Notre Dame University-Louaize

Faculty of Engineering

Department of Electrical, Computer and

Communication Engineering

Fire Mapper

By:

Daniella Sarkis - 20172092

Mohamad Diab - 20162542

May, 2022

FIRE MAPPER

A Senior Project Report

Presented in Partial Fulfillment of the Requirements for

the Degree Bachelor of Engineering in the

Faculty of Engineering of Notre Dame University

By:

Daniella Sarkis

Mohamad Diab

Notre Dame University - Louaize

Spring 2022

Advised by:

Dr. Jad Atallah

Department of Electrical, Computer, and Communication Engineering

Fire Mapper

A Senior Project Report Presented in Partial Fulfillment
of the Requirements for the Degree Bachelor of
Engineering in the Faculty of Engineering of Notre Dame
University

By:

Daniella Sarkis

Mohamad Diab

*****

Notre Dame University

Spring 2022

Committee Member

Dr. Abdallah Kassem

Advised by:

_____

Dr. Jad Atallah

Department of Electrical, Computer, and
Communication Engineering

Disclaimer

This report is prepared by students of the Faculty of Engineering at Notre Dame University – Louaizeh, who are the sole responsible for the integrity of its content. The advisor and committee member act only in an advisory capacity and cannot be held responsible, under any circumstances, for any eventual misuse or violation of intellectual property laws.

ABSTRACT

This project aims at building an air-based fire detection system using Insta360 Go 2 camera, Arduino Uno and a drone.

We employed a system engineering approach in this project, combining different software and hardware elements to create a unique automatic fire detection system.

To detect the fire, this project uses a simple fire detection Matlab algorithm based on a threshold. Furthermore, to receive the drone's location it uses an Arduino Uno circuit attached to the drone to work as a transmitter, and a second Arduino Uno circuit connected to the PC to work as a receiver of the drone’s location.

ACKNOWLEDGMENTS

Without the help and support of wonderful people, as well as their valuable advice and feedback, this project would not have been a success. First and foremost, we want to express our gratitude to Dr. Jad Atallah for his assistance.

Dr. Atallah has been extremely helpful throughout this project, from providing us with valuable feedback at our weekly meetings to constantly pushing us to complete the project on time.

When we came to a fork in the road, he led us in the right path. Without Dr. Atallah this project would not have been successful.

We'd also want to thank Dr. Abdallah Kassem for his contribution to the knowledge we gained throughout our time at NDU, which allowed us to complete this project.

A particular thank you to Notre Dame University's Faculty of Engineering for the time and effort it has put into us, as well as its dedication to providing us with all of the resources we need to become the successful engineers we aspire to be.

TABLE OF CONTENTS

Abstract……iv

Acknowledgments……v

Table of Contents……vi

List of Figures……viii

List of Tables……x

CHAPTER 1-INTRODUCTION……1

1.1-Problem Statement……1

1.2-Project Needs and Objectives……2

1.2.1-Project Needs……2

1.2.2-Project Objectives……3

1.3- Knowledge and Skills……3

CHAPTER 2-STANDARD CONSTRAINTS AND REQUIREMENT SPECIFICATIONS ……5

2.1-Relevant Realistic Standards……5

2.2-Relative Realistic Constraints ……5

2.2.1- Economic Constraints……5

2.2.2- Health and Safety Constraints ……6

2.2.3- Social Constraints ……6

2.2.4- Environmental Constraints ……6

2.2.5- Ethical Constraints ……7

2.2.6- Manufacturability Constraints ……7

2.2.7- Sustainability Constraints ……7

2.3- Requirement Specifications……8

CHAPTER 3-FUNCTIONAL DECOMPOSITION……9

CHAPTER 4- PROJECT MANAGEMENT……14

4.1- Gant Chart……14

4.2- Work Breakdown structure……15

4.3- Cost Analysis……17

4.4-System Failure Modes And Effects Analysis……18

CHAPTER 5-PROJECT SOFTWARE AND HARDWARE IMPLEMENTATION ……22

5.1- Schematic and overview of the whole system ……*22*

5.2- Schematic of the transmitter circuit ……*23*

5.3- Schematic of the receiver circuit ……*25*

5.4- RF Transmitter & Receiver(433MHz) ……*26*

5.7-Flow Charts ……*27*

CHAPTER 6-SYSTEM TESTING AND FINALIZING……**33**

6.1-GPS ……*33*

6.2-Image Processing……*36*

CHAPTER 7-ACHIEVED REQUIREMENTS……38

CHAPTER 8-SUMMARY, CONCLUSION, AND FUTURE WORK……39

8.1-Summary……39

8.2-Conclusion……39

8.3-Future……40

REFRENCES……41

APPENDIX……42

LIST OF FIGURES

Figure 1.1- Objective tree 3

Figure 3.1- Level 0, Fire Mapper Design 9

Figure 3.2- Level 1 10

Figure 4.1.1- Gant Chart..... 14

Figure 5.1.1- Schematic of the Fire Mapper.....22

Figure 5.2.1- Schematic of the Transmitter 23

Figure 5.2.2- Schematic of the Transmitter using fritzing .....…...23

Figure 5.2.3- Neo 7m Connections..... 24

Figure 5.3.1- Schematic of the Receiver 25

Figure 5.3.2- Schematic of the Receiver using fritzing..... 25

Figure 5.4.1- Transmitter Receiver Module 26

Figure 5.5.1- Simplified flow chart of the procedure till we do image processing…27

Figure 5.5.2- Image Processing flow chart 29

Figure 5.5.3- Receiver flow chart 31

Figure 5.5.4- Transmitter flow chart 32

Figure 6.1.1- Latitude and Longitude of the transmitter Circuit..... 33

Figure 6.1.2- Drone Location in Google Maps 34

Figure 6.1.3- Receiver location 34

Figure 6.1.4- Receiver location 35

Figure 6.1.5- Timer 35

Figure 6.2.1- Drone close to fire..... 36

Figure 6.2.2- Drone far away from fire 37

LIST OF TABLES

Table 2.3.1. System requirements for a Fire type distinguisher……8

Table 3.1- Fire Mapper Level 0 9

Table 3.2- Video 10

Table 3.3- Software 11

Table 3.4- Arduino Uno-1 11

Table 3.5- RF transmitter..... 11

Table 3.6- Neo 7m 12

Table 3.7- GPS Antenna 12

Table 3.8- RF Receiver..... 12

Table 3.9- Arduino Uno-2..... 13

Table 4.2.1- Work breakdown structure..... 15

Table 4.3.1- Cost of equipment used in the project 17

Table 4.4.1- Table showing severity, occurrence, and detection ranking..... 18

Table 4.4.2- FMEA Risk Assessment..... 19

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Since its inception, fire has aided the evolution of human society in a variety of ways. However, if a fire has gotten uncontrolled, it can do serious harm to people and to property. Therefore, it is very important to avoid tragedies that result in the loss of human life and property.

According to statistics issued by the Korean National Fire Agency, there were 40,030 fires in South Korea in 2019, with 284 deaths and 2219 injuries. Furthermore, on a daily basis, 110 fires and 0.8 fire-related fatalities occurred, resulting in fire property damages of 2.2 billion KRW. In 2020, a 33-story tower block building fully burned down in Ulsan, and a warehouse blaze happened in Incheon, both in major Korean cities, killing more than 50 people.

Because of the ongoing threat of fire to both economic properties and public safety, fire detection systems are attracting a lot of attention [1].

As a result, Fire Mapper was created. This system is made to be effective in situations where having an insight of the fire spread area is needed. Having an overview of the geographical spread of a fire gives the fire fighter a clear idea on what they are facing, and helps them in being one step ahead in controlling the fire.

1.2 Project Needs and Objectives Statements

1.2.1 Project Needs

As the number of fires in the world grows, there is a greater desire to build fire systems that can identify fires especially at early stages. Sensor technology has long been employed in fire detection, with physical features like pressure, humidity, and temperature, as well as chemical parameters like carbon dioxide, carbon monoxide, and nitrogen dioxide, being measured. However, for a variety of reasons, including as high cost, energy consumption by the sensors, and the required proximity of the sensor to the fire for precise sensing, these systems are difficult to implement in vast open spaces, resulting in physical damage to the sensors [2]. Furthermore, sensor systems have a high rate of false alarms and a long response time [3].

There are various reasons why an image processing-based approach of fire detection is needed and should be used. The first factor is the rapid advancement of digital camera technology, which has resulted in an increase in image quality and a reduction in camera costs. The second factor is that digital cameras may capture images in a variety of formats. Third, the time it takes for a computer to respond is measured in milliseconds. Finally, the whole cost of image processing systems is considered less expensive than present systems.

This image processing-based approach of fire detection will make fire responders more efficient and will help moderate the damages.

1.2.2 Project Objectives

The objective of this project is to map the geographic extent of a fire.

To accomplish that we will build a system using drone, camera, an Arduino Uno and, other modern technologies. The speed, accuracy and user-friendliness will be the main characteristics of this system which will allow quick, informed decisions to be made to save lives and properties.

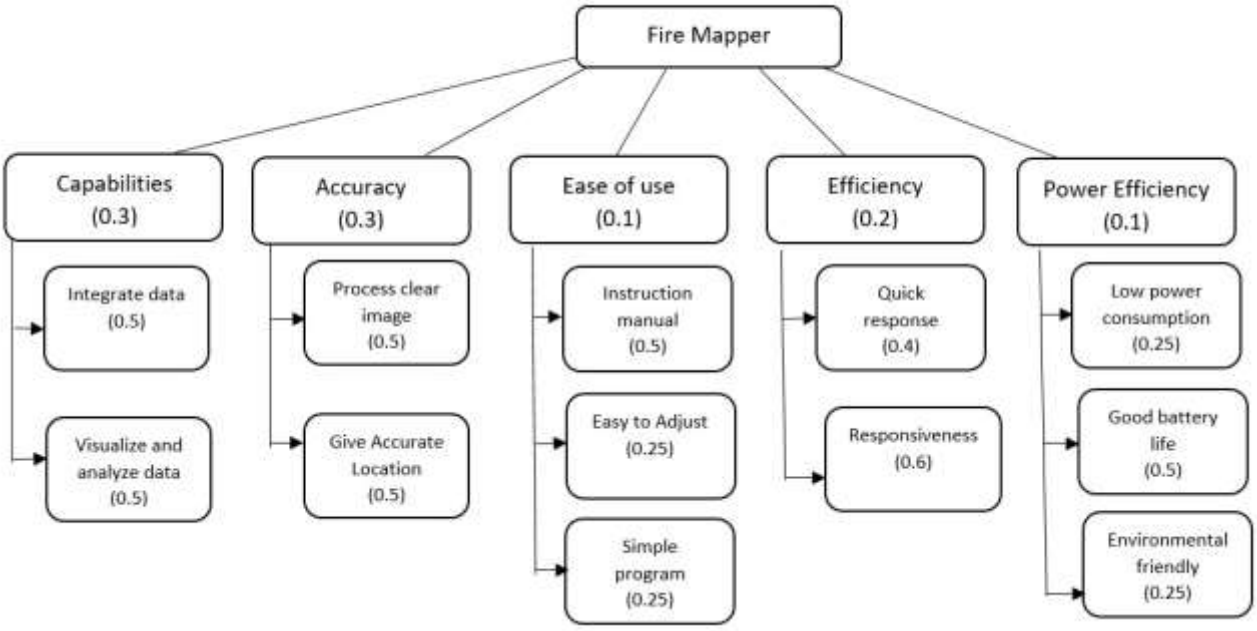

Figure 1.1. Objective Tree.

