S.A.R.A

Search And Rescue Assistant

Software Engineering | Group 8

Date: Apr 14, 2019

https://abhiek187.github.io/emergency-response-drone/

TEAM MEMBERS

Sahana Asokan
Won Seok Chang
Avnish Patel
Abhishek Chaudhuri
Shantanu Ghosh
Srikrishnaraja Mahadas
Sri Sai Krishna Tottempudi
Vishal Venkateswara

Individual Contributions Breakdown

All team members contributed equally.

Table of Contents

Summary of Changes

- Updated the contributions breakdown matrix to indicate equality
- Reorganized references and labeled points of interest with numbers
- Added low battery alert to REQ4
- Added more detail about the drone and controller connection in REQ6
- Clarified useability in nonfunctional requirements
- Elaborated more on UC-6 and UC-7's casual description
- Indicated that UC-5 and UC-8 could be considered for future work
- Altered main success scenario for UC-6
- Removed fully-dressed description for UC-8
- Recalculated the Effort Estimation section
- Calculated duration in effort estimation
- Removed mathematical model for AvoidObstacles and added information about GetData
- Removed interaction diagram for UC-8
- Updated Project coordination and Progress report, so that it reflects recent work.
- Added the description for traceability matrix

Customer Problem Statement

Customer Statement of Requirements

Search and rescue operations can often involve first responders and volunteers trying to cover a vast area in as little time as possible to save the most lives. These operations can be categorized by the environment take place in. They can further be categorized by the specific type of operation that needs to take place, such as in urban areas or in remote mountainous regions. The circumstances that could merit such operations could involve natural disasters such as earthquakes and hurricanes. Regardless of the type of operation, technology is being increasingly used to streamline the efforts of first responders and volunteers in their efforts to try to save as many people as possible. There have been many search and rescue missions in the past.[1] Many of these missions involved the use of large amounts of people and resources. Even with all the effort put by the people involved many lives were lost in the process. One such organization that is involved in search and rescue operations is the Coast Guard. The table below illustrates the statistics of these operations conducted by the Coast Guard from 2011 through 2015.[2]

Fiscal Year	Cases	Responses	Sorties	Lives Saved	Lives Lost	Unaccounted Lives
2011	20,512	43,954	21,566	3,793	735	392

2012	19,787	43,940	21,609	4,037	713	440
2013	17,803	38,272	19,420	3,753	651	252
2014	17,508	38,282	19,032	3,443	595	308
2015	16,456	37,215	18,781	3,536	603	330

The sheer number of cases and responses conducted by the Coast Guard shows how big of an issue search and rescue missions are in the United States. The table also emphasizes that concept that these operations are not always successful or efficient. This is based on the number of lives lost and the number of people not accounted for along with a high number of responses for the cases. Our method looks into a possible alternative approach to these search and rescue missions.

The Search and Rescue Assistant, or S.A.R.A. will modernize the techniques employed by first responders on search and rescue operations. A drone can cover more distance than a single person is able to. Currently, the most frequently used techniques to cover a lot of areas very quickly is to either use a helicopter or to use a lot of people. The problem with helicopters is that they usually have to fly in from somewhere else and that can take time. Another problem with the use of helicopters is its lack of ability to search in narrow or tight areas. The issue with using a lot of volunteers is that people end up risking their own lives to find survivors. Often times these search parties tend to be time-consuming and depending on the circumstances, unorganized.[3]

Ready-to-launch drones can be set up in minutes, which will save time. By attaching a phone camera to the drone, the user will able to see the video feed that the drone is transmitting. Doing so will reduce the risk of potentially sending people in harm's way to get the most accurate information about where people might be trapped. Additionally, this would also be cost-efficient. This would reduce actual labor since we would mainly be investing in developing an efficient algorithm, and the device. This algorithm would take one initial investment and would be developed for improvement. Due to the cost effectiveness of the device and the reusability of the algorithms involved, if the resources were to be available, it should be rather simple to manufacture multiple devices.

Naturally, an important aspect of an aerial vehicle of this nature is whether or not it can survive the challenges/harsh conditions it can face while in the field. To bolster S.A.R.A's ability to withstand these conditions, it will be able to avoid obstructions in its' path, in part due to the implementation of ultrasonic sensors. Using these sensors, and the usage of the primary camera, the user can easily maneuver through different obstacles that he/she may encounter during a search and rescue mission. To assist the end-user in knowing the immediate environment, a thermal imaging attachment will be mounted to the mobile phone that serves as the drone's primary camera. Image processing will not occur on the drone itself, but instead on a centralized hub located back on an emergency vehicle, which receives relayed images/video real-time so that emergency responders can quickly determine the best course of action. The sensors/equipment necessary to accomplish this will be either be purchased or obtained by the team members from existing laboratories/organizations.

Regarding the working environment, S.A.R.A. will have to be able to maneuver in potentially tight/enclosed spaces. In such an environment, being able to receive data on how close an object is to the drone is a specialized function ultrasonic sensors can provide.

The drone can then properly take a course of action based on the proximity data it receives, such as change the amount of thrust in a particular direction or instead start pushing in an entirely different direction. With regards to processing visual data, the S.U.R.F. identification algorithm can be used to accurately determine an image's correlation/accuracy to a specific desired target object. In this case, the target would be the human faces/heat signatures.

Even with many solutions to search and rescue operations, S.A.R.A. offers a new take on optimizing the field. One of the key priorities of search and rescue missions includes safety, not just for the missing people, but for the people involved in the rescue operation. This approach makes it easy for even a single person to actively investigate the search and rescue operation in a safe manner. There would be more focus on the actual goal of the mission instead of also worrying about the safety of the people working the rescue/search missions.

Glossary of Terms

Database - Server that will keep data of the drone and pictures from the drone camera.

UI - A physical program that allows the user to see the environment from the camera, information of the drone speed and health, and distance away from the objects.

Controller - A device that will allow the user to control the drone movements and avoid any obstacles.

Proximity Alert - Internal mechanism that will use proximity sensors to see if the drone is getting dangerously close to any obstacles in the flight path.

Wireless Connection - The connection between the drone, controller, and database that allows the user to stay in control of the drone.

Drone Sensors - Devices that allow the drones to detect its speed, distance from user, stability, to detect obstacles, the drone's health, etc. Examples include an IR/Thermal sensor, Accelerometer, and Gyroscope.

S.U.R.F. - Speeded Up Robust Features, an algorithm that finds key points of an image using Hessian Matrices and scaled space, making it simpler to compare different images and see if they correlate appropriately.

System Requirements

Functional Requirements

REQ1 - Database/Server
REQ2 - UI Screen
REQ3 - Controller
REQ4 - viewDroneCondition
REQ5 - Proximity Alert
REQ6 - Wireless Connection
REQ7 - GPS tracking
REQ8 - Infrared Sensor

Requirement	Priority	Description
REQ1	5	Data server that will store the information from the drone and allow the user to access it
REQ2	2	The user interface will allow the user to see the drones footage and any other relevant information
REQ3	1	The user should be able to control the drone’s movements
REQ4	4	The drone will send a signal to the controller to notify of its operating status and alert the user if the battery is low.
REQ5	1	The drone should be able to correctly identify any close obstacles and be able to able to avoid them.
REQ6	2	A connection between the drone and controller is established via remote control.
REQ7	2	The user will be able to know exactly where the drone is.
REQ8	3	This sensor will allow the user to detect heat signatures through any material walls

Nonfunctional Requirements

Usability - User will figure out the user-friendly interface for controlling the drone by labeling buttons, displaying drone data, and requiring very little taps on the screen.

Security - User will be able to use the interface without having to jeopardize his/her safety by using the drone from a reasonable distance.

Accessibility - The user will be able to run the software to operate the drone, on any smartphone regardless of the OS on the device.

Efficiency - User will be able to use the software with any accompanying hardware through a wireless connection.

Recovery - User will be able to recover the drone if the signal is lost. If the drone software is program to auto course to controller and land back safely.

User Interface Requirements

Requirement	Priority	Description
REQ1	1	The controller for the drone will have a live feed of what the camera is seeing
REQ2	3	It will also display various properties of the drone. Some properties include speed of motors, drone battery level, and current location
REQ3	2	Proximity alerts will be sent to the controller so the user knows which direction to avoid
REQ4	4	The operating status of the drone will be sent to the controller so the user will know if they have to pull the drone back in case of low battery level.

