the drone subsystem will incorporate sensor modules such as a temperature, humidity, and GPS sensors. Data transmission of the location, images of the fire as well as the time and date will occur using an interface between the microcomputer and SDR Ettus B205-Mini transmitter.

Communication Subsystem

In the air, our Electrical Engineering team will be responsible for implementing a wireless communication system that transmits aerial images using an interface between the two cameras and NVIDIA Jetson Nano. The Computer Science Subteam will process the data. At the base station, our design team will utilize SDR Ettus B210 Receiver to receive data and display relevant information using a computer interface on to a Graphical User Interface (GUI).

Base Station Subsystem

Data of the fire images, location, date, and time will be received from the drone using a system interface between a ground computer/laptop with Linux OS and SDR Ettus B210 Receiver. The data will be processed and displayed using a graphical user interface (GUI). The information displayed will depend on user selection. The system includes a user friendly interface that enables the user to select between two viewing modes. Mode one selects video streaming of several images in real time. Mode two displays the location of the drone on a map.

Prototyping

Our design team produced three prototypes that are defined within our system architecture. The first prototype is defined within the drone system while the second and third prototypes are defined within the communication subsystem.

a. *Drone Subsystem*

For the drone subsystem prototyping, we worked with the NVIDIA Jetson Nano and the Thermal Camera supplied by our client. With both components, we needed to be able to capture an image and save it to a folder as specified by our client. This prototype is a part of the drone subsystem as it includes the interface between the microcomputer, cameras, sensor modules, and SDR transmitter. From this prototyping, we hoped to learn more information about the operation of the Thermal Camera and become familiar with the NVIDIA Jetson Nano. This included important tasks such as Python coding on the NVIDIA Jetson Nano with a Linux Operating System to drive the camera, utilizing the general purpose input/output (GPIO) pins, and the Camera Serial Interface (CSI). Since we had no prior experience with Python or NVIDIA Jetson Nano, we expected to encounter many difficulties working on this prototype. With this prototype, we reduce the risk to the project as a whole since this prototype accomplishes one of the major functions of our product. Through this prototype, we will be able to capture images of the fire using a thermal camera.

In addition, our client requested that the user be able to switch between different modes of the camera. During the demonstration, we were able to capture images with the thermal camera. However we were unable to complete the additional task of switching the modes of the camera. We learned how to approach that problem with the help of the company that develops the thermal camera. Overall, we believe that the prototype was a success. Valuable information was gathered about the operation of the thermal camera's function and being able to capture images as well as video.

A majority of the time spent in this prototyping process was dedicated to research. As we were unfamiliar with both the NVIDIA Jetson Nano and Python language, we had to reference many example hardware setups as well as relating software programs to better understand the language of Python. We spent approximately 6 to 8 hours a week researching and setting up the hardware necessary. We believed this was a reasonable amount of time required to complete the prototype. Additional time is required to change the modes of the thermal camera. We believe that our approach towards creating the prototype was decent. In the future, we would also like to have started working on the different modes of the camera. We believe that the results of the prototype will influence the rest of the rest of the project in that with the ability to capture these images and video we may be able to integrate it with the computer science (CS) team's image segmentation and object detection algorithms. We will continue to work on the modes of the thermal camera as well to eventually incorporate our GUI for the user's ability to change the viewing modes of the camera.

b. Software Defined Radio Transmitter

The SDR transmitter prototype combined with the SDR receiver form the basic transmission components. The transmitter increases all input signals at low frequency to high frequency signals. Based on the SDR B205 mini signal range and ISM band, we plan to use the 2.4 GHz as our SDR frequency. The transmitter is a decomposition of the data transmission process and is a fundamental function in this capstone. Hence, it is defined within our system architecture. In order to approach this prototype, we researched different articles about the SDR and studied its implementation through countless videos. We expected the SDR and the computer interface to be challenging due to our inexperience with wireless communication. This was a completely new topic and would require some time to learn. Completing this prototype should reduce the risk of failure in our capstone project as it eliminates a key function for data transmission.

As we reflect on this prototyping experience, the biggest challenge is programming code that will drive the transmitter to function properly. We need to know which method is most effective in order to make it successful. Therefore we must compare different data transmission approaches as well as coding languages. The success of this prototype is an important component to our communication subsystem. Since the SDR is part of the data link, if the data link does not function properly then the data connection from the drone to base station would be unsuccessful.

We spent a lot of time working on this prototype but we were unable to successfully implement a portion of it. The results of this prototype indicate that there is more research to be done to fully understand the GNU Radio software.

c. Software Defined Radio Receiver

The SDR receiver prototype was meant to accomplish the task of receiving a text file or signal using the SDR B210. During the initial planning process, we recognized that receiving a signal is essential for data collection from the drone to the base station. This is defined in our system architecture. Through this prototype, we hoped to learn about this aspect of wireless communication. Based on our client's feedback of his personal experience working with the SDR B210 and its driver, the GNU radio, we anticipated that it would be difficult to interface the SDR B210 with its Linux based software. This prototype reduces risk for the entire project by enabling us to work with both software and hardware that is new to our team and become familiar with it. It reduces risk in terms of our schedule. The process of receiving a signal is very intensive and challenging since we must write python code to control the blocks of the graphical based software driver (GNU radio). However, for this prototype we interacted with GNU radio graphically initially to become familiar with the functions of the blocks and how to use them to control the SDR B210.