1.3 Knowledge and Skills

This project relies greatly on electric circuits, microprocessors, communication systems and image processing.

Each circuit whether transmitter circuit or receiver circuit in this project is made up of a microcontroller (Arduino Uno). So the comprehension of these microcontrollers and how they function is of great importance. In addition, sending information from a

transmitter circuit to a receiver circuit, will need the implementation of a suitable communications system between the two microcontrollers while taking into account the range of the signals.

The second main thing that have been acquired from earlier courses and is required in this project is image processing which will be used to detect the presence of fire.

After the completion of this project, we will acquire a deep understanding about Arduino boards and image processing algorithm, how they are programmed and installed in systems.

CHAPTER 2

STANDARDS, CONSTRAINTS, AND REQUIREMENT SPECIFICATION

This chapter will go over all of the constraints that this project may face, as well as the required specifications that apply.

2.1 Relevant Realistic Standards

Standards that are relevant to this project:

- The IEEE 802.11 Wireless Local Area Network (WLAN)/ WIFI.
- The IEEE 610.4 Image Processing.
- The IEEE 1118.1-1990 Microcontroller System Standard Serial Control Bus.

2.2 Relevant Realistic Constraints

2.2.1 Economic Constraints

This project has the potential to save money by minimizing property damage. It would save the government millions of dollars because fires can be easily contained in early stages. Our project does not require that much in terms of funding. However, the system has cost us around twenty eight thousand Lebanese Lira. The insta360 Go2 camera and the drone, has cost us around one thousand dollars. On top of that, there's the cost of the Arduinos, the case, the battery, the breadboard, the LCD... The camera and the drone are the most expensive components.

2.2.2 Health and Safety Constraints

Because the camera will be attached to the drone, the greatest risk is that the camera, or even the drone itself, will fall and hit someone on the ground. There are no health risks associated with utilizing the product because it is mostly automated.

2.2.3 Social Constraints

Because it can inform firefighters early enough, who can then alert nearby residents, and because it can save government money if fires can be extinguished in their early phases, the project has a direct influence on firefighters, people living in dry forest areas, and the government. When it comes to social limitations, they can occur as a result of a project's acceptance or opposition. Regardless of how big or small a project is, public concern and media pressure can sometimes impose greater scrutiny and tighter limits on it, and all of these variables could potentially result in major changes to the original plans.

2.2.4 Environmental Constraints

The most significant environmental consequence associated with the usage of this project is the prevention of fires through the fire detection system. Fires cause problems on the environment, as well as on animals and plant life. This initiative will have a significant impact on the environment since it aims to prevent fires by notifying the fire service by every identified tiny fire or heat source.

2.2.5 Ethical Constraints

When it comes to ethical considerations, our project will be completely educational and non-commercial, according to strong ethical and moral norms that every engineer is taught throughout their college. This project is intended to aid in the saving of lives and the reduction of fire-related damage. Because the system detects indicators of fire, there is no possibility for this device to be misused.

2.2.6 Manufacturability Constraints

When it comes to the manufacturing side of things, choosing the materials hardware/software that will help bring the project to life was one of the first and most important steps. However, not all components and materials were compatible, therefore determining what is needed and what is not will be difficult.

2.2.7 Sustainability Constraints

The battery is the most difficult part of maintaining the entire system. The battery must be recharged and later on after few years replaced. As a result, the system is unsustainable because it requires a new battery every few years, and the old batteries must be properly disposed of. Designing a project that does not require being recharged for example a project that relies on solar power, would be an upgrade. The project would still need a battery, but it would not need to be recharged manually.

2.3 Requirements Specification

Table 2.3.1. System requirements for a Fire type distinguisher

Marketing Requirements	Engineering Requirements	Justification
1-2-4	The system should gather, store, analyze, and show data and information that are spatially related.	In order to detect fire, a lot of information and data should be collected.
1-4	The system should collect and process information via mobile wireless networks.	All data and information are going to be collected from a drone, the camera should send video of the incident wirelessly.
2-3	The system should have the ability to cover many geographical area.	The coverage area depends on the battery capacity of the drone therefore the device will be improved in order to achieve this requirement.
2-3	The system should detect fire from a high altitude.	The strength of the fire varies depending on a variety of circumstances. Therefore, different types of fire should be detected.

Marketing Requirements:

1. The device should be easy to use.
2. The device should give accurate data.
3. The device must priorities safety.
4. The device must have a well-designed communication system that responds quickly.

CHAPTER 3

FUNCTIONAL DECOMPOSITION

This chapter divides the design concept into levels, stating the function of each block clearly.

Figure 3.1. Level 0, Fire Mapper Design

Table 3.1. Fire Mapper Level 0.

Module	Fire Mapper.
Inputs	Videos captured by Insta360 Go 2, location of the drone and a 5V power supply for the Arduino situated on the drone.
Outputs	Data about fire.
Functionality	Detect fire.

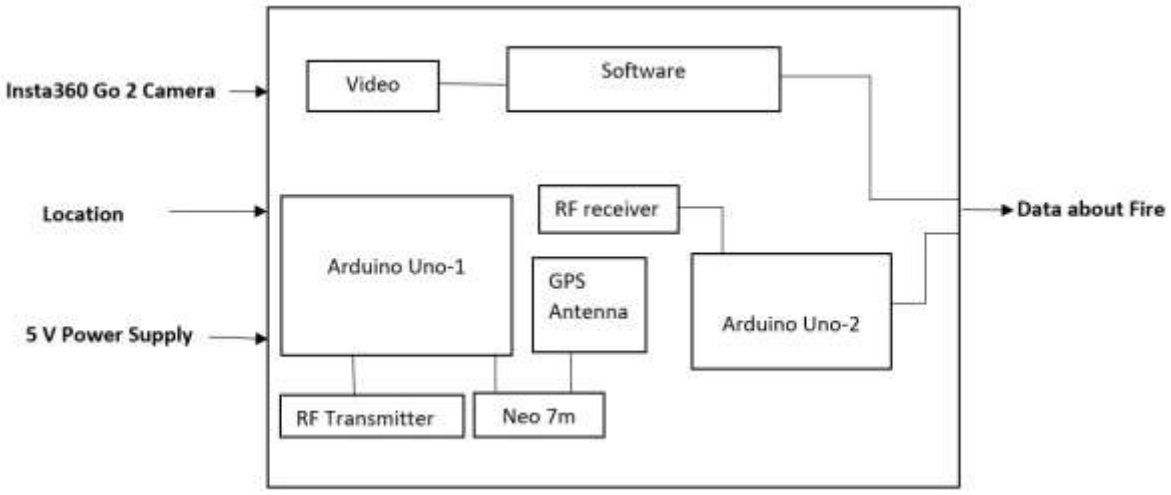

Figure 3.2. Level 1

Table 3.2. Video.

Module	Video.
Inputs	Camera.
Outputs	Recorded videos.
Functionality	Record of the fire scene.

Table 3.3. Software.

Module	Software.
Inputs	Videos.
Outputs	Image Processing.
Functionality	Image processing to detect fire.

Table 3.4. Arduino Uno-1.

Module	Arduino Uno-1.
Inputs	Location and 5V power supply.
Outputs	Signal to the RF transmitter and to the Neo 7m.
Functionality	Produce data to the network.

Table 3.5. RF Transmitter.

Module	RF Transmitter.
Inputs	Power from the Arduino.
Outputs	Output Data to the Arduino.
Functionality	Exchange data from Arduino Uno-1 to the RF receiver.

Table 3.6. Neo 7m.

Module	Neo 7m.
Inputs	Power from the Arduino.
Outputs	Output Data about location to the Arduino.(GPS)
Functionality	Exchange data about location with the Arduino Uno-1.

Table 3.7. GPS Antenna.

Module	GPS Antenna.
Inputs	Data from the Arduino.
Outputs	Output Data to the Arduino.
Functionality	Gives the right location of the drone.

Table 3.8. RF Receiver.

Module	RF Receiver.
Inputs	Power from the Arduino Uno-2.
Outputs	Output Data to the Arduino.
Functionality	Exchange data from RF transmitter to the Arduino Uno-2.

Table 3.9. Arduino Uno-2.

Module	Arduino Uno-2.
Inputs	Data about location from the RF receiver.
Outputs	Data about the location to the PC.
Functionality	Produce data to the network.

CHAPTER 4

PROJECT MANAGEMENT

4.1 Gantt Chart

In order to accomplish our final output, we completed a sequence of tasks over a period of time as shown in figure 4.1.

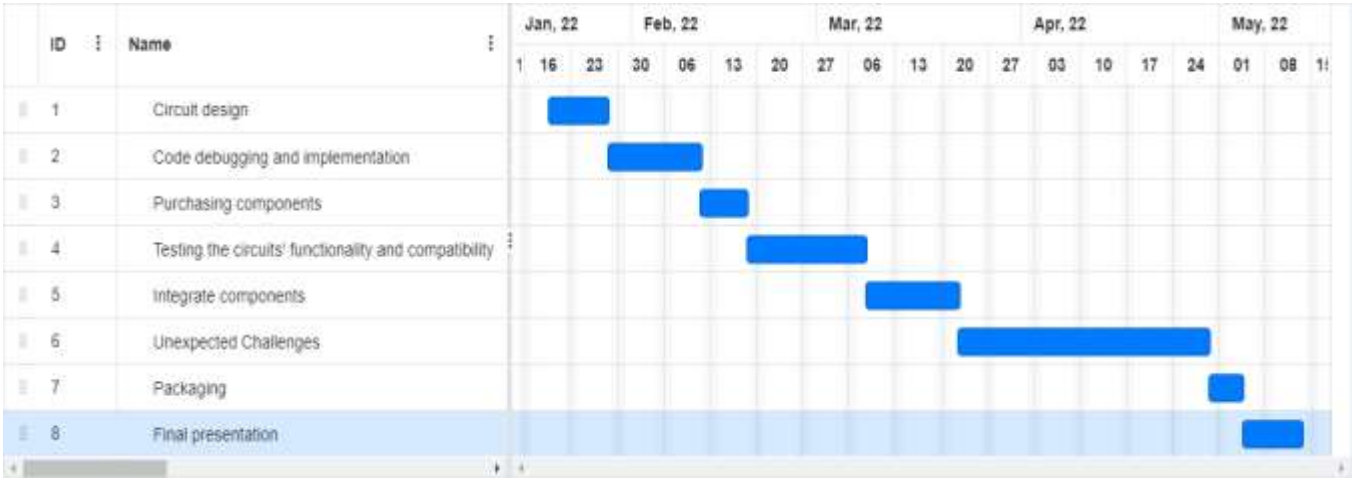

Figure 4.1.1. Gantt chart.

4.2 Work breakdown structure

Table 4.2.1. Work breakdown structure.