Image 1[4]

Functional Requirements Specification

Stakeholders

Stakeholders

- Licensed User
- First Responders
    - Police Officers
    - Authorized Volunteers
    - Firefighters
    - EMT’s
    - Emergency Dispatchers

Actors and Goals

Actor	Type	Actor's Goal	Use Case Name
User	Initiating	To control the drone	MoveDrone (UC-1)
User	Initiating	To view a live video feed of the drone	ViewCamera (UC-2)
User	Initiating	To get the drone's current location.	GetLocation (UC-3)
Drone	Initiating	To check for and avoid obstacles.	CheckObstacles (UC-4), AvoidObstacles (UC-5), GetLocation (UC-3)
User	Initiating	To get the drone's operating status.	GetStatus (UC-6)
First Responder	Initiating	To identify the emergency from the drone.	ViewCamera (UC-2), GetStatus (UC-6)
Sensors	Participating	To locate nearby objects.	CheckObstacles (UC-4), AvoidObstacles (UC-5)
GPS	Participating	To track the current location of the drone.	GetLocation (UC-3)
Server	Participating	To store all of the data that the drone has obtained.	getData (UC-7), GetStatus (UC-6)
Drone	Participating	To return to home when the signal is lost for more than a certain time.	ReturnToHome (UC-8)

Use Cases

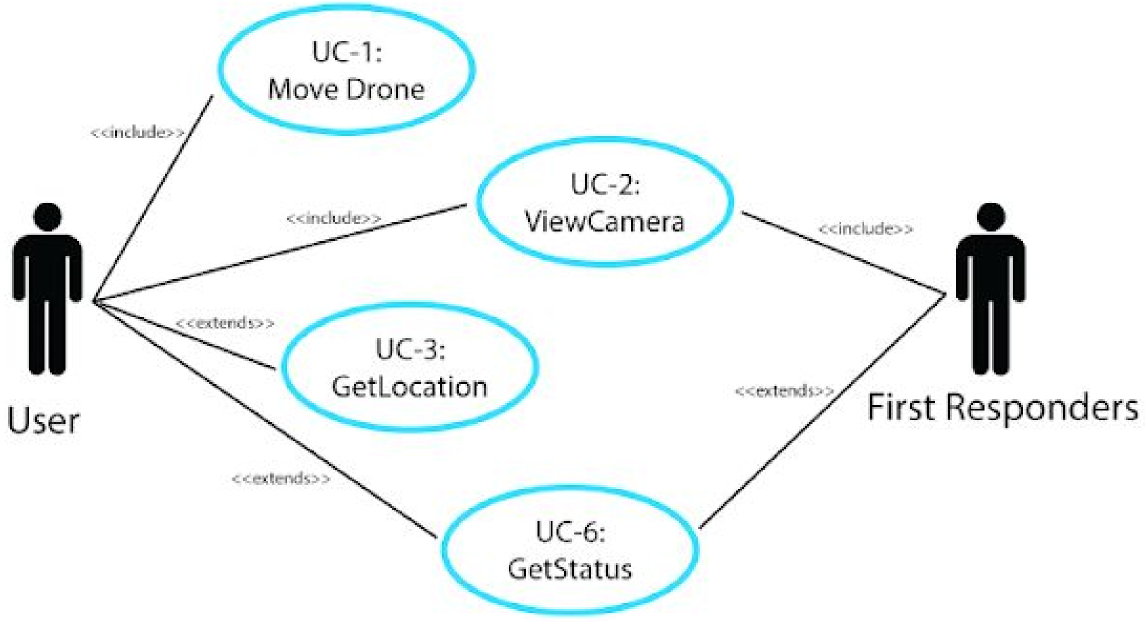
Casual Description

Use Case Name	Description	Requirements
MoveDrone (UC-1)	The user can move the drone using the controller.	REQ3, REQ6
ViewCamera (UC-2)	The user can view a video of the drone via the phone's camera.	REQ2
GetLocation (UC-3)	The user can detect the drone's location using GPS.	REQ2, REQ7
CheckObstacles (UC-4)	The drone can detect obstacles in its path.	REQ5, REQ8
AvoidObstacles (UC-5)*	The drone can avoid obstacles based on its surroundings.	REQ5, REQ8
GetStatus (UC-6)	The user or a first responder can check the current state of the drone, such as its power level or the phone's battery life.	REQ1, REQ2, REQ4
GetData (UC-7)	The user can check all of the data that the drone is transmitting through the sensors.	REQ1, REQ6
ReturnToHome (UC-8)*	The drone can safely autopilot back to the home (controller) in case the connection is lost. The user will know the drone's last location until it gets back.	REQ5, REQ7, REQ8

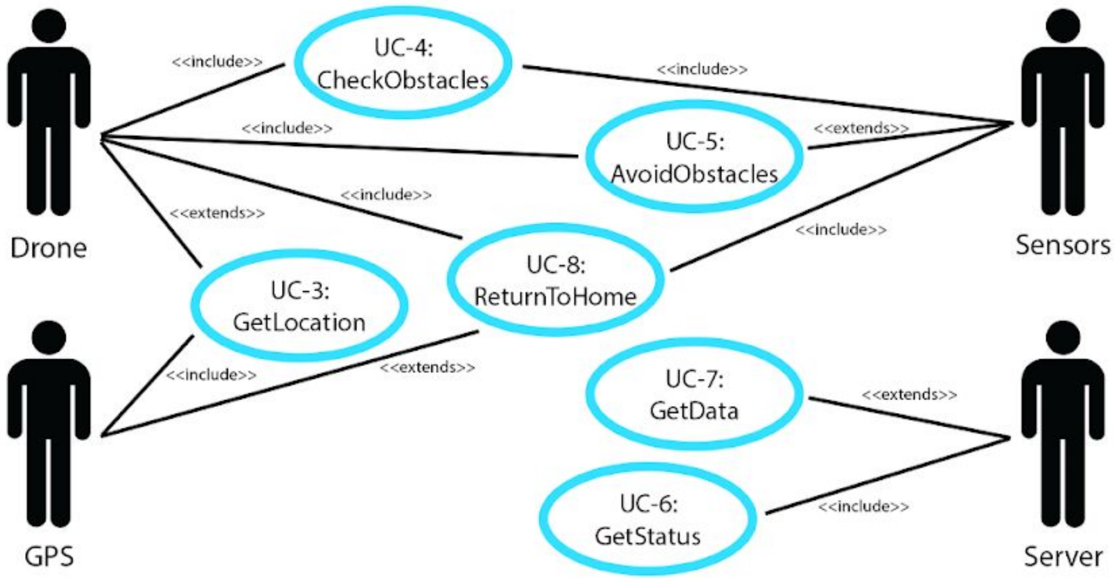
*This can be considered for future work.

Use Case Diagram

Use Case Diagram: Human Interaction



Use Case Diagram: Drone Operation



Traceability Matrix

Requirements	Priority	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
REQ1	5						x	x	
REQ2	2		x	x			x		
REQ3	3	x							
REQ4	4						x		
REQ5	1				x	x			x
REQ6	2	x						x	
REQ7	2			x					x
REQ8	3				x	x			x
Total Priority	-	5	2	4	4	4	11	7	6

The traceability matrix above shows the relationship between the use cases and the functional requirements. It also ranks each of the use cases based on which use cases wee believe have a higher priority. The matrix also ranks the priority of the requirements by what we think are the most important features for the drone to have. The way the matrix works is that each requirement has a set priority and if a use case incorporates a requirement; the priority points of the requirement are added to the use case. The priority points of each use case are the sum of the requirements for that use case. For example, use case 6 has a priority of 11 which comes from REQ1 (5), REQ2 (2), REQ4 (4). If the there values of the requirements are added together, you get the priority of use case 6 which is 11.

Fully-Dressed Description

<u>**Use Case 6:**</u>	GetStatus
Related Requirements:	REQ1, REQ2, REQ4
Initiating Actor:	Drone
Goal:	To get the drone's operating status
Participating Actor:	Server
Preconditions:	A signal between the drone and the controller is available
Postconditions:	Allows the user to know if the drone is active or not.
Main Success Scenario:	1. The user will know how much power the drone is using. 2. The user can see the phone's battery level and is alerted whenever it drops below 20%.

<u>**Use Case 7:**</u>	GetData
Related Requirements:	REQ1, REQ6
Initiating Actor:	User
Goal:	Collect data on the various operations of the drone
Participating Actor:	Server
Preconditions:	Drone is on and a connection between

the drone and controller is established.

Postconditions: Allows the user to manipulate and store that data.

Main Success Scenario:

1. The user can adjust motors speeds based on collected data.
2. The user uses the controller to move the drone if needed based on altitude.

<u>Use Case 1:</u> MoveDrone

Related Requirements: REQ3, REQ6

Initiating Actor: User

Goal: Ability to move the drone using a controller

Participating Actor: Drone, Controller

Preconditions: Drone is available
Controller is Available

Postconditions: Allows the user to maneuver the drone using a controller and a camera

Main Success Scenario:

1) The user sets the drone on the field.
2) The user uses the controller to test the drone's ability to move.
3) The controller will send signals to drone which will allow the user to control and move it.