3. Final Design

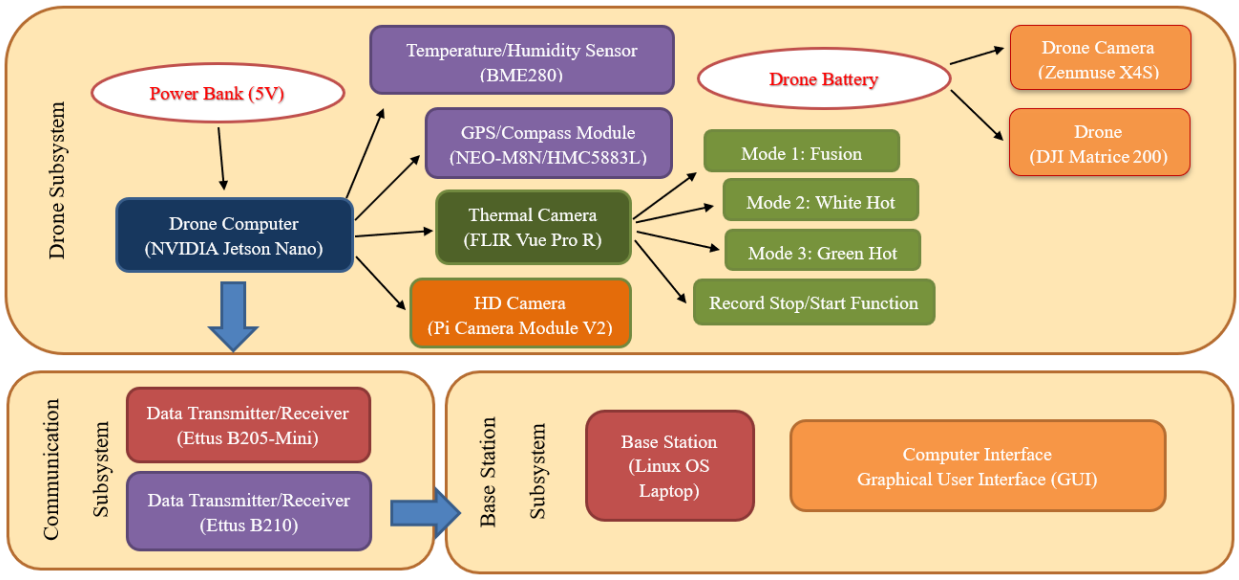


Figure 1. System Architecture

Behavior: Our final project will use Thermal and HD cameras to detect fire in the forest. Meanwhile, we will use sensors which are installed on the drone to continuously record the

temperature, humidity and altitude. We will be using Software Defined Radio (SDR) to tramsmitate all the data and the picture which was captured by these two cameras. We will send commands which were generated by the base station to let the drone send data or the picture.

During the SDR transition, we will use a 2.4GHz signal to send data from the drone to base station and will use 2.5GHz signal to send common from base station to the drone in order to make sure there is no interference.

Once we get the data from the drone, we will use the GUI to show what we get on the base station laptop. Also, we designed several buttons on the GUI so that we can change to classification, objectect detection or other functions that we want the drone to run. After that , our base station GUI will generate the command sending back to the drone and letting the drone change its function.

Major Components of the System

For our drone part, we have an HD camera to capture pictures in the air, The HD camera is connected with a Jetson Nano through a CSI port, The captured images from the HD camera will be saved to a specified folder for transmission through the SDR. Besides the HD camera, we also have a thermal camera to help us distinguish the fire on the ground. We have white mode, green mode, and fusion mode. Those three modes are operated by the PWM connection to the Jetson Nano where we change the duty cycle in order to switch between viewing modes. We will also have temperature, humidity, and altitude sensors to record the data around the drone. We are using a python script to automatically run these sensors every three seconds recording the data and sending that data to the base station.

For our SDR part, we design two ways to transmit the data; one for the text(including sensor information and commands) and one for images (including HD pictures and thermal pictures) on the GNU Radio Software. However, due to some technical trouble, we cannot successfully send pictures with more than 200KB. And because we are planning to run the system in the sky, we have to make the SDR automatically run on bootup on the drone. We have the basic setting of the python code generated by the GNU Radio Software and we put all the functions on one python file. We designed the main function that would utilize the python code generated by the GNU Radio Software so we can operate each function as we want. In that way, we change the destination for file receiving and transmission. We also designed a special button letting our SDR send pictures or sending text from the drone to the base station since we had hardware limitations.