Activity	Description	Deliverable	Duration (Days)	People	Resources
Circuit Design	This step consist of designing the circuit that gives the location of the drone.	Circuit	8 Days	Mohammad	PC
Code debugging and implementation	This step consist of implementing codes for the image processing and the transmitter/receiver circuit.	- Image processing algorithm. - Working communication system.	11 Days	Daniella Mohamad	Matlab IDE
Purchasing components	This step is about determining and purchasing the needed components.	Equipment.	6 Days	Daniella Mohamad	Boujikian Incotel EKO
Testing the circuits' functionality and compatibility	This step consist of checking whether the components of the system are compatible and working correctly.	Testing outcomes.	13 Days	Mohammad Daniella	Our hands-on testing

Integrate components	This step consist of connecting the different part of the system together.	System integration.	11 Days	Mohammad Daniella	Arduino IDE Matlab PC
Unexpected challenges	This step consist of testing the operation of the whole circuit.	Working system.	29 days	Daniella Mohammad	Arduino IDE Matlab
Packaging	This step consist of finalizing the work and integrating all the components into one package.	Final Product.	4 Days	Mohammad Daniella	Arduino IDE Matlab
Final Presentation	This step consist of gathering our findings and the result of our testing and debugging into a report.	Final report	8 Days	Mohammad Daniella	

4.3 Cost Analysis

The approximate cost of the equipment used in the drone fire detector project is shown, with description, in Table 4.3.1:

Table 4.3.1. Cost of the equipment used in the project.

Equipment	Description	Approximate Cost ($)
Neo 7m module	**GPS**	**10 $**
GPS Antenna	**Antenna**	**15 $**
Two Arduino Kit	**Arduino Uno / Cables**	**80 $**
Insta360 Go 2	**Camera**	**350 $**
Drone	**DJI Mavic Drone**	**500 $**
Breadboards	**Board to hold electronics components**	**10 $**
RF transmitter and RF receiver	**Electronic devices to transmit and receive radio signals**	**2.5 $**
Battery	**Power supply**	**2 $**
LCD	**Flat panel display**	**5 $**
Total		**975 $**

4.4. System Failure Modes and effects Analysis

Table 4.4.1: A table showing the severity, occurrence, and detection ranking.

Number	Severity	Occurrence	Detection
1	It is not important & could be neglected	Once in 5 years	Almost certain
2	It is not important & it is better to fix it	Once in 2 years	Very high
3	It is more important & it is better to fix it	Once a year	High
4	It is important & it should be fixed	Once in 6 months	Moderately high
5	It is very important to fix it	Once in 3 months	Moderate
6	It cannot work without fixing it	Once a month	Low
7	It might get damaged	Once in 2 weeks	Very low
8	It will slightly get damaged	Once a week	Remote
9	It will get damaged	Once a day	Very remote
10	It will get destroyed	Once in an hour	Absolute uncertainty

Table 4.4.2: FMEA Risk Assessment

Item	Function	Failure Mode	Effect	severity	Cause	occurrence	Detection	RPN
Drone	Provide proper flying.	Low battery life	Possible collision	7	Degraded battery	6	8	336
		Battery leaking	Legal and safety issues	10	Manufacturing defect	2	2	40
		Structural imbalance	Possible collision	10	Unable to bear the weight of the camera and the transmitter circuit	8	7	560
		Structural imbalance	Unable to fly properly	7	High winds and dust	4	6	168
		Motor malfunction	Possible collision	7	Motor mechanical failure	2	2	28
		Losing control	collision	10	Drone is out of range	5	7	350
Camera	Record video of the scene	Fall from the drone	Injuries	10	Not having a strong sticky connection	3	3	90
		Battery draining	Not being able to record the fire	3	The use of the camera for a long period time	6	4	72
		Blurry videos	Unsuccessful image processing	5	Instability of the drone	10	4	200
Arduino circuits	Give location	Arduino operate randomly	System failure	6	Coding error	3	5	90

The RPN was found by using the formula RPN = SEV * OCC *DET.

The Failure Mode and Effects Analysis (FMEA) is a commonly used failure analysis and risk assessment method.

While each phase of the FMEA process is important for a successful analysis, the risk assessment component is critical for identifying the most risk-sensitive areas that must be addressed in order to fulfill our quality and safety goals.

In table 4.4.2 we discussed the failure modes and effects of our Fire Mapper system based on three items: Drone, Camera, Arduino circuits.

First, the drone role is to provide a proper flying however, many failure modes are associated with this item:

1- Low battery life caused by the degradation of the battery and this failure might lead to possible collisions.
2- Battery leaking caused by manufacturing defect and the effect of this failure is some legal and safety issues.
3- Structural imbalance this failure is caused by the inability of the drone to bear so much weight. The effect of this failure is the possibility of collision especially in bad weather.
4- Motor malfunction caused by motor mechanical failure and this might lead to possible collision.
5- Losing control the cause behind this failure is drone out of range, the effect is collision.

Second the camera its basic role is to record video of the scene. The failure modes that are associated with this item are:

1- The camera might fall from the drone the cause is not having a strong sticky connection between drone and camera and this failure might lead to injuries.
2- Battery draining the cause for this failure is the excessive use of the camera for a long period time and this might lead to not being able to record the fire.
3- Blurry videos the cause is an instability in the drone the effect is an unsuccessful image processing.

Finally the failure modes that are associated with the Arduino circuit:

1- Arduino operate randomly the cause for this failure might be some error in the code and this will lead to a system failure.

CHAPTER 5

PROJECT SOFTWARE AND HARDWARE IMPLEMENTATION

5.1 Schematic and overview of the whole system

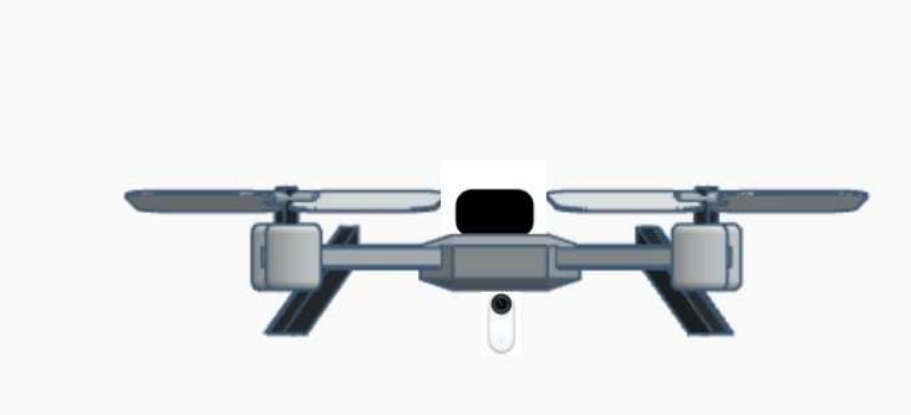

Figure 5.1.1. Schematic of the Fire Mapper

As shown in figure 5.1.1 the system is composed of a drone, a camera and, a small box on top of the drone.

The camera will be used to take videos of the fire, and the box will contain the transmitter circuit which is used to send the location of the drone.

5.2 Schematic of the transmitter circuit:

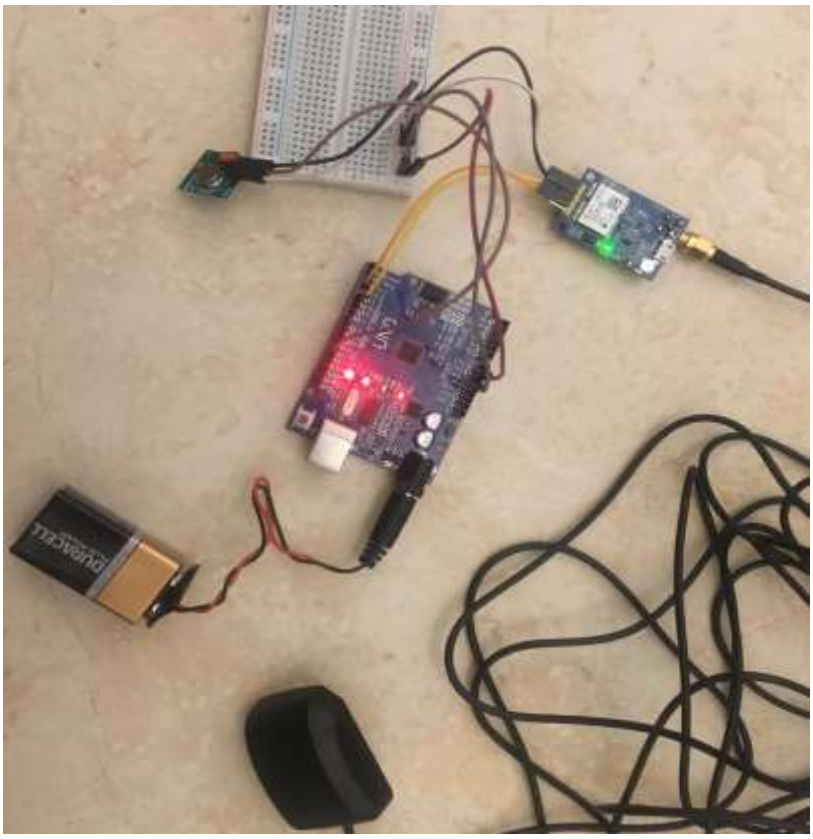

Figure 5.2.1. Schematic of the Transmitter

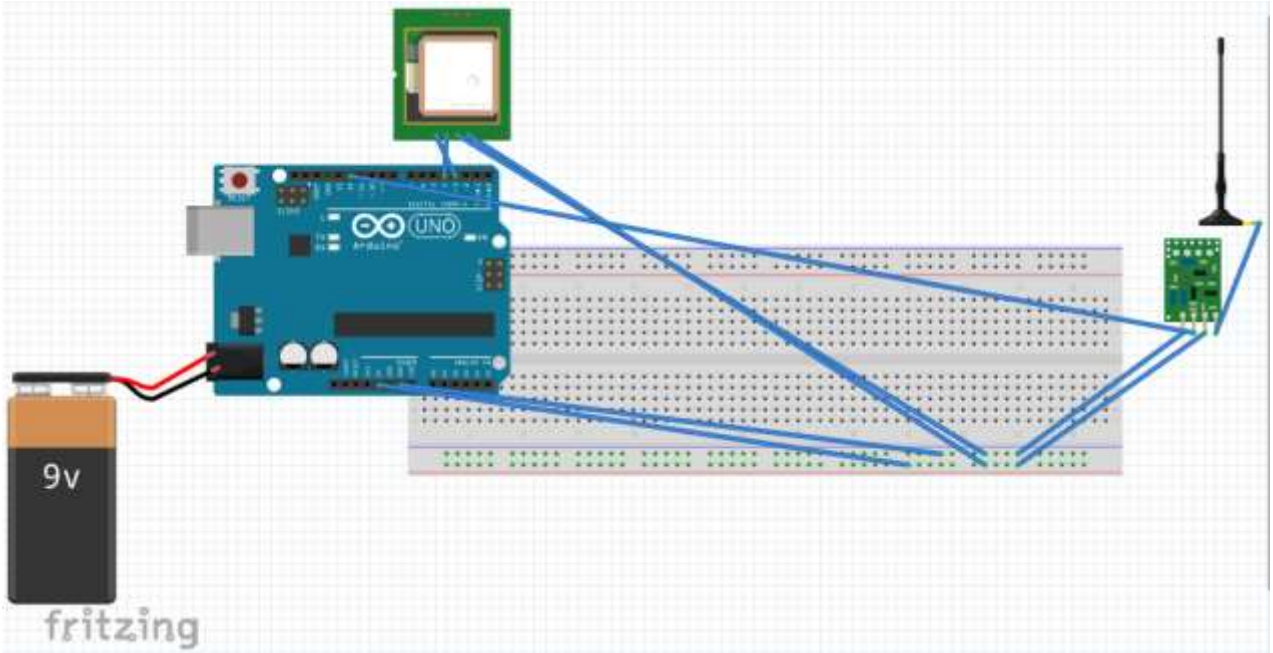

Figure 5.2.2. Schematic of the Transmitter using fritzing

This schematic represents our transmitter circuit. The components that are present here are the following:

- Arduino UNO board
- Neo 7m module (GPS)
- Rf transmitter (433 MHz)
- GPS Antenna
- Mini breadboard
- 5V battery
- Jumper wires

We connected the Neo 7m module to an external antenna for better connection and using 4 jumper wires we connected it to the Arduino Uno. Moreover, we connected the transmitter using the data pin to the pin number 12 of the Arduino. The 5V Battery shall act as a power supplier to the Arduino since it will be attached to the drone. Finally, we used a mini breadboard to be able to connect more inputs to the same pin (VCC-GND).