<u>Use Case 3:</u> Get Location

Related Requirements: REQ2, REQ7

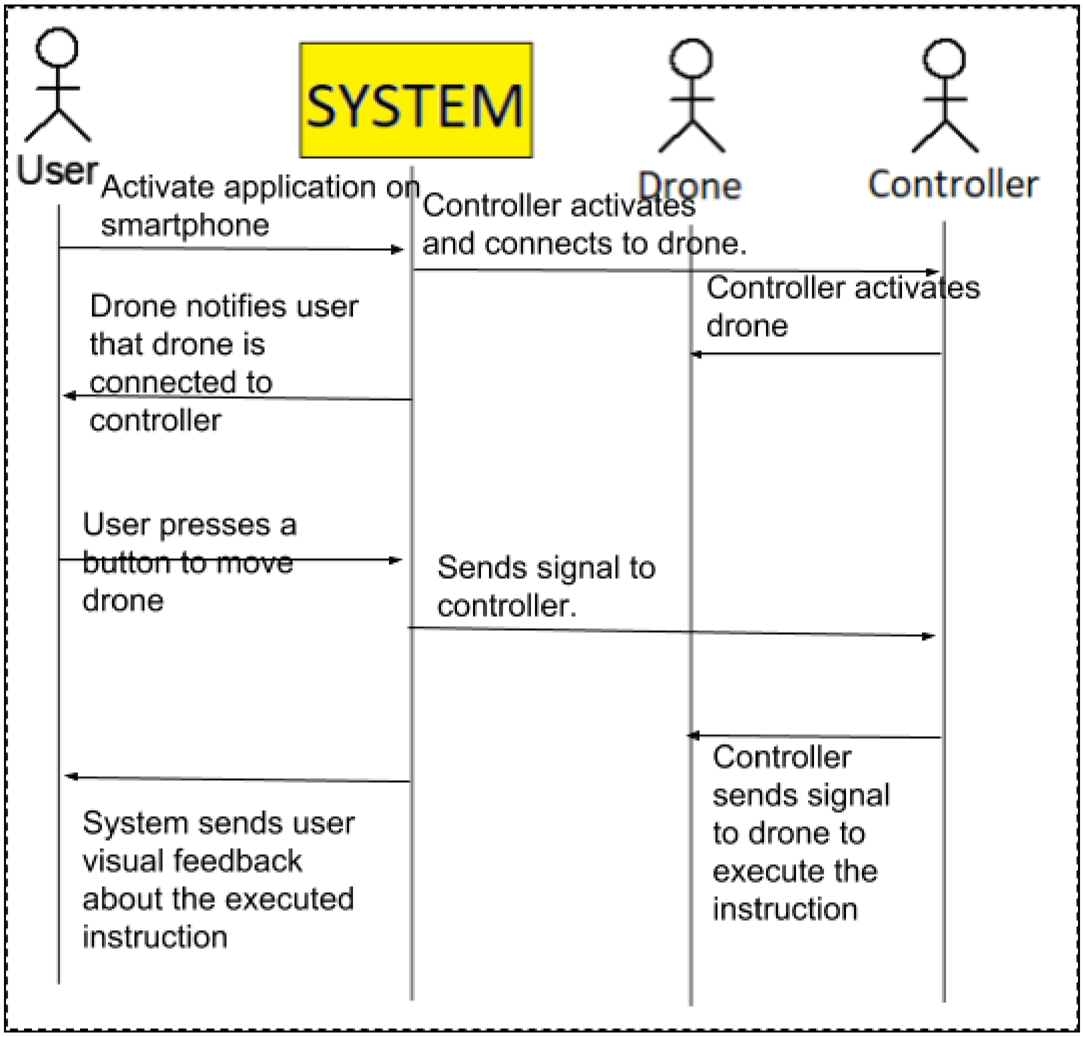
Initiating Actor:	User
Goal:	Ability to detect the current location of drone
Participating Actor:	Drone, Controller
Preconditions:	The GPS is on and in a working condition. The connection between the drone and controller is stable.
Postconditions:	Allows the user to retrieve the current location of drone displayed on the controller.
Main Success Scenario:	1)The controller receives the GPS signal from the drone. 2) The user can see the current location of drone.

<u>**Use Case 4:**</u>	CheckObstacles
Related Requirements:	REQ5, REQ8
Initiating Actor:	Drone
Goal:	To enable drone to detect obstacles in its path.
Participating Actor:	Sensors
Preconditions:	The sensors are on and in a working condition. The physical mechanism of drone is undamaged and operable.
Postconditions:	Allows the drone to detect obstacles that can possibly damage or interrupt its mission.

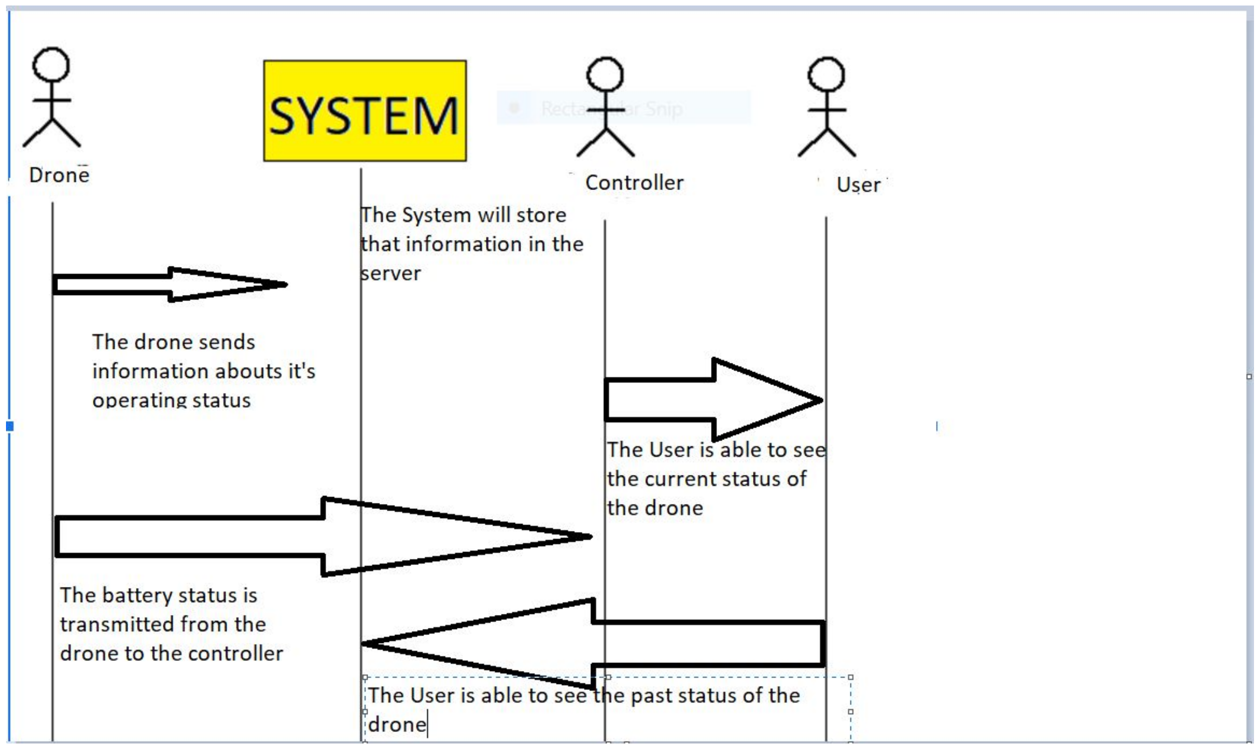
Main Success Scenario: 1)The sensors built on drone detect the obstacles.
2) It alerts the user, thus the user can maneuver the drone.

System Sequence Diagrams

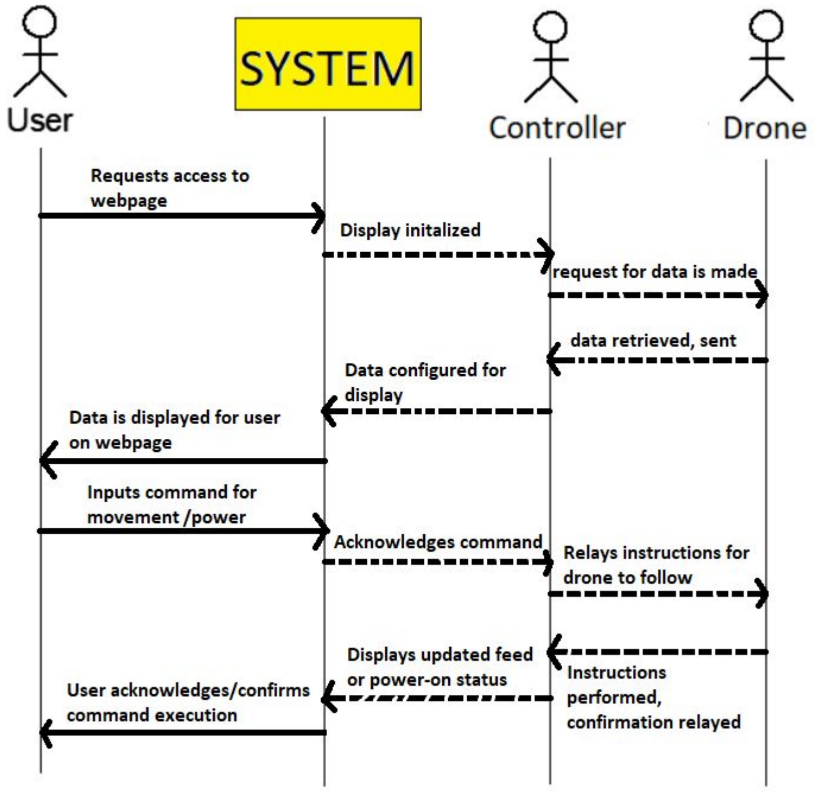
Use Case 1: MoveDrone


User
SYSTEM
Drone
Controller
Activate application on smartphone
Controller activates and connects to drone.
Controller activates drone
Drone notifies user that drone is connected to controller
User presses a button to move drone
Sends signal to controller.
Controller sends signal to drone to execute the instruction
System sends user visual feedback about the executed instruction