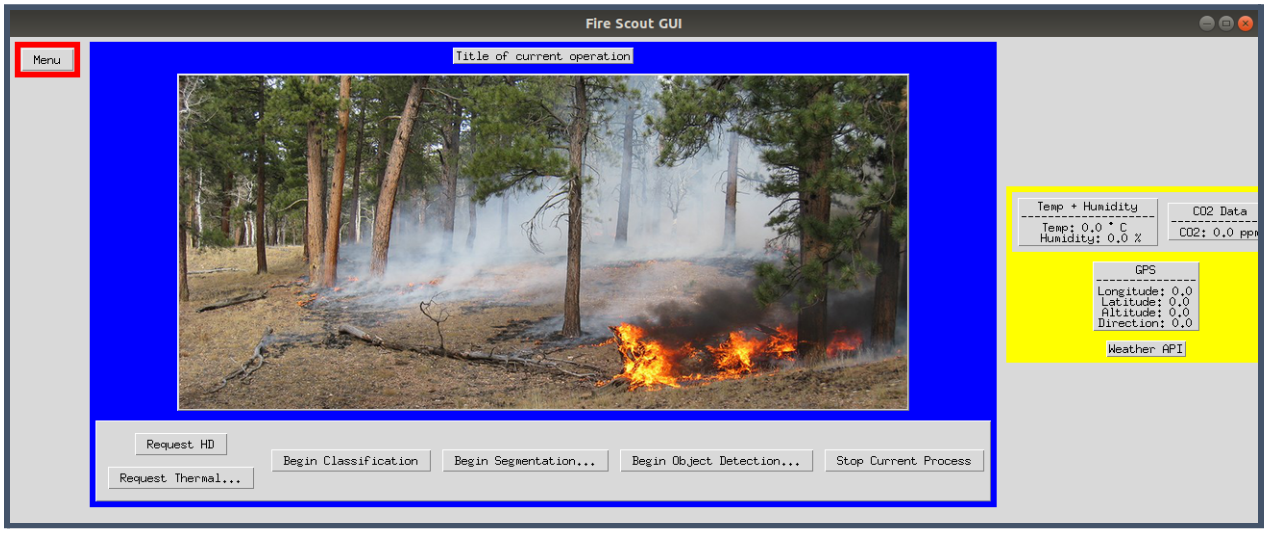


Figure 2. System User Interface

At the base station, we will show the picture and sensor which was received from the drone on our GUI. (In Figure 2) The GUI was designed by our CS subteam. And there are six buttons below the picture, which generate different commands depending on what the user would like to see from the drone. Once that command is generated, it will be sent to the drone through the SDR for the drone to then run and execute the command given.

4. Results

a. The requirements spreadsheet with color indication of requirements testing results

	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

b. Important test results

First 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.

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.

1. Automatic operation and Image capture capability of the Cameras

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.

2. The thermal camera autonomously switches between its three viewing modes

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 is expressed in *Figure 2.*

a. White Hot b. Green Hot c. Fusion

Figure 3. Thermal Images Output from Test

3. Sensor Readings are conducted automatically

The third test determined whether the humidity, temperature and GPS sensors were able to automatically provide data. The sensors were interfaced with the Jetson nano as expressed in Appendix A and B and then a python script was executed to collect the data. During the test, the humidity and temperature sensor was successful 100% of time in generating data. However, the GPS sensor could not be tested.

4. Automatic transmission from drone and base station using SDR

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. The automatic transmission of text or images is driven by a command file found at both the ground station computer and drone microcomputer. When the command files at the base station and drone is left empty the SDR communication enables text to be transmitted and received. When the command files are no longer empty but some random written text then images can be transmitted and received. Throughout this test, our team simulated this process by manually clearing the command files to initiate text transmission and writing to the command files to initiate image transmission. 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. We failed to connect a third antenna on the SDR B210 at the base station. Automatic transmission of images was successful 100% of the time for images captured through 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.

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.

c. *Analysis of Results*

First Unit Test, Step by Step (UTS)

The success rate of executing the fire classification model was unexpected. We anticipated that this model would have a 100% success rate. 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 this algorithm to incorporate this change. Hence, it still has bugs and errors that must be fixed.

Unit Test Matrix (UTM)

1. Automatic operation and Image capture capability of the Cameras

The test results for the first part of the Unit Test Matrix was expected. The HD and thermal cameras functioned properly and were able to automatically capture images upon user request. Hence, this requirement was met.

2. The thermal camera autonomously switches between its three viewing modes

The results of the second part of Unit Test Matrix were as expected. The thermal camera was able to function optimally and automatically switch between the three viewing modes. Hence, this requirement was met.

3. Sensor Readings are conducted automatically

The results of the third part of the Unit Test Matrix was expected. The temperature and humidity sensor was able to automatically provide the sensor data. However, since our design team

was unable to implement the GPS sensor this aspect of the test failed. Hence, this requirement was partially met.

4. Automatic transmission from drone and base station using SDR

The results of the fourth part of the Unit Test Matrix were as expected. The transmission of text was successful 100% of the time while that was not the case for images. In terms of images, 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.

Integration Test

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

5. Conclusion of Capstone Report

a. Most important requirements and their results

Requirement 2.6-2.6.1: The system should be able to transmit images, sensor information and the GPs location from the drone to the base station computer

A majority of the specifications of this requirement was met. The system provides the user with relevant data about the state of the fire. By utilizing sensors it is able to provide the base station with data of the environmental conditions of the surrounding of the fire such as the temperature and humidity. It also provides the user with HD or thermal images of the fire. The system is able to switch between the three viewing modes of the thermal camera: fusion, green hot and white hot. Based on user request, the system is able to provide thermal images using one of the three viewing modes. Unfortunately, our design team experienced difficulties interfacing with the UART serial communication of the Nvidia Jetson Nano and were unable to implement the GPS sensor thus the system is unable to pinpoint the location of the fire.

Requirement 2.8-2.8.1: The system should function autonomously enabling automatic data transmission between the drone and base station subsystems

This requirement was met. The system is able to function autonomously providing the user with sensor data in the form of a text file and real time images of the fire in both HD and

thermal using the SDR from the drone to the base station computer. At the base station, our SDR communication design text files are received every three seconds and images every eleven seconds. The SDR communication design has limitations where it is able to successfully receive images that are less than 200KB whereas transmission of images larger than 200KB result in data loss or shifting.