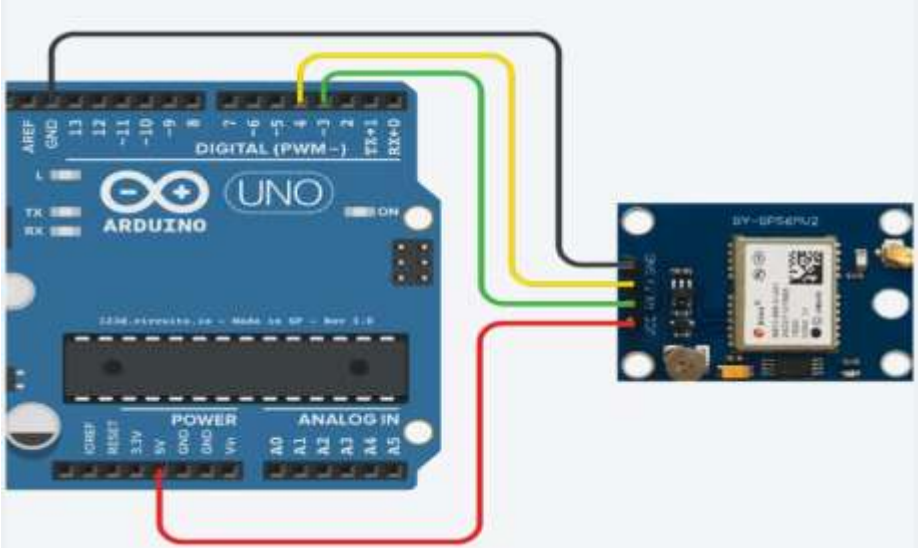

Figure 5.2.3. Neo 7m Connections.

Neo 7m module has 4 pins (VCC-Rx-Tx-GND):

- VCC: connected 5V (Red Wire)
- Rx: connected to pin 3 (Green Wire)
- Tx: connected to pin 4 (Yellow Wire)
- GND: connected to GND (Black Wire)

5.3 Schematic of the receiver circuit:

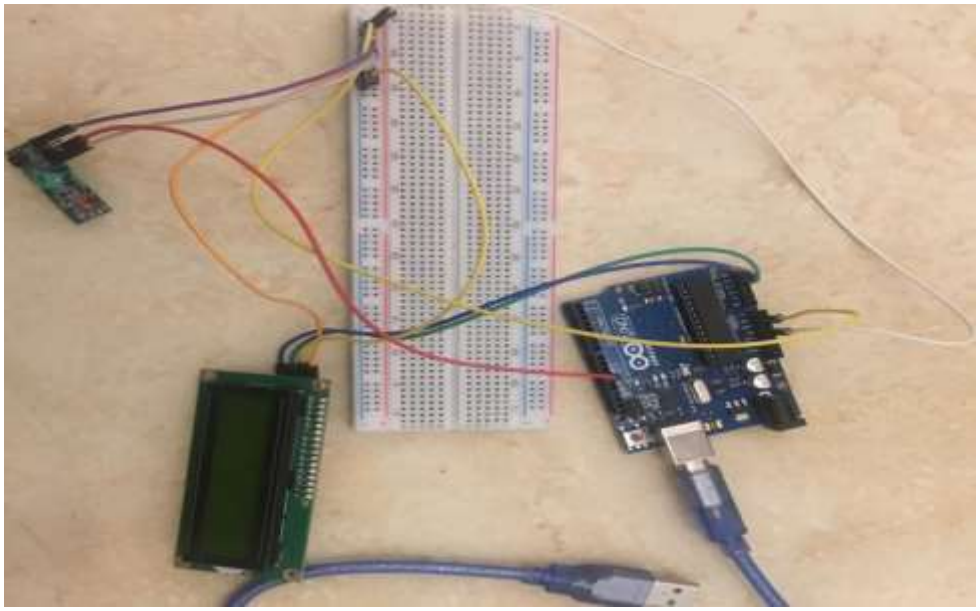

Figure 5.3.1. Schematic of the Receiver

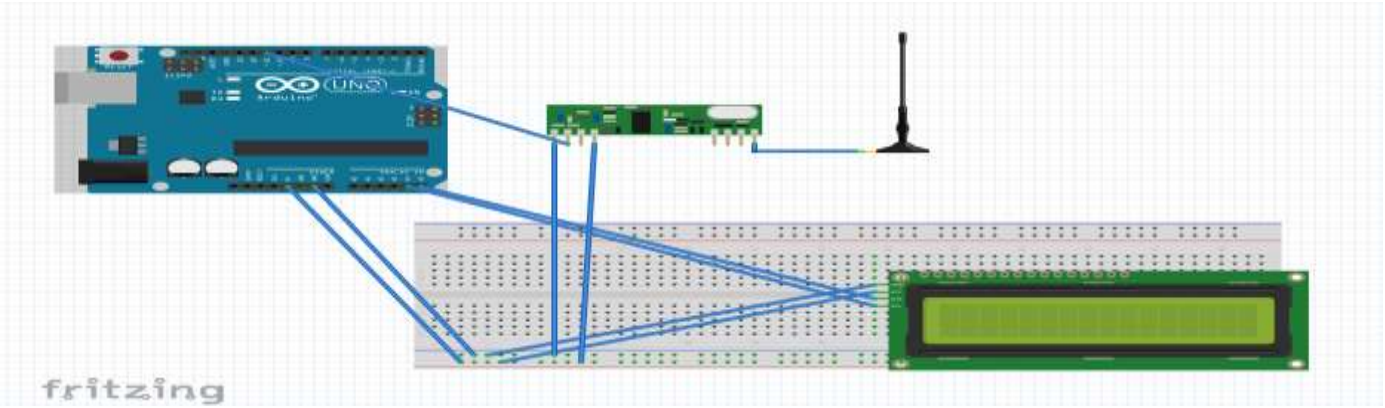

Figure 5.3.2. Schematic of the Receiver using fritzing

This schematic represents our receiver circuit. The components that are present here are the following:

- Arduino UNO board
- liquidCrystal (lcd)
- Rf receiver (433 MHz)
- Breadboard
- Jumper wires
- USB cable (connected to the laptop)

We connected the LCD using 2 jumper wires to the analog pins (A0-A1) of the Arduino and the other 2 pins (VCC-GND) were connected the breadboard. We used a breadboard to be able to connect more inputs to the same pin (VCC-GND). Finally, we connected the receiver using the data pin to the pin number 11 of the Arduino.

5.4 RF Transmitter & Receiver (433MHz):

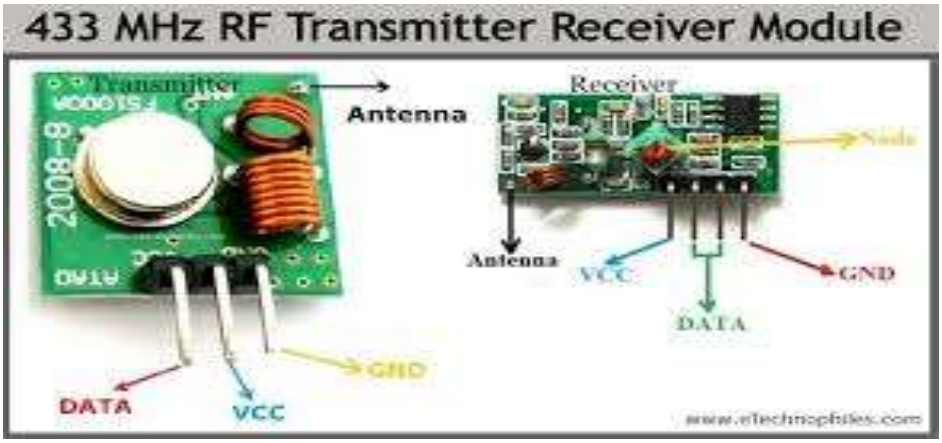

Figure 5.4.1. Transmitter Receiver Module

In figure 7, we can see the simple connections of the transmitter (3 pins →VCC-Data-GND) and receiver (4pins→VCC-2 Data-GND). These standard 433 MHz RF modules have a data rate of 1Kbps to 10Kbps. They can operate over a distance of 50 to 80 meters without any antenna. Moreover, we connected a 25cm single core antenna to the transmitter and a 32cm single core antenna to the receiver in order to increase the power output and the gain. This causes an increase in the range between the transmitter and receiver up to approximately 100 meters.

5.5 Flow Charts

Image Processing:

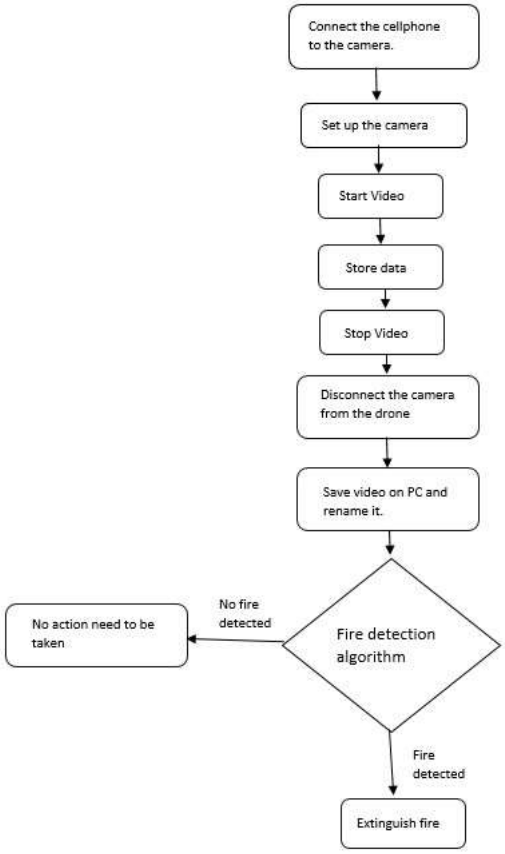

Figure 5.5.1 Simplified flow chart of the procedure till we do image processing

The flow chart in figure 5.5.1 shows the steps that we followed in order to be able to do the image processing.

First, we connect the cellphone to the camera, afterwards we set up the camera on the drone and, we start recording the video. At this point the camera is storing data. When we are done from exploring the area of the fire, we stop the video, we disconnect the camera from the drone and using a USB cable we copy the video from the camera to the PC.

Finally, the video is passed through the image processing algorithm if the algorithm detect fire than the fire fighter can extinguish the fire however if no fire is detected, no action need to be taken.