Use Case 6: GetStatus


SYSTEM
Drone
Controller
User
The System will store that information in the server
The drone sends information abouts it's operating status
The User is able to see the current status of the drone
The battery status is transmitted from the drone to the controller
The User is able to see the past status of the drone

Use Case 7: GetData


User
SYSTEM
Controller
Drone
Requests access to webpage
Display initalized
request for data is made
data retrieved, sent
Data configured for display
Data is displayed for user on webpage
Inputs command for movement /power
Acknowledges command
Relays instructions for drone to follow
Displays updated feed or power-on status
Instructions performed, confirmation relayed
User acknowledges/confirms command execution

Effort Estimation using Use Case Points

	Use case Points
UUCP=UUCW+UAW	108
TCF	.955
UCP = UUCP × TCF	103.14

UAW
Simple=1
Average =2
Complex = 3

Actor Name	Description of Relevant characteristics	Complexity	Weight
User	To control the drone	Complex	3
User	To view a live video feed of the drone	Complex	3
User	To get the drone's current location	Simple	1
Drone	To check for and avoid obstacles	Complex	3
User	To get the drone's operating status	Simple	1
First Responder	To identify the emergency from the drone	Simple	1

Sensors	To locate nearby objects	Average	2
GPS	To track the current location of the drone	Average	2
Server	To store all of the data that the drone has obtained	Average	2

UAW(home access) = _3 * Simple + 3_ * Average + 3_ * Complex = 18

UUCW
Simple=5
Average =10
Complex = 15

Use Case	Description	Category	Weight
MoveDrone (UC-1)	The user can move the drone using the controller.	complex	15
ViewCamera (UC-2)	The user can view a video of the drone.	complex	15
GetLocation (UC-3)	The user can detect the drone's location using GPS.	average	10
CheckObstacles (UC-4)	The drone can detect obstacles in its path.	complex	15
AvoidObstacles (UC-5)	The drone can avoid obstacles based on its surroundings.	complex	15
GetStatus (UC-6)	The user or a first responder can check the current state of the drone based on the	simple	5

	emergency.		
GetData (UC-7)	The user can check all of the data that the drone is transmitting.	complex	15

UUCW = 1_ *Simple + _1 * Average + _5 *Complex = 1x5+1x10+5x15= 90

TCF

Technical Factor	Description	Weight	Complexity	calculations
T1	Distributed web based System	2	5	10
T2	Performance objectives	2	3	6
T3	End-user efficiency	2	4	8
T4	Reusable code and design	1	2	2
T5	Easy to use	0.5	1	.5
T6	Moderately difficult to change	1	3	3
T7	Range of operation	1	3	3
T8	Signal Strength	1	3	3

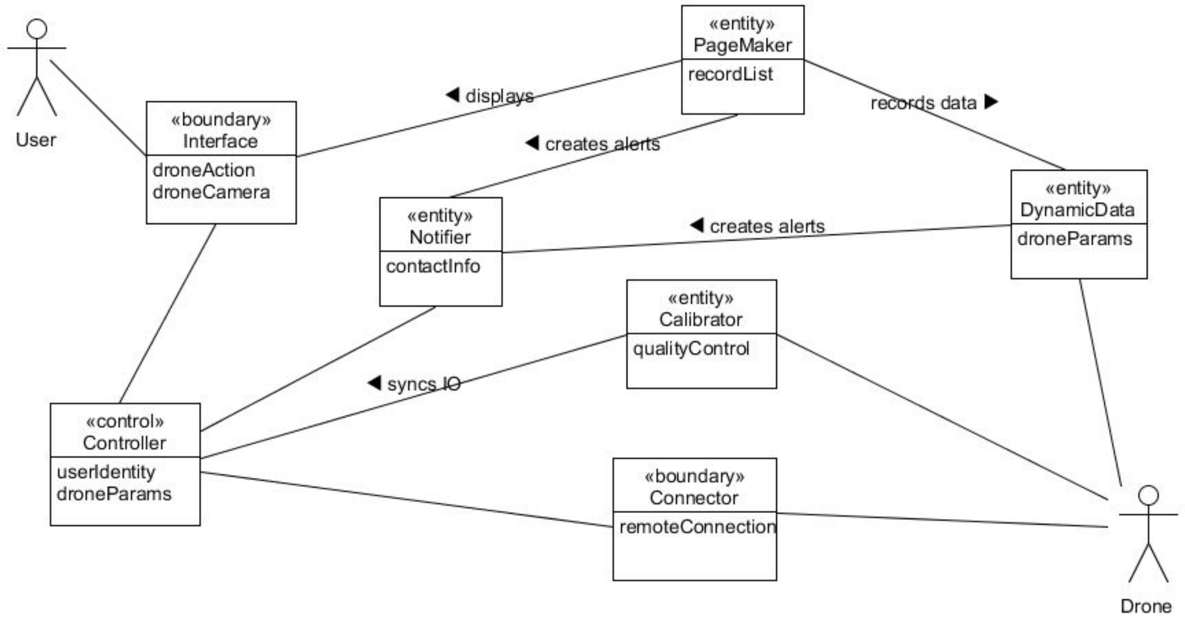
TCF=C1+C2x Technical Factor Total=35.5

$$C_1 + C_2 \cdot \sum_{i=1}^{13} W_i \cdot F_i$$

C1=0.6, C2=0.01, Technical Factor Total=
TCF= .955
Duration = UCP*PF = 103.14*28 = 2887.92

Domain Analysis

Domain Model



The domain model is derived from the concepts, attributes, and associations from all the use cases and requirements.

Concept Definitions

Rs#	**Responsibility Description**	**Type**	**Concept Name**
Rs1.	Coordinate the actions that the user wants the drone to take.	D	Controller
Rs2.	An HTML file that shows the user all the possible actions of the drone along with live camera feed.	K	Interface
Rs3.	Establishes a remote connection between the camera of the drone and the controller.	D	Connector
Rs4.	Renders the records onto an HTML file.	D	Page Maker
Rs5.	Calculate the speed, battery life and the location of the drone.	K	Dynamic Data
Rs6.	Notifies the user of potential issues such as low battery or obstacles.	D	Notifier
Rs7.	Makes adjustments to insure that the movements between the controller and drone are synchronized.	D	Calibrator

Association Definitions

Concept Pair	Association Description	Association name
Page Maker ↔ Interface	The Page Maker prepares the Interface	Display
Dynamic Data ↔ Notifier	The Dynamic Data informs the Notifier if there is any issue that the user needs to know.	Create Alerts
Calibrator ↔ Controller	The Calibrator makes sure that the inputs given by the user with the Controller are properly executed with minimal latency.	Sync IO
Page Maker ↔ Dynamic Data	The Page Maker records the different types of data in regards to the drone on an html file.	Record Data
Page Maker ↔ Notifier	If specific data record meets a certain alert condition, the user will need to know.	Creates Alerts.

Attribute Definitions

Concept	**Attributes**	**Attribute Description**
Controller	User's identity Drone parameters	Used to determine who will be controlling the drone and will handling the specific actions the user wants the drone to take. These actions are different so they will lie under different parameters.
Interface	Drone action list Drone camera feed	Used to show the user a physical view of all the actions available to the drone. Will show the user a physical view of the live camera.
Connector	Remote connection	Used to essentially connect the camera of the drone with the controller. So this will ensure everything is happening in live time and the user will be operating controls with an accurate visual.
Page Maker	Record list	Used to make sure the HTML file has a list of all the user records including action list.
Dynamic Data	Drone parameters	Will look for specific types of drone data such as speed, battery life and the location of the drone at all times.
Notifier	Contact information	Will contact user if there are any ter with the drone such as low battery, and physical obstacles.
Calibrator	Quality control	Quality will be ensured because of synchronization checks between drone and controller.