Requirement 2.9: The system should integrate the Computer Science team's algorithms

This requirement was met. The system integrates the Computer Science team's object detection, image segmentation and fire classification algorithms. Once the system detects a fire, it provides the user with images that utilize these algorithms to assess the state of the fire. The object detection model forms bounding boxes around objects, includes a label describing the object and the percent accuracy of that prediction. The fire classification model labels whether there is a 'fire' or 'no fire' detected in the frame and the accuracy of that prediction.

b. Lessons Learned

Throughout the design process, our team learned important lessons. These lessons include gaining an experience in wireless communication using Software Defined Radio; familiarity with the Nvidia Jetson Nano microcomputer, its GPIO and SCI ports for autonomous systems; deepening our familiarity with the Linux Ubuntu System; improving our python programming skills; gaining experience working with the Raspberry Pi camera and utilizing the duty cycle of the thermal camera to switch between the three camera modes. In addition, we learned to collaborate within our design team and an interdisciplinary team on a common task; improved our project management skills as well as our problem solving and critical thinking skills. These lessons are important to our personal and professional growth in the field of electrical engineering.

6. User Manual

a. Introduction

We, the members of the Phoenix Drone, would like to extend our deepest gratitude for giving our team the opportunity to design a wildfire monitoring system for you. As our client, we hope that you are satisfied with the outcome of our product and hope that you find great use in it for years to come! As it stands, there is a definite need for development in the area of wildfire control and monitoring for the reason of the damage and costs that come with the spread of wildfires. Currently, the technology used to monitor wildfires are slow and permit extra time for the wildfire spread which requires more time and effort to respond to. With our system, we are about providing users with real-time information about the situation of the fire which could then be used to determine a plan-of-action for responders. With our product, we have provided the hardware as well as how to set up the software that could be used for any drone system, it will only require a mounting mechanism depending on the drone in use. Once mounted onto a drone, our system on bootup will be able to provide you with the information you request in a three-dimensional space while being away from any harm with the interaction of the Graphical User Interface (GUI) at our base station.

Working on our system, we decided to break it up into three subsystems that were worked on separately, then later combined together to make our product. These subsystems consisted of the drone, communication and base station. This was an essential step in our design process for the overall system, all to ensure the functionality of each subsystem on their own before integrating them with each other. On the drone subsystem, we wanted to make sure that all of the components were working as they should: sensors reading accurate values, cameras capturing images, and thermal camera could change viewing modes. With communication subsystem, we made sure that the transmission of data was stable between the Software Defined Radio (SDR) devices, this included being able to send images and text from one device to the other and view that the information has been received. While at the base station, for functionality, we wanted to make sure that GUI would be capable of displaying the information received from the drone.

For our product, we would like to highlight some of the features in the system we believe you will enjoy:

- With the Flir Vue Pro R in our system, you will be able to switch viewing mode of the camera that will show the user a whitehot, greenhot, or a fusion view of the situation.
- With the collaboration of a Computer Science team, the system has algorithms of classification, fire and object detection, and segmentation for further identification of the events/situation.
- The GUI has been created for the ease of use and simplicity in mind for any user, with buttons that are pretty self explanatory in what information the user wishes to receive from the drone.

- Because the hardware is set up separately from the drone, the system is configurable with most drones with the right mounting equipment.

We are happy to have had you as our clients, this way, we were able to easily define the requirements and where our work needed to be focused on. Without the constant communication and support with our client, we would not have been able to produce a system such as this one. Please review the rest of the user manual for more information about how our system works!

b. Installation

<u>Drone Microcomputer NVIDIA Jetson Nano:</u>
Items Needed

- NVIDIA Jetson Nano Developer Kit
- USB Keyboard and Mouse
- Display Monitor (HDMI)
- 32 GB MicroSD Card (minimum)
- Micro-USB or DC Barrel Power Supply

Preparing the microSD Card
(Requires a Computer with internet access)

1. Download the file from this link: https://developer.nvidia.com/jetson-nano-sd-card-image
2. This file will then need to be written to the microSD card. For further instructions on this step for different Operating Systems refer to this link: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write

Setting Up and Booting Up the Jetson Nano

1. Insert the written microSD card into the Jetson Nano located on the underside of the device.
2. Connect the USB mouse, USB keyboard, and Monitor into their respective locations.
3. Plug in the power supply (DC power or MicroUSB)
4. Upon powering the Jetson Nano, a green LED will light up.
5. On the first boot up of the system, you will need to complete the next tasks:
    a. Review and accept NVIDIA Jetson software EULA
    b. Select system language, keyboard layout, and time zone
    c. Create username, password, and computer name
    d. Select APP partition size—it is recommended to use the max size suggested
6. This will complete the set up for the Drone's Microcomputer. For more information on how to setup the Jetson Nano, please refer to https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro

Software Defined Radio Communication System:

We used the verion 3.7.11.5 for GNU Radio software. Therefore, there is a way to install that software.

1) Prepare a computer and OS is Ubuntu (18.04 is preferred)
2) Open in terminal
3) Enter the code below

```
$ sudo add-apt-repository ppa:gnuradio/gnuradio-releases-3.7
$ sudo apt-get update
$ sudo apt install gnuradio
```

4) The GNU software should be ready to use

At this moment, we already installed the GNU radio. But since we need to use SDR drive the Ettus B210 and B205mini, we also need to install hud image to drive our device. We will use UHD Images Downloader to download that.