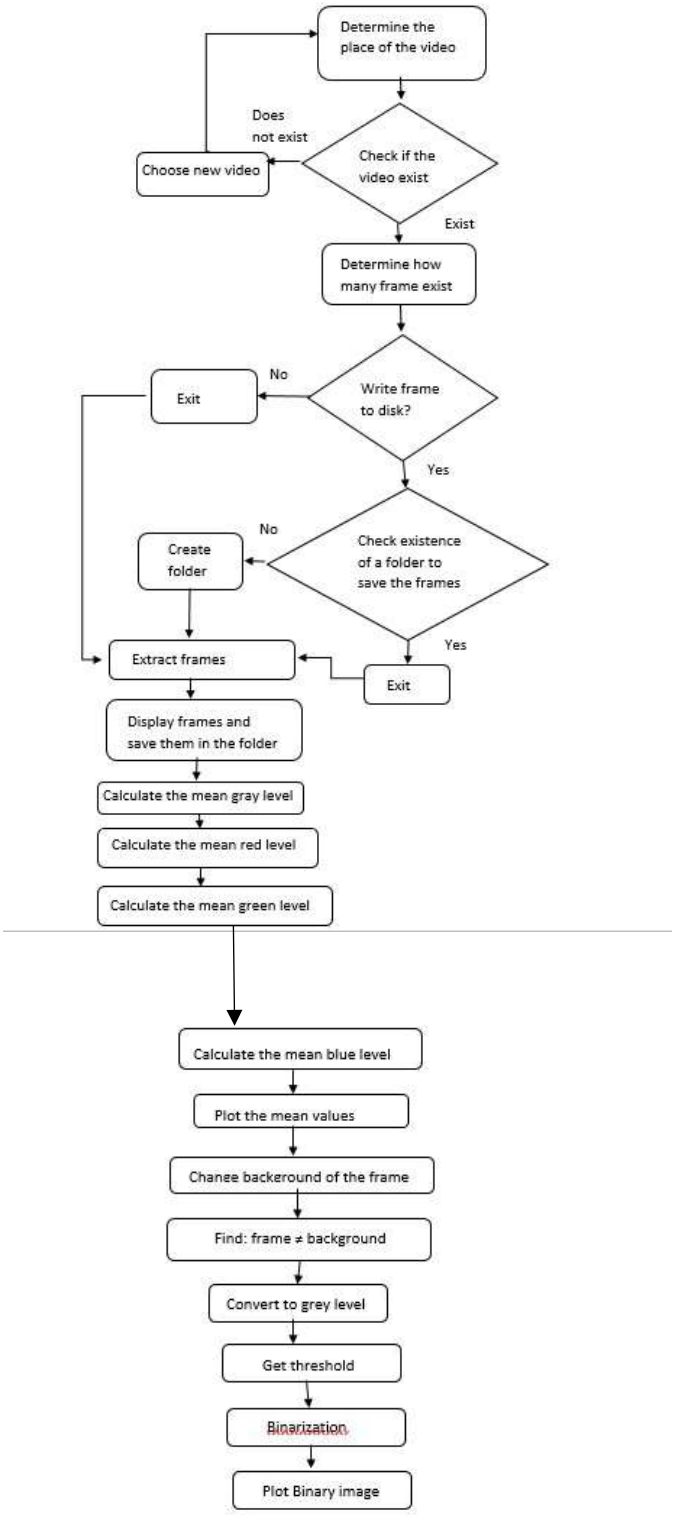

Figure 5.5.2. Image processing flow chart

The flow chart in figure 5.5.2 identify the essential steps and simultaneously offer a bigger picture of the image processing algorithm.

The image processing was done using Matlab functionalities. This Matlab code was originally written by Alex che[7], however many modifications have been done to the code in order for it meet our requirements.

First, Matlab determines the place of the video and checks if it exists or no. If the video does not exist, Matlab asks the user to choose another video else if the video exists, Matlab reads the video. Next, the software counts the number of frames in the video and asks the user whether he wants to save the frames on his PC or no. If the user does not want to save the frames on his PC then Matlab continues. However if the user wants to save the frames, Matlab searches for a folder in the directory of the image processing code to save the frames, if no folder exists it creates a new folder.

At this point, the software will go through the movie of the fire and will start writing the frames in the subfolder (in case the user requested to save the frames on the PC). Each frame will be saved as image separately from the other and each frame will have a unique name which is its number.

Subsequently, to do detect the fire Matlab calculates the mean gray level, the mean red level, the mean green level and the mean blue level of the frame. Then it plots all of them on the same figure.

Finally, Matlab do differencing by changing the background slightly at each frame and, calculate the difference between the frame and the background. Then it converts to gray level, gets threshold, do binarization, and plot the binary image.

Receiver flow chart:

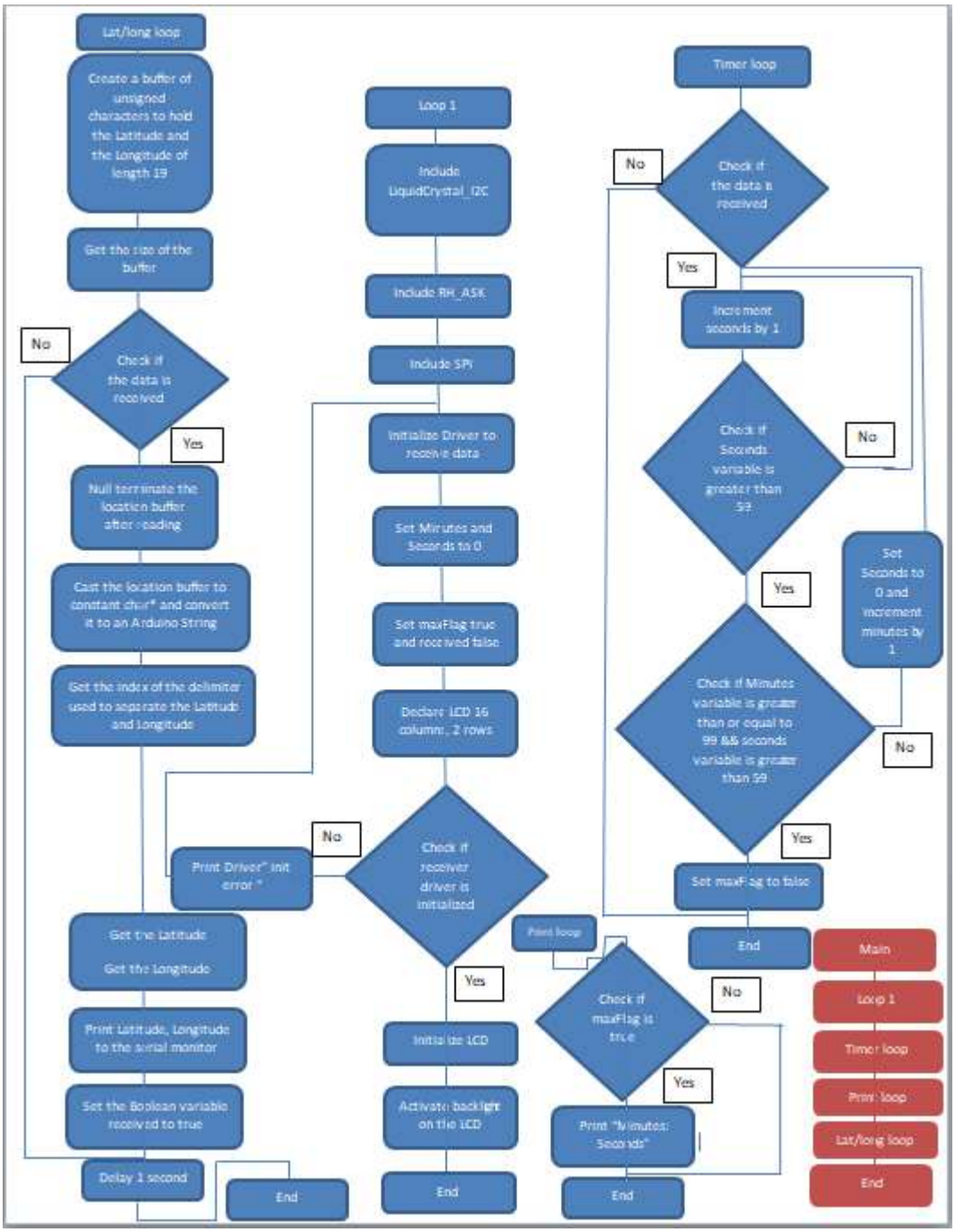

Figure 5.5.3 Receiver flow chart

Transmitter flow chart:

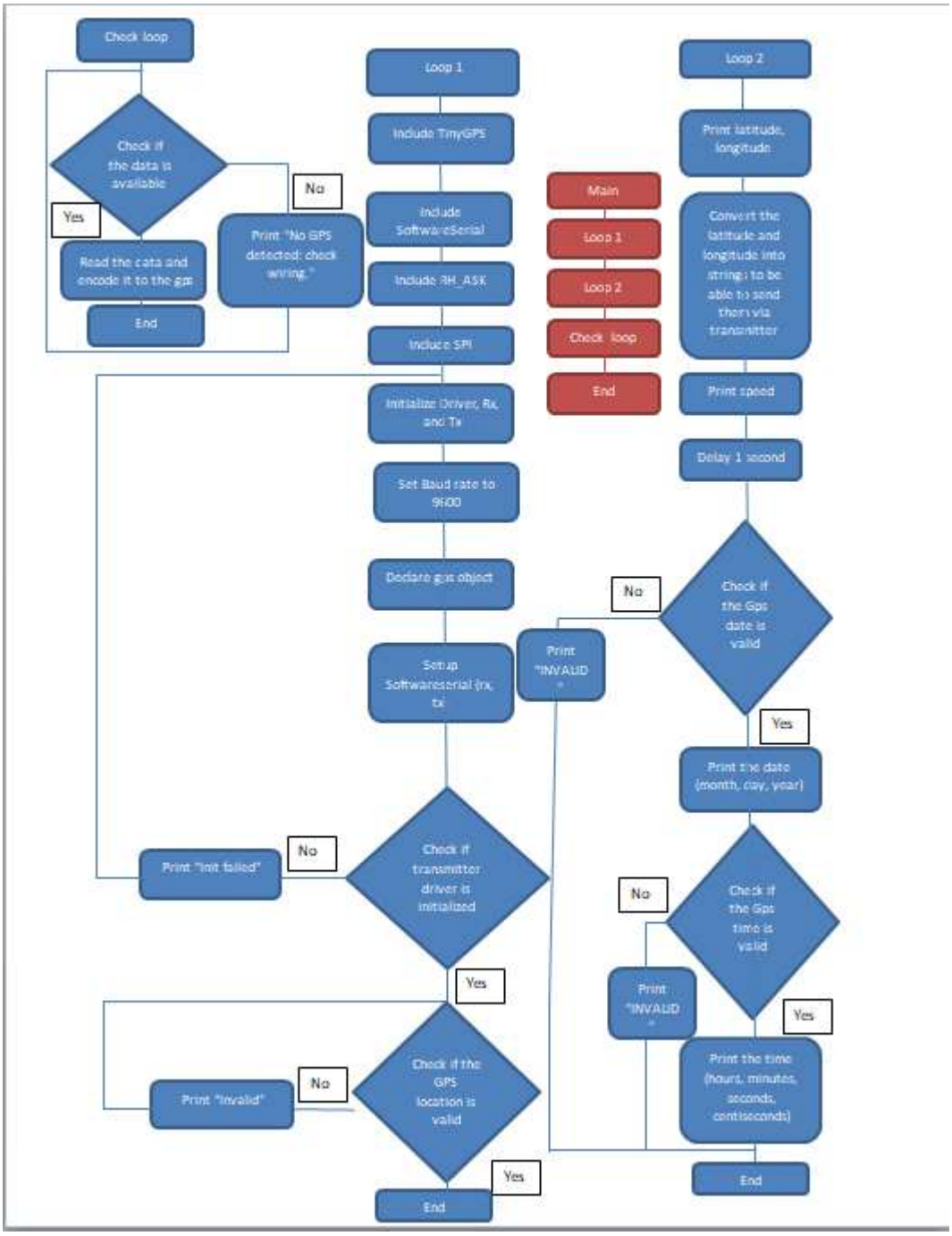

Figure 5.5.4 Transmitter flow chart

CHAPTER 6

SYSTEM TESTING AND FINALIZING

6.1 GPS

For this section, we are going to discuss the results of our work.