Traceability Matrix

		Domain Concepts						
Use Cases	Priority Weight	Interface	Controller	Connector	Page Make r	Dynamite Data	Notifier	Calibrato r
UC-1	5	x	x					x
UC-2	2	x			x			
UC-3	4			x		x		
UC-4	4						x	
UC-5	4		x					x
UC-6	11			x	x	x	x	
UC-7	7	x		x		x		x
UC-8	6		x				x	

System Operation Contracts

Operation	MoveDrone
Preconditions	● Drone is available ● Controller is available ● Application is open ● The physical mechanism of drone is undamaged and operable
Postconditions	● Allows the user to maneuver the drone using a controller and a camera ● Get visual feedback about the movement

Operation	GetLocation
Preconditions	● The GPS is on and in a working condition ● The connection between the drone and controller is stable
Postconditions	● Allows the user to retrieve the current location of drone displayed on the controller

Operation	CheckObstacles
Preconditions	● The sensors are on and in a working condition ● The physical mechanism of drone is undamaged and operable
Postconditions	● Allows the drone to detect obstacles that can possibly damage or interrupt its mission using the sensors.

Operation	**GetStatus**
Preconditions	● A signal between the drone and the controller is available
Postconditions	● Allows the user to know if the drone is active or not

Operation	**GetData**
Preconditions	● Drone is on and a connection between the drone and controller is established
Postconditions	● Allows the user to manipulate and store that data. ● Get webpage of data

Operation	**ReturnToHome**
Preconditions	● Drone is active ● Controller is available ● The physical mechanism of drone is undamaged and operable
Postconditions	● Allows the user to acquire the drone and turn off and pack up the drone

Mathematical Model

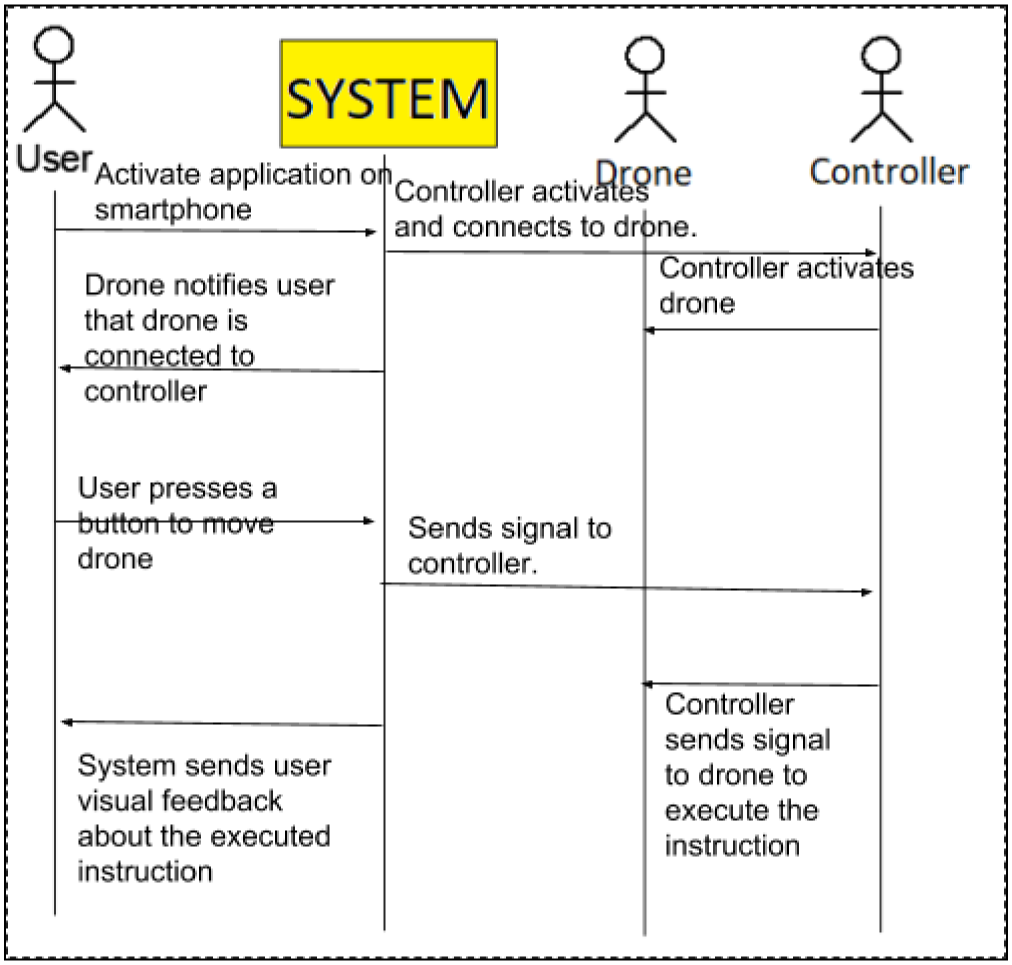
The drone will be calibrated to use math in order to move and avoid obstacles. This model is correlated with UC-1, UC-5, and UC-7.

UC-1 (MoveDrone): A controller is used to move the drone. The user can control to rotate, move forward/backward, throttle, and strafe the drone. This will involve adjusting the speed of motors to change the velocity and angle of the drone.

UC-7 (GetData): The user will be able to retrieve the battery level, speed, and position of the drone. These values will be updated in real time via the drone's wireless connection. The speed is calculated by taking the latitude and longitude, and when it updates, calculate the distance divided by the time it took to update.

Interaction Diagrams and Design Principles

Use Case 1: MoveDrone


SYSTEM
User
Drone
Controller
Activate application on smartphone
Controller activates and connects to drone.
Controller activates drone
Drone notifies user that drone is connected to controller
User presses a button to move drone
Sends signal to controller.
Controller sends signal to drone to execute the instruction
System sends user visual feedback about the executed instruction

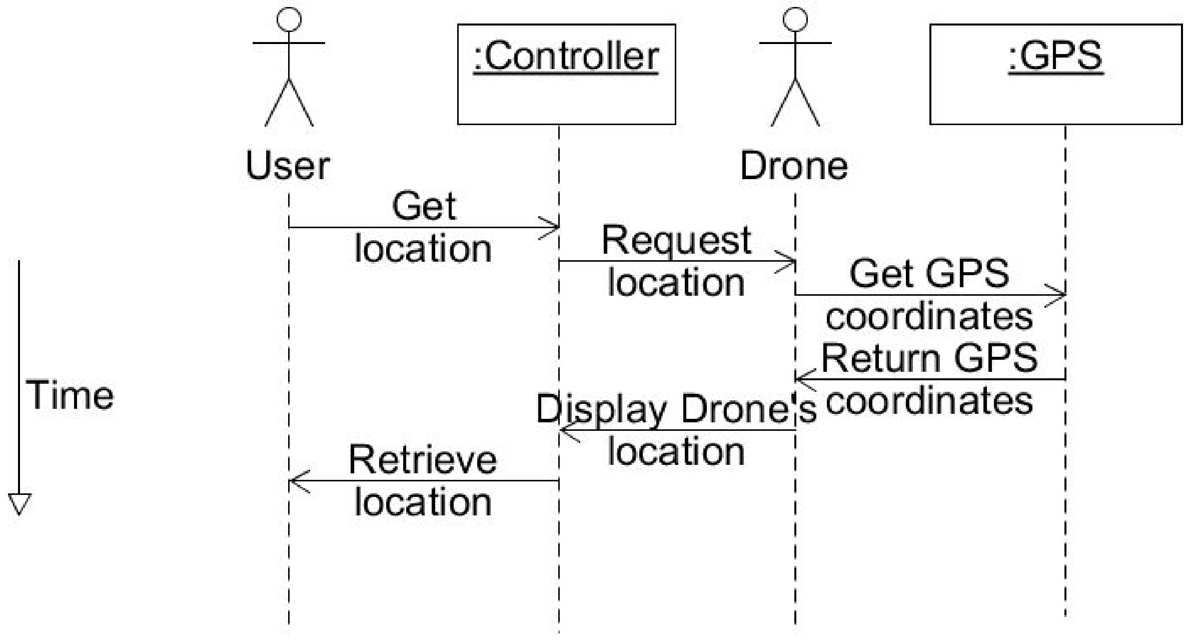
The diagram for the first use case is displayed above. In this case, the User will first activate the system in order to gain access to the controller. From there the User is able to use the controller to move the drone. The User will be able to see visual feedback from the drone.

Design Principles:

The design principles utilized in this use case include the Low Coupling Principle. This design principle is utilized as the communication links that exist are very short. Most of the communication is done between the User and controller, and then the controller and drone.

In this use case diagram the controller could act as the publisher while the drone will act as the subscriber. The user utilizes the controller to send events to the drone which will work if it has received a valid event.