1) Open in terminal
2) Running "<install-path>/lib/uhd/utils/uhd_images_downloader.py"
3) Waiting for installation.
4) Open another terminal and running " uhd_find_devices"
5) If the terminal tell you the serial for Ettus device then you get success

Thermal Camera FLIR Vue Pro R:

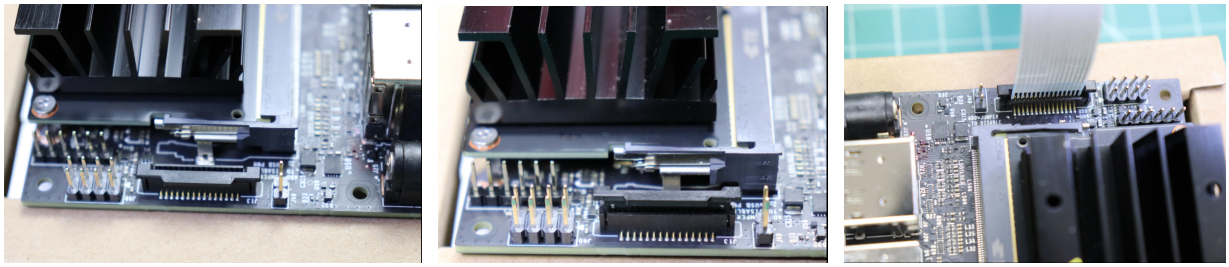
Items utilized for the Thermal Camera

- FLIR Vue Pro Camera with Camera Mount
- MicroSD Card
- Bench Cable
- Accessory Cable

For operation of the Thermal Camera, it will need a 4.8-6.0V power supply which will be given from the USB connection from the Jetson Nano using the Bench Cable this will also allow access from the Jetson Nano to the memory card of the thermal camera. It is important that you DO NOT exceed 6.0V for the input voltage, otherwise it will damage the camera! We will be using PWM3 and PWM4 of the Accessory cable for the operation of the viewing modes as well as a way to record information for manipulation through Pulse-Width-Modulation (PWM) from the Jetson Nano pins. PWM3 and PWM4 will have two wires each, one for ground and the other is for the PWM signal. Respectively, they will be attached to ground pins of the Jetson Nano and then the signals for PWM3 and PWM4 will be connected to pins 32 and 33. This will require the user to reconfigure the pins on the Jetson Nano, see "Reconfiguring the Jetson Nano GPIO Pins" in Configuration of the User Manual for further instructions. In our system, PWM3 is for the purpose of changing the different viewing modes, while PWM4 is meant to stop/start recording of the thermal camera.

HD Camera:

The installation of the Raspberry Pi camera requires that python 3 Open Source Computer Vision Library (openCV) software is installed on the Nvidia Jetson Nano. Our system operates with openCV2 version 4.1.1. We recommend that this version be installed. We have imported the OpenCV2 library on the python script that captures an image using the HD camera and must be made available to operate the camera. Then, the Raspberry Pi camera must be interfaced with the CSI port of the Nvidia Jetson Nano as shown in figure 4. Begin by first opening the CSI port camera connector as shown in Figure 4b and then place the cable of the raspberry pi camera with the contacts facing towards the Jetson Nano as shown in Figure 4c.



a. Camera connector closed *b. Camera Connector open* *c. Camera connected with Pi*

Figure 4. The Raspberry Pi camera interfaced with the CSI port of the Nvidia Jetson Nano

The Raspberry Pi camera is now installed and ready for operation.

Temperature, Altitude, and Humidity Sensor:

For our system, we will be using an Adafruit BME280 for reading temperature, altitude, and humidity values. We will need to download the necessary libraries to run the sensor.
Installing the BME280 Library

1. On the Jetson Nano, you will need to open up a terminal.
2. In the terminal, type to download the library onto the Jetson Nano:
   `$ pip3 install adafruit-circuitpython-bme280`
3. To update the library for any bug fixes, use this in the terminal:
   `$ pip3 install adafruit-circuitpython-bme280`
4. Library for the sensor should be installed. To check installation, use in the terminal:
   `$ pip3 list`

c. Configuration and Use (All)

Software Defined Radio Communication System: suzan

To activate the SDR communication system first run the python script for the SDR communication system at the base station and then run the python script for the SDR communication system at the drone. The python script must be executed using python2. The complete communication system is now ready to receive a command from the GUI for operation. The autonomous operation of the SDR is driven by the user. The user interacts with the GUI indicating what type of information that they wish to receive about the state of the fire. Therefore, before proceeding please refer to the Computer Science team's user manual to install, configure and then use the GUI. With the GUI installed, it may be activated by running the GUI.py python script. Once the GUI has been made available to the user, the user may press a button on the GUI indicating whether they would like to receive HD images, thermal images, sensor information, object detection, image segmentation or fire classification data. The GUI then produces a GUI output text file that the ground station SDR communication system transmits and is received by the drone SDR communication system. At the drone subsystem, the task specified by the GUI output text file is executed. For example if the user requests HD images of the fire then the HD camera is activated and once a fire is detected an image is captured. This image is then transmitted from the drone through the SDR and is received at the base station computer. Finally, the GUI outputs the received image to the user for viewing.

Thermal Camera:

In our system, PWM3 is for the purpose of changing the different viewing modes, while PWM4 is meant to stop/start recording of the thermal camera.

Setting Up the PWM Functions of the Thermal Camera

1. Download the FLIR Vue Pro Application on a phone. Enable Bluetooth on your phone.
2. Open the App and connect with the camera. On the home screen, click on the settings and go to the Accessory Port Tab and enable both PWM3 and PWM4. For our system, PWM3 will have three states for the color viewing modes (whitehot, greenhot, and fusion) and 2 states for PWM4 (stop and start recording).
3. Now the thermal camera is set up for use.

Reconfiguring the Jetson Nano GPIO Pins

1. Open up a terminal on the Jetson Nano. Expand the window of the terminal for easier use.
2. Enter the following command in the terminal:
   `$ sudo /opt/nvidia/jetson-io/jetson-io.py`
3. Once entered, this screen will display in the terminal.