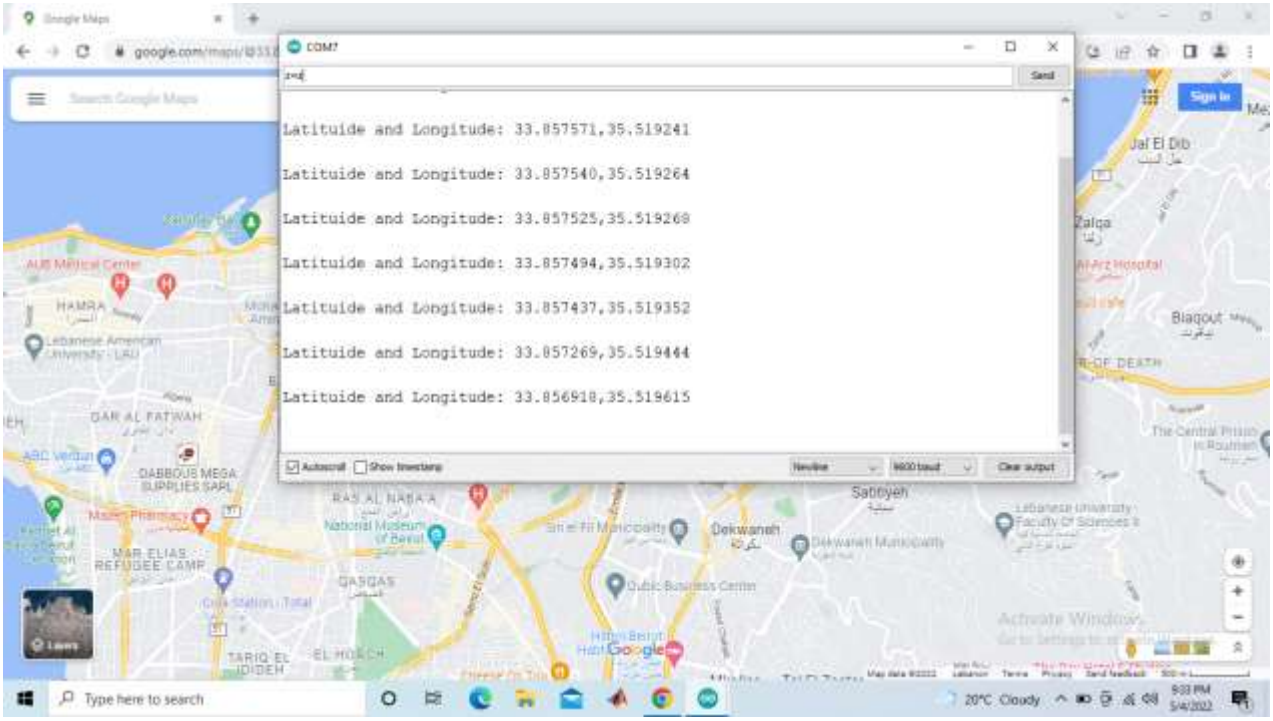

Figure 6.1.1. Latitude and Longitude of the Transmitter Circuit

Here, we are tracking the location of the drone in real time. Latitude and Longitude are used to determine the current location, and in this case, we are outputting the values in the Serial Monitor inside the receiver sketch.

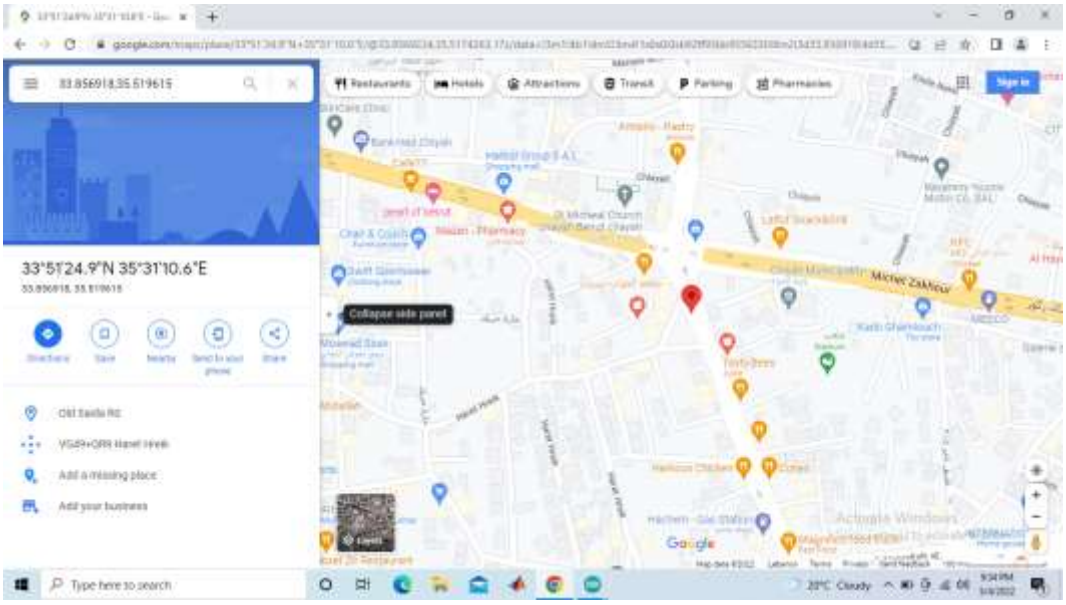

Figure 6.1.2. Drone Location in Google Maps.

In order to pinpoint the Drone's location, we are supposed to copy the results (Latitude, Longitude) into Google Maps' search bar. Once we submit the data, a red marker should display the location.

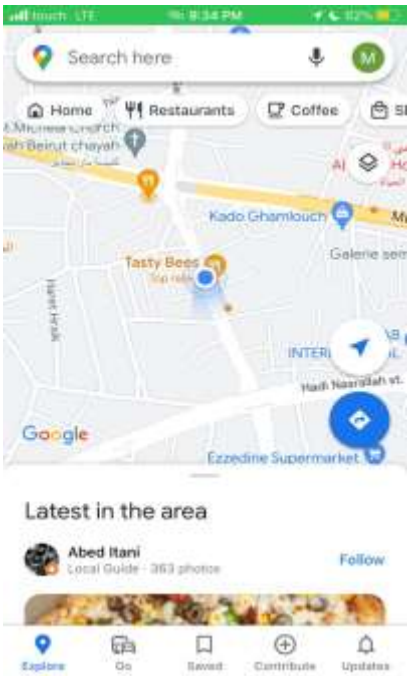

Figure 6.1.3. Receiver location.

This is the actual location from where we are monitoring all the process.

Note: We were about 10 meters far from the transmitter location.

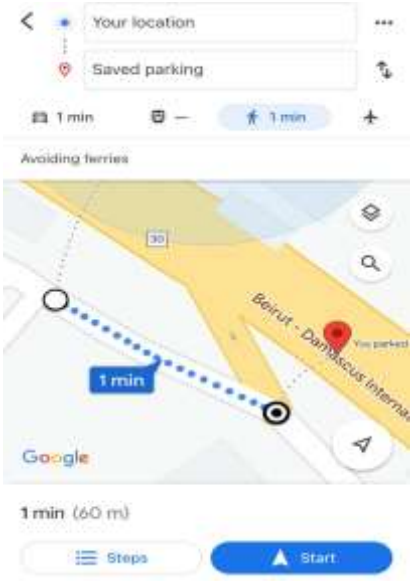

Figure 6.1.4. Receiver location.

Upon testing, we found that the maximum operating distance was 60 meters.

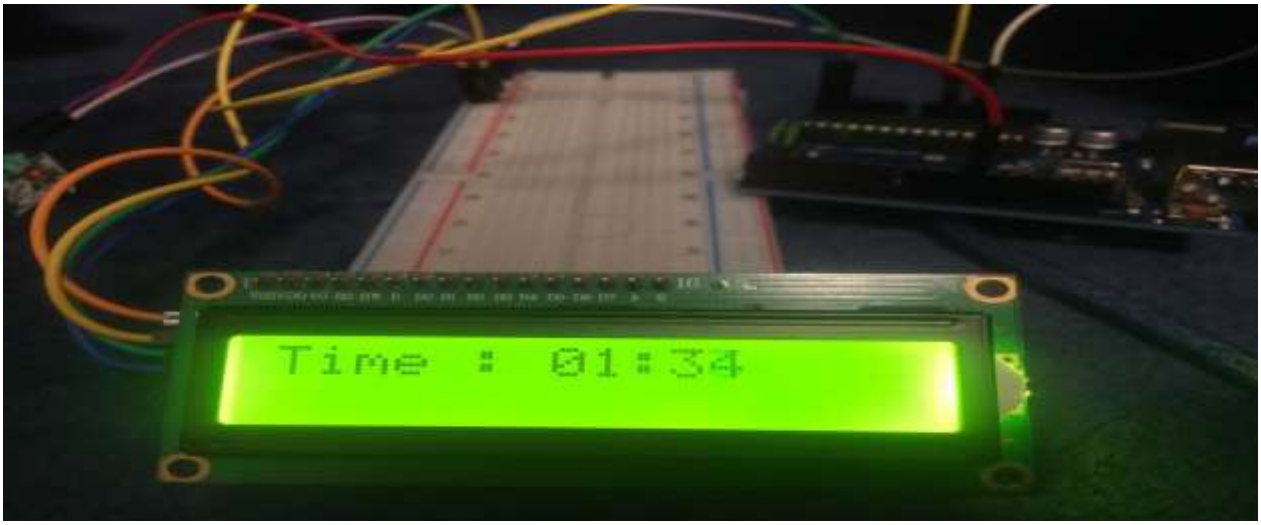

Figure 6.1.5. Timer

The timer starts as soon as the data gets received. We limited the timer up to 99 minutes and 59 seconds.

In case we want to relaunch the drone, the timer will be reset back to 0

We tried to receive the actual time from our Neo7m GPS, but because it is made in China, it will display China's time zone. Therefore, the only solution was to set a timer to approximate the time when a fire is detected.

6.2 Image Processing

The testing was done based on two criteria: first, the range, which refers to how far away the drone is from the fire, and second, the fire intensity.

We used film of a fire taken from a drone retrieved on the internet because we live in the city and there is no open place to start a small fire and do the tests.