Use Case 3: GetLocation

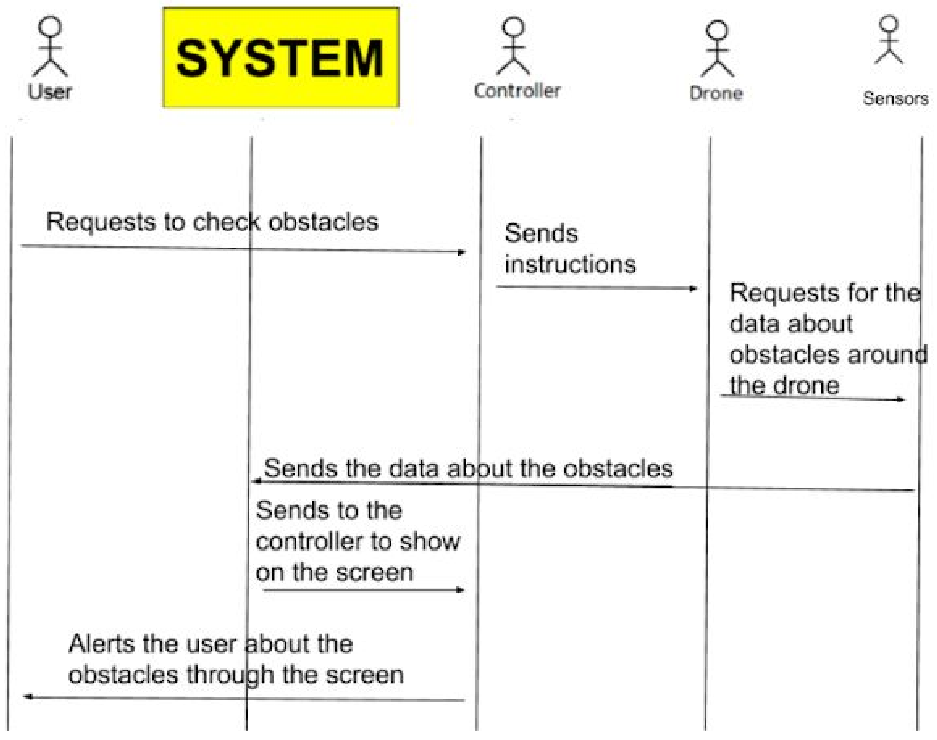


The diagram above demonstrates the interactions between classes in UC-3: get the location. Once the user has control of the drone and being able to maneuver around obstacles. First, the user sends a request to get the location of the drone to the controller which then gets requests it to the drone. The drone then requests the coordinates to the GPS server and gets the coordinates and sends it back to the controller to make it visible. The user sees the drone's location based on longitude and latitude.

Design Principles:
The design principles employed in the process of assigning responsibilities were the expert doer principle and high cohesion principle. The expert doer principle is used because each of the classes is an expert for specific functions. An example, the drone is responsible for getting the coordinates from the GPS server and relaying it back to the controller for the user to see.

This use case diagram will also work with the publisher-subscriber design pattern. The controller will act as the publisher while the drone and the GPS act as the subscribers. The controller will send a request to update the current gps location of the drone to the GPS, which in turn will verify the location of the drone and then return the coordinates.

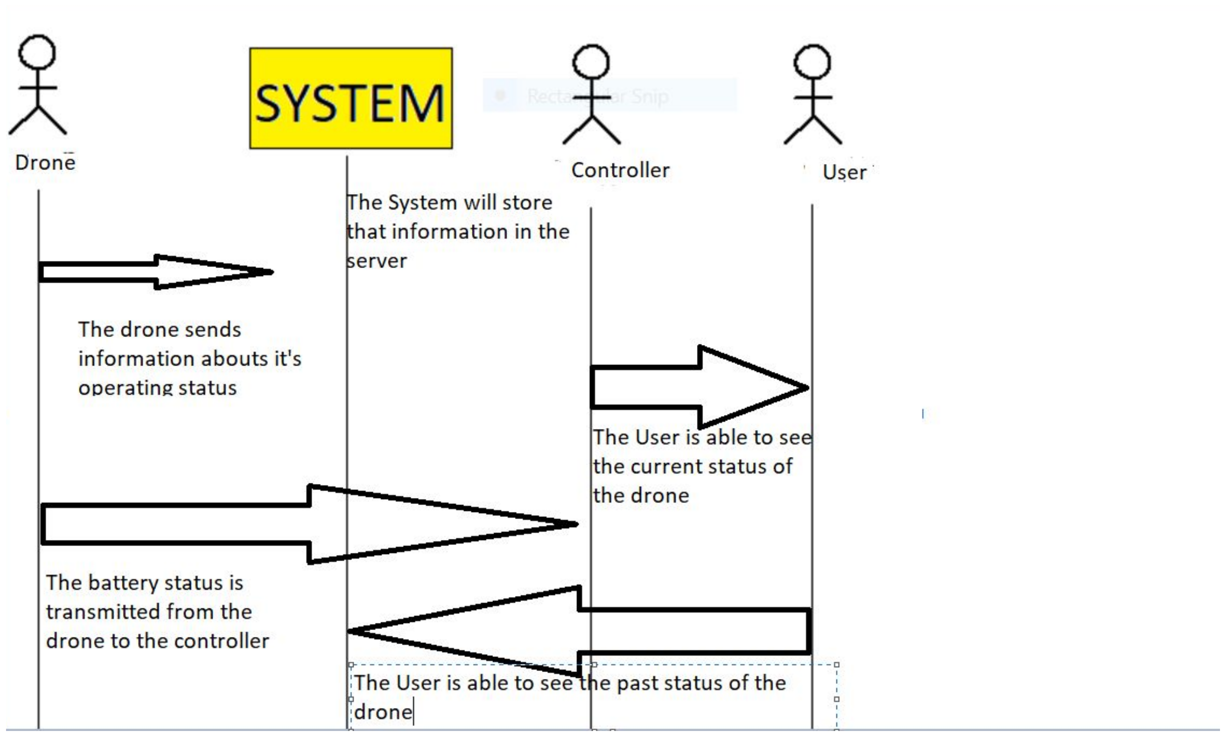
Use Case 4 : CheckObstacles



SYSTEM
User
Controller
Drone
Sensors
Requests to check obstacles
Sends instructions
Requests for the data about obstacles around the drone
Sends the data about the obstacles
Sends to the controller to show on the screen
Alerts the user about the obstacles through the screen

Design Principle:
The design principle for this use case is the expert doer principle and high cohesion principle because the parameters for the obstacle is super specific. And that data should be focused on because it can affect the overall behavior of the drone. It is also important that the specific obstacles that are being checked for are being communicated to other sources.

Use Case 6: GetStatus

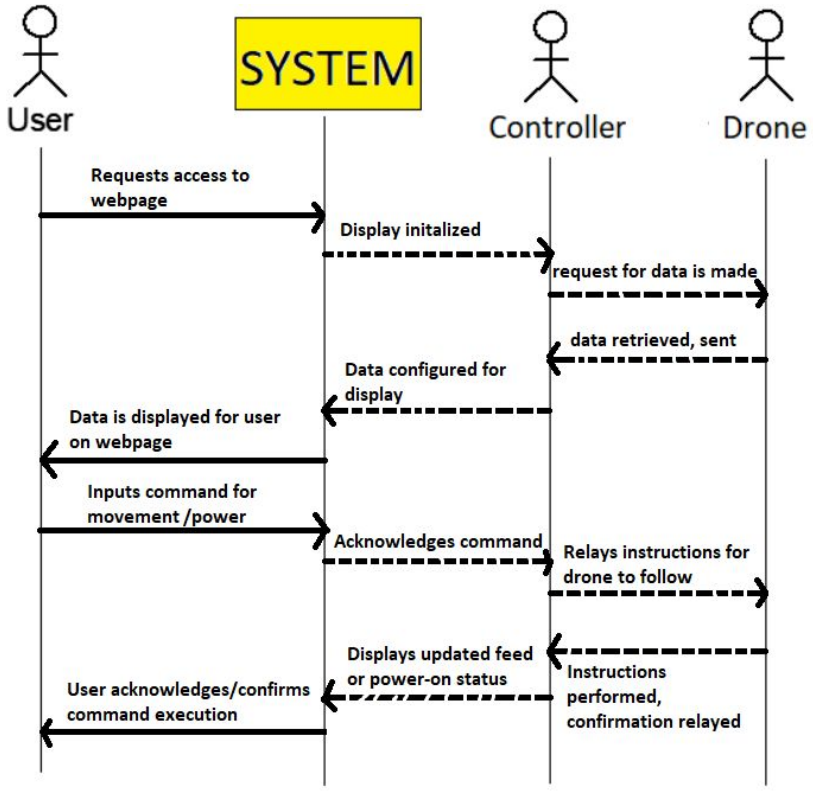

Drone
SYSTEM
Controller
User
The System will store that information in the server
The drone sends information abouts it's operating status
The User is able to see the current status of the drone
The battery status is transmitted from the drone to the controller
The User is able to see the past status of the drone

The interaction diagram for use case 6 is displayed above. The drone basically sends a signal to the system which the user can see the result of through the controller. The part of this use case is to let the user know of the operation status of the drone, in particular, the battery level. The user is also able to view the past values sent by the drone.