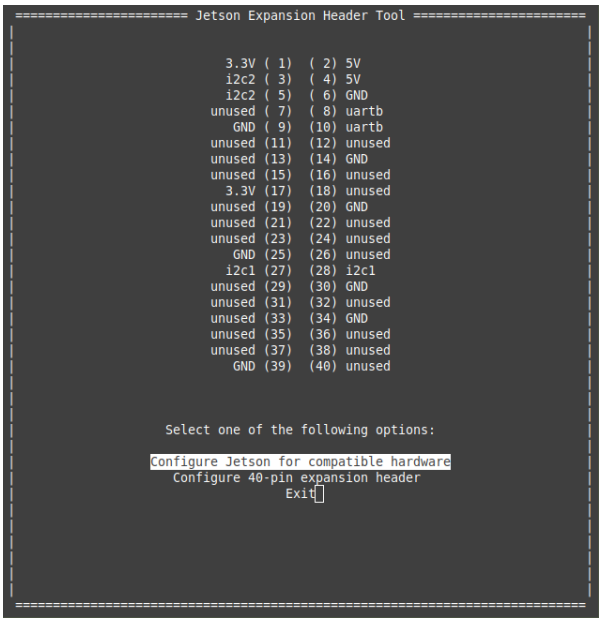


Figure 5. Jetson Expansion Header Tool Screen

4. Using the Arrow Keys on the keyboard move to “Configure 40-pin expansion header” and press enter.
5. A new screen will display as shown below. Once again with the keyboard, move to “pwm0” and press enter to enable PWM output for the Jetson Nano. Do the same for “pwm2”. Then move to “Back” and press enter.

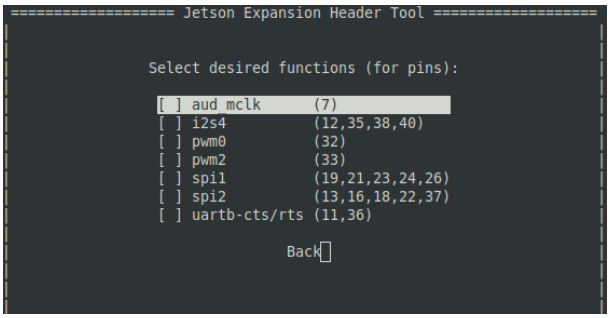


Figure 6. Jetson Expansion Header Tool Screen

6. Now it will return you to a similar screen from Step 3. Now you will want to move to “Save and reboot to reconfiguring pins”. The system will then begin a reboot.

Figure 7. Jetson Expansion Header Tool Screen Updated

7. Now your Jetson Nano will be able to produce PWM outputs on pins 32 and 33.

HD Camera:

The automatic operation of the camera requires that the GUI and SDR communication are set up and in operation. Please refer to the User Manual provided by the computer Science team to set up the GUI and to this user manual to setup the Software Defined Radio communication subsystem (6a and 6b). Next, the user must request to turn on the camera from the GUI. A request is made by pressing the button on the GUI that turns on the HD camera. This activates the SDR communication system initiating the transmission of the GUI output text file from the base station to the drone. Once at the drone, the GUI output text file communicates to the drone that the user requests HD images from the scene of the fire. Once a fire is detected the HD camera captures that image and transmits it to the base station. The GUI receives that image and displays it to the user.

Temperature, Altitude, and Humidity Sensor:

For the BME280 Sensor, we will be using four pins of the Jetson Nano: 3.3V (pin1), GND (pin9), I2C_2_SDA (pin3), and I2C_2_SCL (pin5). On the sensor, we will be using Vin, GND, SDI, and SCK respectively. The other connectors of the sensor will be not connected to anything.

d. Maintenance (All)

Software Defined Radio:

Please keep Ettus B205mini running between -40 – 75 °C and Ettus B210 running around 25 °C. Once using the SDR, connect three antennas with Ettus B210 and connect two antennas with Ettus B205mini (in Figure 8 shows)

Figure 8. Ettus B210 and Ettus B205mini Connection to Antennas

Thermal Camera:

For the thermal camera, the operational temperature should be between -4^{0}F to 122^{0}F. The rated altitude for the camera is tested by the company and should not exceed 40,000 ft. If the software or firmware receive any updates of fixes please refer to the instructions below on how to update the device.

Software/Firmware Update

Application Update

1. Any update for the app can be found on the respective app store.

Firmware Update

1. The latest firmware for the camera will be found here: http://www.flir.com/suas/vuepro/software/
2. Backup the camera settings. Only make note of the settings you enabled. With the update the settings will reset to default settings.
3. Download the firmware update to a computer with internet access with the camera connected through the Bench Cable and save the firmware to the microSD memory.
4. You will then need to power the camera for interaction through the app.

5. Now in the app, you will begin the update process with the new firmware. This will enable another re-boot.
6. You will now reset the device to its factory defaults. This can be done in the advance tab, then the About tab, and should have an option to reset to factory defaults.
7. All the firmware should be updated and can be checked through the settings. From step 2 revert any settings you had before the update to the newly updated device. For more information, please refer to the user manual of the FLIR Vue Pro: https://www.flir.com/products/vue-pro-r/

HD Camera:

The maintenance of the Raspberry Pi camera requires that the camera not be susceptible to the heat radiating off of the vent of the Nvidia Jetson Nano. Close proximity to heat may compromise the camera's functionality. After interfacing the camera with the Nvidia Jetson Nano, do not attempt to break this connection while the microcomputer is powered on. Doing so may adversely affect the camera and eventually comprise its functionality. Also, we recommend that OpenCV2 version 4.1.1 be installed in python 3 to operate the camera. If an update is available, please determine first if this new version is compatible with the Raspberry Pi camera Module 2 before installing.