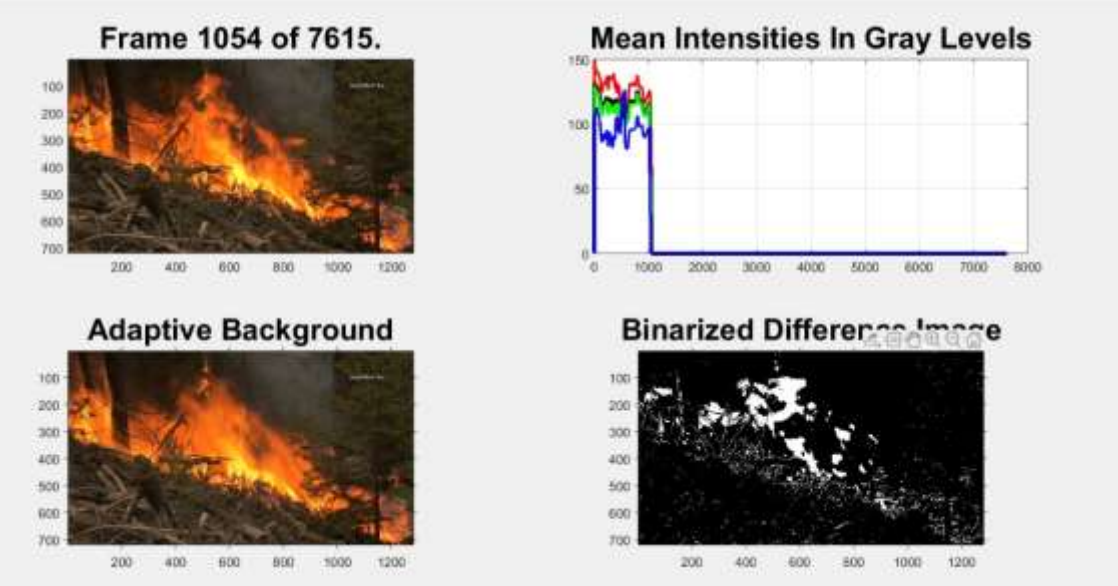

Figure 6.2.1. Drone close to fire

In figure 6.2.1 the drone was very close to the fire, the picture of the fire in the figure is extracted from a video that have 7615 frames and our example is about the frame1054.

There are a huge variation in the mean intensity in gray levels which indicate that a fire is detected. In addition, the adaptive background was used to calculate the difference between it and the frame in order to get finally image processing of the fire.

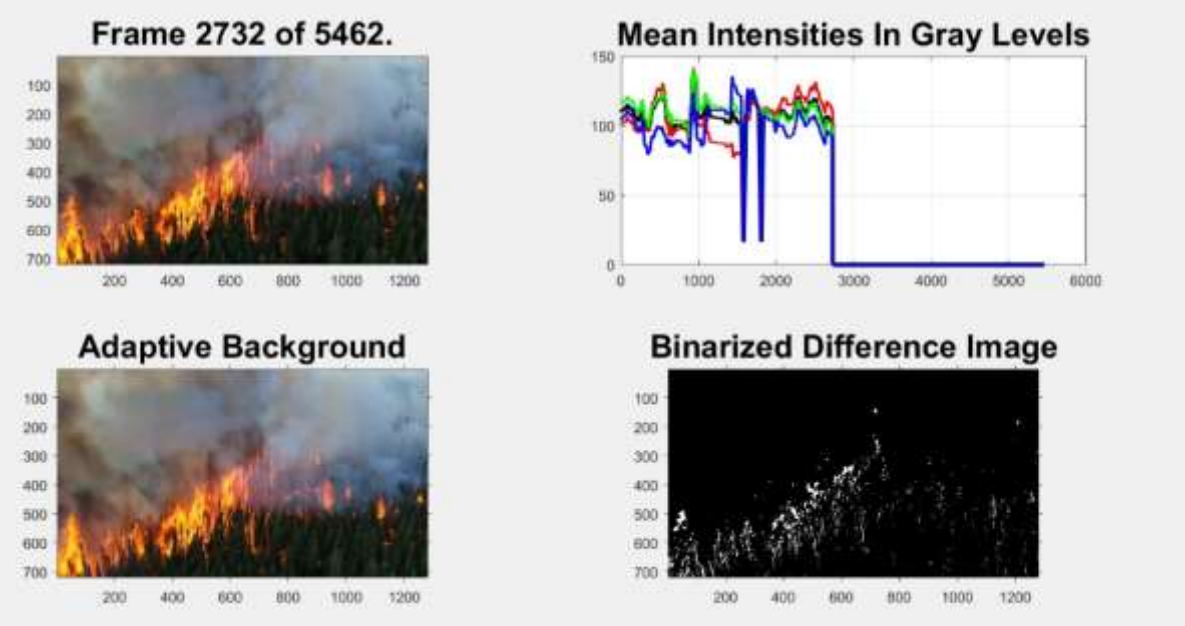

Figure 6.2.2 Drone far away from the fire

In figure 6.2.2 we have the same principle however the algorithm was simulated this time for a video taken far away from the drone, and it was a success.

CHAPTER 7

ACHIEVED REQUIREMENTS

Engineering Requirements	Justification	Outcome
The system should gather, store, analyze, and show data and information that are spatially related.	In order to detect fire, a lot of information and data should be collected.	We were able to collect data about the location of the drone.
The system should collect and process information via mobile wireless networks.	All data and information are going to be collected from a drone, the camera should send video of the incident wirelessly.	We were able to control the camera wirelessly.
The system should have the ability to cover many geographical area.	The coverage area depends on the battery capacity of the drone therefore the device will be improved in order to achieve this requirement.	The device was able to cover a 60-meter radius and pinpoint the drone's location.
The system should detect fire from a high altitude.	The strength of the fire varies depending on a variety of circumstances. Therefore, different types of fire should be detected.	We were able to detect fire far away from the drone.

CHAPTER 8

SUMMARY, CONCLUSION, AND FUTURE WORK

8.1 Summary

The goal is to implement an air-based fire detection system capable of detecting fire within few minutes with a range up to 60 meters.

The whole system is powered using batteries for automatic monitoring. The system is basically composed of a drone, camera, transmitter circuit and, a receiver circuit.

The transmitter circuit role is to get the location of the drone and send it to the receiver circuit; this circuit is situated on top of the drone and is enclosed in a casing to prevent damages while in flight.

The camera is insta360 Go2 attached to the drone, it is controlled wirelessly and it is used to take video of the scene, this video is later on image processed.

The receiver circuit role is to receive the location and other data from the transmitter circuit and it is connected to the PC.

8.2 Conclusion

This project is designed to allow us to look at a fire scene and detect the fire. The system is divided into two parts, one of which is connected to the drone and the other to the computer. During our testing, we were able to establish a link between the transmitter and receiver up to a distance of 60 meters, and we were able to interpret images of the fire from both a near and a far distance. This fire detection system is a promising prototype for minimizing fire's damages, with plenty of possibility for improvement in the future.

8.3 Future

Future fire detection prototypes hold a lot of promise and possibility for improvement. The following sections detail some of the hardware and software enhancements.

A hardware enhancement would be to use smaller components, not only for portability, but also to achieve the performance we are looking for and to pack more functionality into the same volume.

The software improvement would be to improve the image processing algorithm for it to detect every small detail of the fire. A second enhancement in the software side would be to increase the distance between the transmitter and receiver.

REFRENCES

[1] B. Ko and S. Kwak, “Survey of computer visionbased natural disaster warning systems,” Optical Engineering, vol. 51, no. 7, Article ID 070901, 2012

[2] P. M. Hanamaraddi, “A Literature Study on Image Processing for Forest Fire Detection,” IJITR, vol. 4, pp. 2695–2700, 2016.

[3] P. Podrzaj and H. Hashimoto, “Intelligent space as a framework ˇ for fre detection and evacuation,” Fire Technology, vol. 44, no. 1, pp. 65–76, 2008.

[4] RF transmitter and receiver circuit. Diagram. (n.d.). Retrieved May 6, 2022, from https://circuitdigest.com/electronic-circuits/rf-transmitter-and-receiver-circuit-diagram

[5] Project 012: Arduino Ublox Neo-7N GPS module project. Arduino Project Hub. (n.d.). Retrieved May 6, 2022, from https://create.arduino.cc/projecthub/infoelectorials/project-012-arduino-ublox-neo-7n-gps-module-project-d88500

[6] Pendergast, R. L., Santos, R., Dragon, Anderson, J. R., Javier, Dark, Iain, Kent, Louis, Marc, Trung, Ankit, Adekoya, S., Ramesh, Charley, pawar, S., Rafi, Firas, Lance, … D3nis.Dc. (2021, September 22). RF 433mhz transmitter/receiver module with Arduino. Random Nerd Tutorials. Retrieved May 6, 2022, from https://randomnerdtutorials.com/rf-433mhz-transmitter-receiver-module-with-arduino/

[7] *Fire detection using image processing*. Fire Detection using Image Processing -. (n.d.). Retrieved May 13, 2022, from https://www.mathworks.com/matlabcentral/answers/455793-fire-detection-using-image-processing

APPENDIX

Transmitter code:

```
#include <TinyGPS++.h> // Include the TinyGPS++ library

#include <SoftwareSerial.h> // Include SoftwareSerial to allow communication between RX and TX Pins.

#include <RH_ASK.h> // Driver Library to send data.

#include <SPI.h> // Needed to compile the code.

RH_ASK driver; // Initialize the driver

static const int RXPin = 4, TXPin = 3; // Initialize RX and TX variables on pins 4 and 3.

static const uint32_t GPSBaud = 9600; // Set GPS baud rate to 9600.

TinyGPSPlus gps; // Declare gps object.

SoftwareSerial ss(RXPin, TXPin); // Setup SoftwareSerial on Rx and Tx pins.

void setup() {

  Serial.begin(9600); // Open the serial port and set the data rate to 9600BPS

  ss.begin(GPSBaud);

  Serial.println(F("Drone Location"));

  if(!driver.init()){ // Check if transmitter driver is initialized.

    Serial.println(F("Init Failed"));

  } }

void displaySendGPSInfo()

{Serial.print(F("Location: "));

  if (gps.location.isValid()) // If the GPS Location is valid  {

    Serial.print(gps.location.lat(), 6); // Print Latitude with 6 digits after decimal
```

```
    Serial.print(F(","));

    Serial.print(gps.location.lng(), 6); //Print Longitude with 6 digits after decimal

    Serial.println();

    // Convert the Latitude and Longitude to Strings to be able to send them via the transmitter driver.

    String strLat = String(gps.location.lat(), 6);

    String strLng = String(gps.location.lng(), 6);

Serial.println("Speed \t" + String(gps.speed.kmph()) + " kmph\n\n");

    String loc = strLat + "|" + strLng; // Concatenate the Latitude and the Longitude with a special
delimiter character |

    // So we can split the data in the receiver.

    driver.send((uint8_t*)loc.c_str(), loc.length()); // Convert the string into a uint8_t* (unsigned
char*);

    // Blocks until the transmitter is no longer transmitting

    driver.waitPacketSent();

    delay(1000);  }

  else

  {// If the GPS Location is not valid, the print INVALID.

    Serial.print(F("INVALID"));  }

  Serial.print(F("  Date/Time: "));

  if (gps.date.isValid()) // If the GPS date is valid, then print the date(month, day, year);

  { Serial.print(gps.date.month());

    Serial.print(F("/"));

    Serial.print(gps.date.day());

    Serial.print(F("/"));

    Serial.print(gps.date.year());}
```

```
  else
  { // Print INVALID if the date is invalid
    Serial.print(F("INVALID")); }
  Serial.print(F(" "));
  if (gps.time.isValid()) // If GPS time is valid, then print the time(hours, minutes, seconds,
centiseconds);
  {if (gps.time.hour() < 10) Serial.print(F("0"));
    Serial.print(gps.time.hour());
    Serial.print(F(":"));
    if (gps.time.minute() < 10) Serial.print(F("0"));
    Serial.print(gps.time.minute());
    Serial.print(F(":"));
    if (gps.time.second() < 10) Serial.print(F("0"));
    Serial.print(gps.time.second());
    Serial.print(F("."));
    if (gps.time.centisecond() < 10) Serial.print(F("0"));
    Serial.print(gps.time.centisecond());}
  else
  { // Print INVALID if the GPS time is invalid.
    Serial.print(F("INVALID")); }
  Serial.println();}
void loop()
{ while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displaySendGPSInfo();
```

```
  if (millis() > 5000 && gps.charsProcessed() < 10) // If 5 seconds has passed and we don't have characters from gps

  { // Then there is a wiring error.

    Serial.println(F("No GPS detected: check wiring."));

    while(true);}}
```