Design Principles:
The design principles utilized by this use case are Expert Doer Principle, High Cohesion Principle, and the Low Coupling Principle. Since this use case is the only use case that knows about the battery status it makes sense that the Expert Doer Principle is used. As

for the High Cohesion Principle, the only computation done by this part is the battery level. The Low Coupling Principle deals with the concept that this use case does minor communication between the drone and the controller.

Use Case 7: GetData



User
SYSTEM
Controller
Drone
Requests access to webpage
Display initalized
request for data is made
data retrieved, sent
Data configured for display
Data is displayed for user on webpage
Inputs command for movement /power
Acknowledges command
Relays instructions for drone to follow
Displays updated feed or power-on status
Instructions performed, confirmation relayed
User acknowledges/confirms command execution

The diagram above demonstrates how the user, controller, and drone interact with each other to show necessary data of drone to the user, so the user can control the drone. When the system needs data, the drone sends the data that is saved on it to the controller upon the request by the controller. When the controller receives the data, it displays it on the webpage, so the user can see the data and make necessary judgments of controlling the drone. The user will verify its execution by the updated live-feed.

Design Principles:

The design principle of this use case is High Cohesion Principle. There is more focus on displaying the necessary data and sending instructions to drone to control it, rather than having a high responsibility of computing data.

System Architecture and System Design

Architectural Styles

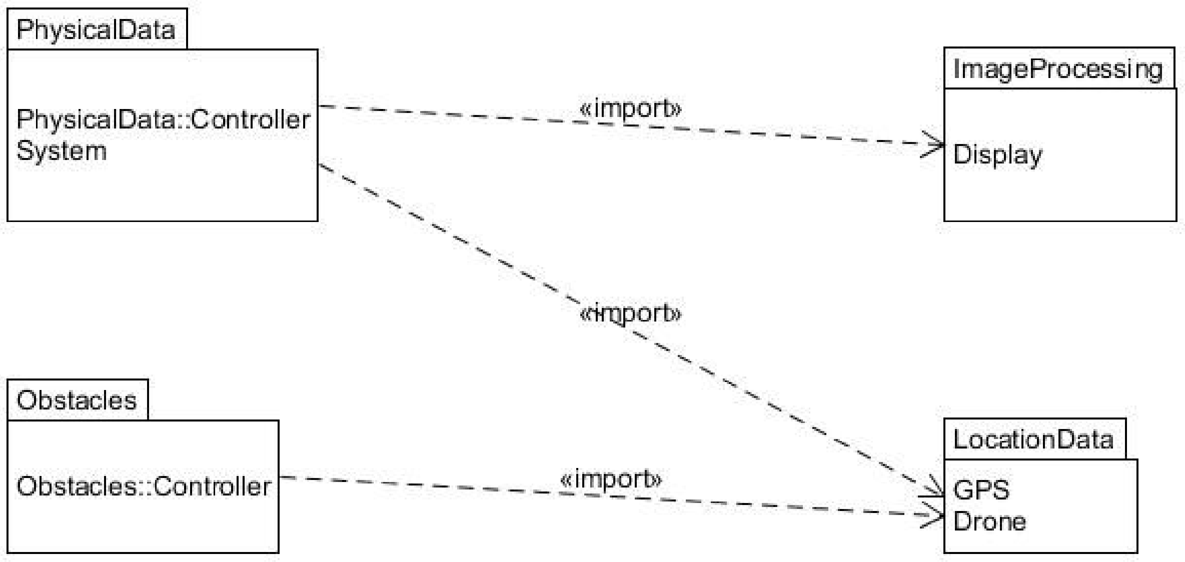
REST: Since all the information about the drone is displayed on a website using HTML, it complies with a RESTful API. The video feed, physical drone data, and controller inputs are gathered from servers controlled by other subsystems and represented as hypertext.

Client/Server: The client is the person controlling the drone and the server is the information picked up by the drone. When the user wants to throttle the drone, for instance, a request is sent to the drone's motors to move up or down, which then responds with movement seen by the live video feed.

Layered: Certain services of the drone depend on each other. For example, getting the current location of the drone is initiated by the controller, which depends on the drone requesting its location, which depends on the coordinates returned by the GPS.

Uniform interface: All resource should be reachable from any devices. It should not be constraint to only one device. The website should be simple but effective.

Identifying Subsystems



The PhysicalData package contains part of the Controller class to power the drone. It imports the LocationData package to get the drone's location and imports the ImageProcessing package to display live data on the controller. It also contains the System class to display the current status of the drone.
The ImageProcessing package contains the Display class for displaying the live video feed and physical data of the drone.
The Obstacles package has its own Controller class for controlling the drone to avoid obstacles. Like the PhysicalData, it imports the drone data from LocationData to steer the drone in the right direction.
The LocationData package has a Drone class to request its location and a GPS class to retrieve the drone's position.

Mapping Subsystems to Hardware

1. Physical Data

The majority of the hardware for physical data will incorporate the Raspberry Pi, which will send the required signals to the software component. The motors will also be involved when it comes to the detection of speed.

2. Image Processing

The hardware needed for this substem will include a camera inside a phone that will be mounted to the drone. The mobile device is going to be a Samsung Galaxy S4. The rear camera is a 13.0 MP autofocus camera with LED flash, with a Sony IMX091PQ sensor. We are also using an infrared lens to be able to detect people.

3. Obstacles

The hardware that is mapped from the Obstacles' subsystem is the the ultrasonic sensor that will be attached in multiple locations around the drone. The model of the ultrasonic sensor that is going to be used is HC-SR04.

4. Location Data

The mapped hardware for the Location Data subsystem would be the GPS, which is inside the smartphone that is mounted to the drone.

Persistent Data Storage

The drone will be equipped with Raspberry Pi. Even with the drone powered off, this system will be capable of saving any data from previous flights. However, this data can be transferred to another system since it is unnecessary for the drone to carry all the data from previous flights.

Another form of data storage is the image component of the drone. Even though it is a live feed, it will require some form of data storage through cache memory. This is due to the fact that the image will be required to be transferred from one device to another. Similar to the image processing, the ultrasonic sensor will also have a cache data component. The ultrasonic sensor will have to transmit the distance between the drone and any obstacles to the controller. This data does not need to be stored for a long time, but is still required if any action is needed to be taken by the drone.

Network Protocol

For managing the network that our system will make use of, the HTTP communication protocol will be utilized. This was chosen because the data that is being transmitted ends up as part of a browser-based display, for which HTTP can be used for simpler client-server interactions. The webpage that the drone operator sees requires data regarding the drone's location (GPS), the live camera feed, and other drone-related physical data (battery level, speed, etc.). These need to be delivered across the drone's connection to the operator's device, which naturally calls for a web-based communication protocol layered around TCP/IP.

Global Control Flow

Our project can be noted as both procedure driven and event driven. The reason behind this is because initially the same steps have to be taken to initially operate the drone however it is mainly an event driven system because the case for why this drone is being used is different. There are a lot of situational factors so the user must generate a different series of actions in different order depending on the specific case we are looking at. So it is mainly event driven because it is very unlikely that the same steps will be taken in the same order for more than one event.

Our system is an event response type with concern for real time. Since it is real time it is not periodic. It is not periodic because the time differs for the different situations. There are no time constraints for each case because we don't know how long each case would take.

Hardware Requirements

The access to control the drone can be done through any touch-enabled device with a internet browser such as a smartphone or tablet. The device requires a minimum of 1 GB since to process the live-feed video from drone smoothly. There will also be a a camera mounted on a phone that is placed on the drone in order to capture video. The interior of the drone will contain a raspberry pi. The device that will process the live-feed from the camera has to have a colored display of a resolution of at least 1920 x 1080 to allow the user to see where the drone is clearly. This can be done with any modern display devices like smartphones or tablets. Because of the quality of the image that is transmitted from the camera on the drone, the connection between the controller and the drone has to operate smoothly, with relatively low latency. The wireless connection bandwidth is a 2.4 GHz connection.

Algorithms and Data Structures

Algorithms

The main factor of this project is to have a safe and efficient flight for drone. The drone will not be capable of performing tasks such search and rescue if the drone is not durable. To accomplish this goal is to control the velocity of drone and locate any obstacles on it's way. Calculating the velocity and distance between the drone and obstacles involve complex algorithms.

The velocity of the drone can be calculated by using the formula that states that $v = a * t$. The variable v will stand for velocity, a will stand for acceleration, and time stands for time. Of course though we will have to account for other factors such such as thrust and pitch for when we are going over the drone's movements. The total amount of thrust is going to be equal to the following equation.

$Ft = Ft0 * (\frac{Vmax - V}{Vmax}) - Fd$.

F_D is the drag force.