Temperature, Altitude, and Humidity Sensor:

For maintenance on the BME280, we recommend that if removed from the system you ground yourself by touching a metal object, this is to prevent ESD harming any part of the sensor or system. The operating temperature for the device is from -40^{0}C to 85^{0}C and should not surpass the limits if you would like better accuracy.

e. Troubleshooting Operation

Software Defined Radio:

If images or text is not being received at either the base station or drone subsystem:

Make sure that at the base station three antennas are connected to the SDR B210. Antenna port J801 is used to transmit text at a frequency of 2.5 GHZ, port J802 is used to receive text at a frequency 2.3 GHZ and port J803 is used to receive images at a frequency of 2.4. At the drone two antennas must be connected to TRX and RX2 of the SDR B205-mini.

Thermal Camera:

If the camera is not changing viewing modes:

Ensure that the PWM3 wires are hooked up to their corresponding pin numbers on the Jetson Nano, in this case pin 32 and GND.

If the camera is not recording:

Ensure that the PWM4 wires are hooked up to their corresponding pin numbers on the Jetson Nano in this case pin 33 and GND.

HD Camera:

If the system cannot find the HD camera:

Make sure that the camera is in fact properly interfaced with the CSI port of the Nvidia Jetson Nano. Refer to the installation section of the User Manual for reference on how to do this properly (section 6b).

Temperature, Altitude, and Humidity Sensor:

If you are unable to get values read from the sensor:

Ensure that the pins of the BME280 sensor are connected to the right pins of the Jetson Nano. Refer to the figure below for extra guidance.

Connection	Function	Jetson Nano 40-Pin Header		Function	Connection
Temperature/ Humidity Sensor Vin (Red Wire)	3.3V	1	2	5V	X
Temperature/ Humidity Sensor SDI (Blue Wire)	I2C_2_SDA	3	4	5V	X
Temperature/ Humidity Sensor SCK (Green Wire)	I2C_2_SCL	5	6	GND	GPS Sensor GND (Brown Wire)
X	---	7	8	UART_TX	GPS Sensor Rx (Orange Wire)
Temperature/ Humidity Sensor GND (Black Wire)	GND	9	10	UART_RX	GPS Sensor Tx (Yellow Wire)

f. Conclusion

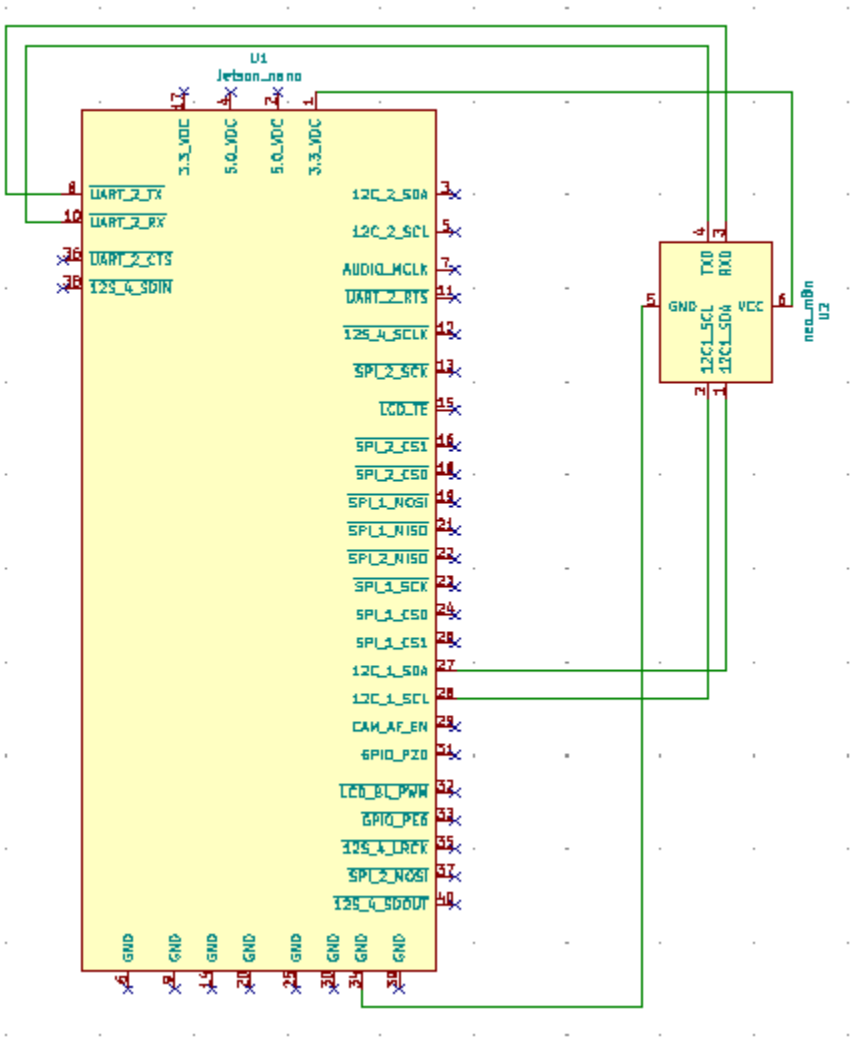
Our design team expresses our deepest appreciation to you, our client for your guidance throughout this design process. It has been a pleasure working on this project and contributing to your effort to monitor wildfires. We wish you many years of productive use of the product. While we are all moving on to new adventures and the next chapter of our journey to developing our professional careers, we would be happy to assist you to get the product working in your organization. Please do not hesitate to contact us, we may be reached at:

Suzan Nasona: sbn26@nau.edu
Ziming Li: zl239@nau.edu
Jimmy Banh: jtb382@nau.edu

7. Appendices

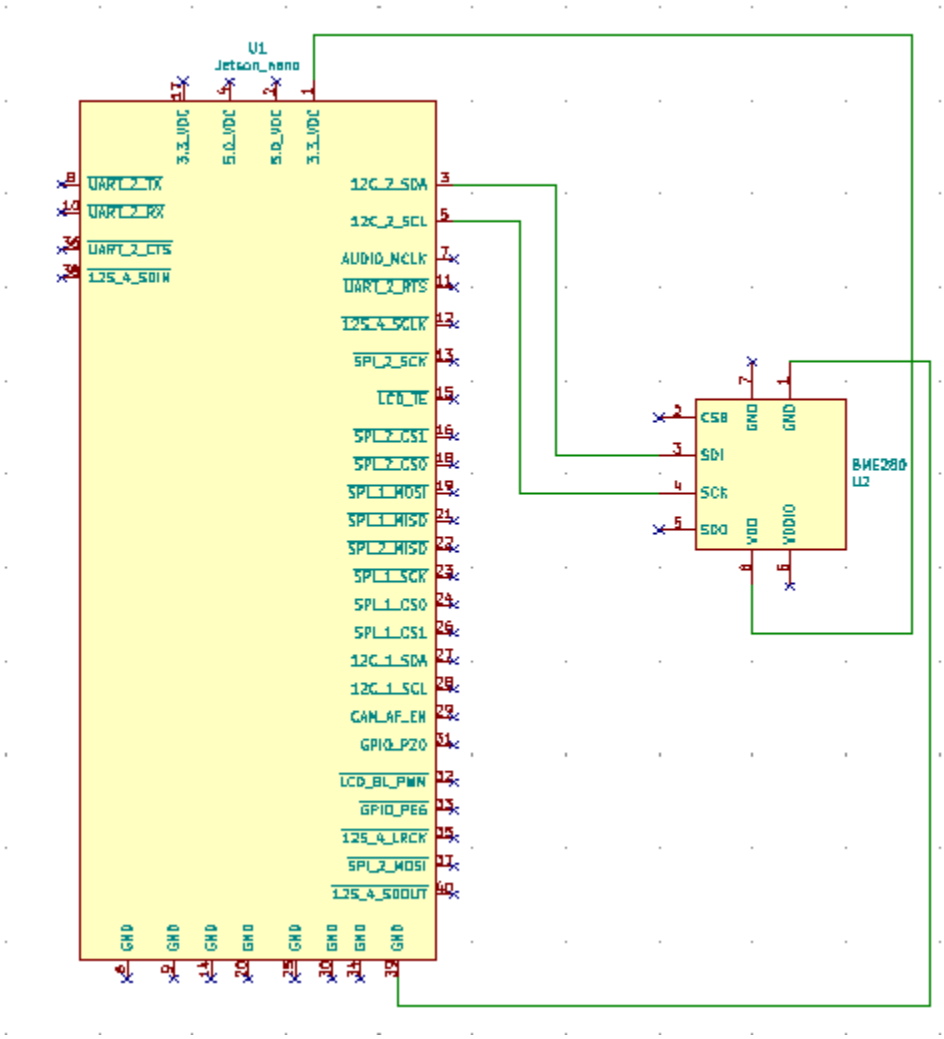
Appendix A. Global Position System (GPS) Design

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



Appendix B. Temperature and Humidity Sensor Design

The schematic of the implementation of the BME280 temperature and humidity sensor interfaced with the Nvidia Jetson Nano



Appendix C. 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 D. 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/Th
ermalFrame_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/tes
t_images/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_imag
es/ThermalFrame_0.jpg')

open('/home/spring2021/Desktop/Capstone/fire_scout_system/drone_station/test_images/Th
ermalFrame_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 E. Jetson Nano Pinout with Devices Connected

These are the pin connections used throughout the design process to interface the thermal camera, GPS and compass sensor, as well as the temperature and humidity sensor with the Nvidia Jetson Nano.

Connection	Function	Jetson Nano 40-Pin Header		Function	Connection
Temperature/ Humidity Sensor Vin (Red Wire)	3.3V	1	2	5V	X
Temperature/ Humidity Sensor SDI (Blue Wire)	I2C_2_SDA	3	4	5V	X
Temperature/ Humidity Sensor SCK (Green Wire)	I2C_2_SCL	5	6	GND	GPS Sensor GND (Brown Wire)
X	---	7	8	UART_TX	GPS Sensor Rx (Orange Wire)
Temperature/ Humidity Sensor GND (Black Wire)	GND	9	10	UART_RX	GPS Sensor Tx (Yellow Wire)
X	---	11	12	---	X
X	---	13	14	GND	X
X	---	15	16	---	X
GPS Sensor 3.3V (Red Wire)	3.3V	17	18	---	X
X	---	19	20	GND	X
X	---	21	22	---	X
X	---	23	24	---	X
X	GND	25	26	---	X
Compass Sensor SDA (White Wire)	I2C_1_SDA	27	28	I2C_1_SCL	Compass Sensor SCL (Green Wire)
X	---	29	30	GND	Thermal Camera PWM GND (Black Wire)
X	---	31	32	PWM0	Thermal Camera Color PWM (Orange Wire)
Thermal Camera Start/Stop PWM (Blue Wire)	PWM1	33	34	GND	Thermal Camera PWM GND (Black Wire)
X	---	35	36	---	X
X	---	37	38	---	X
X	GND	39	40	---	X