Receiver code:

```
#include <LiquidCrystal_I2C.h> // Include the LiquidCrystal Library to support lcd

#include <RH_ASK.h> // Driver library to send and receive data

#include <SPI.h> // Needed to compile the code

RH_ASK driver; // Initalize driver to receive data

int minutes = 0, seconds = 0;

bool maxFlag = true; // Boolean flag to check if the timer is below the maximum time of 99:59

bool received = false;

LiquidCrystal_I2C lcd(0x27, 16, 2); // Declare lcd 16 columns, 2 rows

void setup() {

  // put your setup code here, to run once:

  Serial.begin(9600); // For debbugging

  if (!driver.init()) { // Check if driver is initialized

    Serial.println("Driver init error");  }

  lcd.init(); // Initialize LCD

  lcd.backlight(); // Activate backlight on the lcd}

void updateTimer() { if (received) {

    seconds++;

    if (seconds > 59) {
```

```
      if (minutes >= 99 && seconds > 59) {
        maxFlag = false; // Flip the flag to false if the timer is at maximum.}
      seconds = 0;
      minutes++;   }
    if (maxFlag) { // If maxFlag is true, meaning we are below the maximum time.
      lcd.clear(); // Clear lcd and print the minutes and seconds.
      lcd.print(F("Time : "));
      if (minutes < 10) lcd.print(F("0"));
      lcd.print(minutes);
      lcd.print(F(":"));
      if (seconds < 10) lcd.print(F("0"));
      lcd.print(seconds);
    }  }}
void loop() {
  uint8_t loc[19]; // Buffer of unsigned char* to hold  the Latitude and the Longitude of len 19
  uint8_t locLen = sizeof(loc); // Get the size of the buffer
  if (driver.recv(loc, & locLen)) {
    loc[locLen] = '\0'; // Null terminate the location buffer after reading
    String locStr = (const char * ) loc; // Cast the location buffer to const char* and convert it to an
arduino String
    int delimIndex = locStr.indexOf('|'); // Get the index of the delimiter used to separate the Latitude
and Longitude
    String latStr = locStr.substring(0, delimIndex); // Get the Latitude
    String lngStr = locStr.substring(delimIndex + 1, locStr.length()); // Get the Longitude
    Serial.println("Latituide and Longitude: " + latStr + "," + lngStr + "\n"); // Write the Lat and Lng to
the Serial
```

```
  received = true;  }

 delay(1000);

 // Minutes and seconds timer.

 updateTimer();}
```

Image Processing Code

```
%This matlab code aims to do image processing for fires

clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.
fontSize = 22;

% First we want to get the folder where the video is situated.
folder = fileparts(which('fire.mp4')); % Determine where demo folder
is.
movieFullFileName = fullfile(folder, 'fire.mp4');
% Now we are going to check to see that it exists.
if ~exist(movieFullFileName, 'file')
    strErrorMessage = sprintf('File not found:\n%s\nYou can choose a
new one, or cancel', movieFullFileName);
    response = questdlg(strErrorMessage, 'File not found', 'OK -
choose a new movie.','Cancel', 'OK - choose a new movie.');
    if strcmpi(response, 'OK - choose a new movie.')
        [baseFileNameNoExt, folderName, FilterIndex] =
uigetfile('*.mp4');
        if ~isequal(baseFileNameNoExt, 0)
            movieFullFileName = fullfile(folderName,
baseFileNameNoExt);
        else
            return;
        end
    else
        return;
    end
end

try
    videoObject = VideoReader(movieFullFileName)
    % Determine how many frames there are.
```

```
    numberOfFrames = videoObject.NumberOfFrames;
    vidHeight = videoObject.Height;
    vidWidth = videoObject.Width;

    numberOfFramesWritten = 0;
    % Prepare a figure to show the images in the upper half of the
screen.
    hFig = figure('Name', 'Video Demo by Image Analyst',
'NumberTitle', 'Off');
    hFig.WindowState = 'maximized'; %maximizing.

    % Ask user if they want to write the individual frames out to
disk.
    promptMessage = sprintf('Do you want to save the individual
frames out to individual disk files?');
    button = questdlg(promptMessage, 'Save individual frames?',
'Yes', 'No', 'Yes');
    if strcmp(button, 'Yes')
        writeToDisk = true;

        % Extract out the various parts of the filename.
        [folder, baseFileNameNoExt, extension] =
fileparts(movieFullFileName);
        % Make up a special new output subfolder for all the
separate
        % movie frames that we're going to extract and save to disk.
        folder = pwd;   % Make it a subfolder of the folder where
this m-file lives.
        outputFolder = sprintf('%s/Movie Frames from %s', folder,
baseFileNameNoExt);
        % Create the folder if it doesn't exist already.
        if ~exist(outputFolder, 'dir')
            mkdir(outputFolder);
        end
    else
        writeToDisk = false;
    end

    % Loop through the movie, writing all frames out.
    % Each frame will be in a separate file with unique name.
    meanGrayLevels = zeros(numberOfFrames, 1);
    meanRedLevels = zeros(numberOfFrames, 1);
    meanGreenLevels = zeros(numberOfFrames, 1);
    meanBlueLevels = zeros(numberOfFrames, 1);
    for frame = 1 : numberOfFrames
        % Extract the frame from the movie structure.
        thisFrame = read(videoObject, frame);

        % Display it
```

```
        hImage = subplot(2, 2, 1);
        image(thisFrame);
        caption = sprintf('Frame %4d of %d.', frame,
numberOfFrames);
        title(caption, 'FontSize', fontSize);
        axis('on', 'image'); % Show tick marks and get aspect ratio
correct.
        drawnow; % Force it to refresh the window.

        % Write the image array to the output file, if requested.
        if writeToDisk
            % Construct an output image file name.
            outputBaseFileName = sprintf('Frame %4.4d.png', frame);
            outputFullFileName = fullfile(outputFolder,
outputBaseFileName);

            % Stamp the name and frame number onto the image.
            text(5, 15, outputBaseFileName, 'FontSize', 20);
            frameWithText = getframe(gca);
            % frameWithText.cdata is the image with the text
            % actually written into the pixel values.
            % Write it out to disk.
            imwrite(frameWithText.cdata, outputFullFileName, 'png');
        end

        % Calculate the mean gray level.
        grayImage = rgb2gray(thisFrame);
        meanGrayLevels(frame) = mean(grayImage(:));

        % Calculate the mean R, G, and B levels.
        meanRedLevels(frame) = mean(mean(thisFrame(:, :, 1)));
        meanGreenLevels(frame) = mean(mean(thisFrame(:, :, 2)));
        meanBlueLevels(frame) = mean(mean(thisFrame(:, :, 3)));

        % Plot the mean gray levels.
        hPlot = subplot(2, 2, 2);
        hold off;
        plot(meanGrayLevels, 'k-', 'LineWidth', 3);
        hold on;
        plot(meanRedLevels, 'r-', 'LineWidth', 2);
        plot(meanGreenLevels, 'g-', 'LineWidth', 2);
        plot(meanBlueLevels, 'b-', 'LineWidth', 2);
        grid on;
        % Put title back because plot() erases the existing title.
        title('Mean Intensities In Gray Levels', 'FontSize',
fontSize);

        if frame == 1
            xlabel('Frame Number');
```

```
            ylabel('Gray Level');
            % Get size data later for preallocation if we read
            % the movie back in from disk.
            [rows, columns, numberOfColorChannels] =
size(thisFrame);
        end

        % Update user with the progress.  Display in the command
window.
        if writeToDisk
            progressIndication = sprintf('Wrote frame %4d of %d.',
frame, numberOfFrames);
        else
            progressIndication = sprintf('Processed frame %4d of
%d.', frame, numberOfFrames);
        end
        disp(progressIndication);
        % Increment frame count (should eventually = numberOfFrames
        % unless an error happens).
        numberOfFramesWritten = numberOfFramesWritten + 1;

        % Now let's do the differencing
        alpha = 0.5;
        if frame == 1
            Background = thisFrame;
        else
            % Change background slightly at each frame
            %           Background(t+1)=(1-alpha)*I+alpha*Background
            Background = (1-alpha)* thisFrame + alpha * Background;
        end
        % Display the changing/adapting background.
        subplot(2, 2, 3);
        imshow(Background);
        title('Adaptive Background', 'FontSize', fontSize);
        axis('on', 'image'); % Show tick marks and get aspect ratio
correct.
        % Calculate a difference between this frame and the
background.
        differenceImage = thisFrame - uint8(Background);
        % Threshold with Otsu method.
        grayImage = rgb2gray(differenceImage); % Convert to gray
level
        thresholdLevel = graythresh(grayImage); % Get threshold.
        binaryImage = im2bw( grayImage, thresholdLevel); % Do the
binarization
        % Plot the binary image.
        subplot(2, 2, 4);
        imshow(binaryImage);
        title('Binarized Difference Image', 'FontSize', fontSize);
```

```
        axis('on', 'image'); % Show tick marks and get aspect ratio
correct.
    end
    xlabel(hPlot, 'Frame Number', 'FontSize', fontSize);
    ylabel(hPlot, 'Gray Level', 'FontSize', fontSize);
    legend(hPlot, 'Overall Brightness', 'Red Channel', 'Green
Channel', 'Blue Channel', 'Location', 'Northwest');

    % Alert user that we're done.
    if writeToDisk
        finishedMessage = sprintf('Done!  It wrote %d frames to
folder\n"%s"', numberOfFramesWritten, outputFolder);
    else
        finishedMessage = sprintf('Done!  It processed %d frames
of\n"%s"', numberOfFramesWritten, movieFullFileName);
    end
    disp(finishedMessage); % Write to command window.
    uiwait(msgbox(finishedMessage)); % Also pop up a message box.

    % Exit if they didn't write any individual frames out to disk.
    if ~writeToDisk
        return;
    end



catch ME
    % Some error happened if you get here.
    strErrorMessage = sprintf('Error extracting movie frames
from:\n\n%s\n\nError: %s\n\n)', movieFullFileName, ME.message);
    uiwait(msgbox(strErrorMessage));
end
```