F_{t0} is the force of thrust when the velocity is at 0 meters/second.

F_D

$= 0.5 * \rho * Cd * [(A(front) * cos(P(max) - P(motor))) + (A(top) * sin(P(max) - P(motor)))] * v^2$

For the above equation the constant ρ is going to be equal to the density of air, while the constant C_d is equal to the drag coefficient. The variable P_{motor} is the pitch of the motor, while P_{max} is the maximum pitch the drone is able to achieve without losing altitude.

$P_{max} = cos^{-1}(\frac{m}{T0})$

The variable m is equal to the mass of the drone and the variable T_0 is equal to the total thrust of the drone.

When the ultrasonics sensors recognize any obstacles, the drone needs to know where the obstacles are. The drone is equipped with four ultrasonics sensors. The four sensors should locate the exact location of obstacles and alert the user if necessary, so the user can maneuver the drone. This can be also used in the function such as "return to home" when the drone autonomously return to the base.

The distance between two points in 3-D Cartesian coordinates involves using equation, d(P_1,P_2) = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$, P_1 = (x_1, y_1, z_1), and P_2 = (x_2, y_2, z_2) where P_1 can be the coordinates of drone and P_2 can be the coordinates of obstacle.

However, since the drone is equipped with the ultrasonics sensors, it can use the time it took for an ultrasonic wave to travel to an obstacle to calculate the distance between them. The equation will be, $distance\ to\ object\ =\ \frac{time * speed}{2}$.

Time is divided by two since the time it took is an ultrasonic wave is to be emitted and reflected back to the drone combined. Only one way is needed.

Speed is the speed of ultrasonic wave, which will be 340 meters/second in the air.

The speed of ultrasonic wave is significantly greater than the speed of drone, that speed of drone can be ignored in the calculation.

.

Then d(P_1,P_2) will be the distance between the drone and obstacle. If the distance is less than a safe distance, the user will notice the drone through the alert on screen and will be able to maneuver. It is very important that our algorithm is consistently checking this distance because it can alert a safety issue if needed. The equation we'll be using was mentioned earlier and it is d(P_1,P_2) = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$, P_1 = (x_1, y_1, z_1). This is going be estimated in cartesian coordinates because we are dealing with real time.

Data Structures

The main data structure that we will use is going to be an array. We will use an array because of its flexibility and performance. We need a structure that can store many variables and is possible to easily index. This array will mainly be used for the variables that are used for status and operation, along with physical data.

History of Work, Current Status, and Future Work

Merging the Contributions from Individual Team Members

Shantanu came up with the project idea and was able to explain how we could contribute to the project during weekly meetings. We decided to split the work into four subgroups: image processing, location data, physical data, and obstacles. Since not everyone can make it to the weekly meetings, each subgroup has set up their own meeting times to discuss specific functionalities to be implemented in this project. This also ensures that each person can discuss how they will contribute toward building S.A.R.A.

Krishna Mahadas created and shared the Google Drive for our project so we could easily collaborate on creating the reports.

Abhishek manages the GitHub repository to maintain the project code and divide the work among the team. Each branch corresponds to the different subgroups. Each person works on their subgroup work and when it's ready to be implemented, it is merged into the master branch.

A website is going to be made and developed with relevant updates to the project. This will be managed by Abhishek. Other team members will help.

Project Coordination and Progress Report

Image Processing:
The image processing component of the project mainly implements the use case ViewCamera. So far we have already been able to display what the phone camera is seeing on other devices such as a pc. We tried using multiple third-party applications and features of the Android phone to see which works best. Some third-party applications we tried are Sidesync, Alfred and IPwebcam. All of these applications are able to display decent quality video feed for a reasonable range using wifi. Another approach was using the screen mirroring function of the android. This approach also uses wifi and provides a really good quality image. However, it does not have much of the range due to the fact the phone needs to be close to where ever the display is being transmitted to. So we decided to try using the Alfred application for now. The Sidesync application allowed us to have a better version of screen mirroring and actually control the phone from the pc. So we are able to run our interface on the phone and then display it on the laptop. This is being done by using the HTML code of the webpage version of the application since the camera feed is coming from the phone camera itself. A prototype of the controller can be found on our website.

Location Data:
The use case that is the main function of location data is GetLocation. We already have code for this use case in HTML that provides the location of the given device in latitude and longitude form. Since the Sidesync app transfer the phone screen to the laptop, and the phone is running our interface, the location of the phone is transferred to the laptop.

Physical Data:
The physical data part of the project deals with the GetStatus and GetData use cases. It will also include the MoveDrone use case. Due to the hardware component of the project, this part of the project can be in effect once the drone is in full operation. Specifically this part of the project will depend on the use of a Raspberry Pi. So currently, we are all

working towards the construction and integration of the hardware and software components of the drone.
Obstacles:
The Obstacles section of the project deals with the remainder of the use cases. The use cases include CheckObstacles and AvoidObstacles. Similar to the physical data component, this section will mainly be in effect once the drone is working. As of right now we are using four ultrasonic sensors on the drone to check for nearby obstacles. They utilize the Raspberry pi and Python to detect the obstacles around the drone and print out the distance between the drone and the object.

History of Work

- Milestones:
    - *Drone Camera Transmission:* Be able to provide a reliable stream from the onboard phone camera to a mobile device set aside to mock the operator's control device.
        - Date of Completion: March 8th, 2019
    - *Hardware-Associated Tasks:* After all necessary hardware components arrive between March 1st-3rd, the construction of the drone frame to fit the needs of the project. This includes mounting the onboard camera and microcontroller to the drone frame.
        - Date of Completion: March 20th, 2019
    - *Webpage Integration:* Collecting all relevant data and finalizing transmission/display of said data to the operator's control device.
        - Date of Completion: March 22nd, 2019
    - *Sensors:* Managed to get the sensors incorporated with the raspberry pi and be able to record distance from an object.
        - Date of Completion: March 22nd, 2019

The milestones that have been completed so far and the planned milestone achievements so far are slightly different from each other. We initially experienced delays in receiving the hardware components on time so we could not meet up some of the expected milestones on the hardware side. We did manage to complete several of the other planned achievements on time. These achievements included getting the video feed from the mobile's camera and getting the webpage integration done on time.

The future plans for this project will be focused more on integrating everything around the drone and making sure that the various subsystems can work in cohesion. All future work for this project is going to be focused on the drone itself and not on the systems that utilize the drone or that can be placed on the drone.
Key Accomplishments:

- Webpage Integration
- Camera Transmission

Breakdown of Responsibilities

- Project divisions:(all tasks that are in progress/to be completed)
    - Visual Data Processing:
        - Shantanu: Management of the main wireless network/communication of data
        - Abhishek: Webpage development/Data handling on operator-side
        - Krishna Mahadas: Onboard camera handling, transmission (in progress)
    - Obstacle Management
        - Vishal: Managing sensor data, implementing avoidance/assoc. movement
    - Location Data
        - Avnish: Gathering onboard GPS data, transmission
    - Physical Drone Data
        - Krishna Tottempudi: Determining overall operational status from collected data
        - Sahana: Determining power levels/operational lifespan of drone real-time
        - Won Seok: Determining the strength of signal/connection to the operator

All other contributions to the project can be found in the individual contributions breakdown matrix on page 2.

References

1. “Drone Sense” https://www.dronesense.com/?gclid=EAIaIQobChMIqo-45_Gx4AIVwoCfCh2CbA0QEAAYASAAEgKMu_D_BwE
2. Byran, Cantfil. “ United States Coast Guard Search and Rescue Summary Statistics 1964 thru 2015.” https://www.dco.uscg.mil/Portals/9/CG-5R/SARfactsInfo/SAR%20Sum%20Stats%2064-16.pdf
3. Rhode, Steve.“DRONE SEARCH-AND-RESCUE STUDY REVEALS POTENTIAL, LIMITS” https://www.aopa.org/news-and-media/all-news/2018/october/01/drone-study-reveals-potential-and-limits
4. “Image 1” https://s.yimg.com/ny/api/res/1.2/2P8Y6UqlB8dKOiVIg9Rscg--~A/YXBwaWQ9aGlnaGxhbmRlcjtzbT0xO3c9ODAw/http://media.zenfs.com/en-US/homerun/digital_trends_973/8122e594705a009db372bf32720d9fe9
5. “Using a Raspberry Pi distance sensor (ultrasonic sensor HC-SR04).” https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/
6. “Raspberry Pi Distance Sensor: How to setup the HC-SR04” https://pimylifeup.com/raspberry-pi-distance-sensor/
7. “HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi” https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi
8. “Installing GPIO” https://gpiozero.readthedocs.io/en/stable/installing.html

9. “The Equations for Speed”
   https://quadstardrones.com/the-equations-for-speed/
10. “Implementing an in-browser camera”
   https://davidwalsh.name/browser-camera
11. “Tracking current location”
   https://www.w3schools.com/html/html5_geolocation.asp