Software Engineering

Group 8

Search And Rescue Assistant (S.A.R.A.)

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

Mar. 10, 2019

Team Members:

Sahana Asokan

Won Seok Chang

Avnish Patel

Abhishek Chaudhuri

Shantanu Ghosh

Srikrishnaraja Mahadas

Sri Sai Krishna Tottempudi

Vishal Venkateswaran

Individual Contributions Breakdown

Project Deliverable	Sahana	Won Seok	Avnish	Abhishek	Shantanu	Krishna M.	Krishna T.	Vishal
Interaction Diagrams (10 points)		16.7%	33.2%	16.7%			16.7%	16.7%
Design Principles (10 points)	33.2%	16.7%	16.7%			16.7%	16.7%	
Class Diagram & Description (5 points)		33.3%			33.4%		33.3%	
Signatures (5 points)					33.3%	33.4%	33.3%	
Styles (5 points)		16.7%		83.3%				
Package Diagram (2 points)				100%				
Map Hardware (2 points)							33.3%	66.7%
Persistent Data Storage (3 points)		50%				50%		
Other (3 points)	33.3%		16.7%		16.7%			33.3%
Merging Contributions (11 points)				33.4%	33.3%			33.3%
Project Coordination (5 points)						100%		
Plan of Work (2 points)				50%	50%			
References (-5 points)	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Total	4.29	7.365	5.521	12.509	8.474	12.35	7.371	7.666

Responsibility Allocation

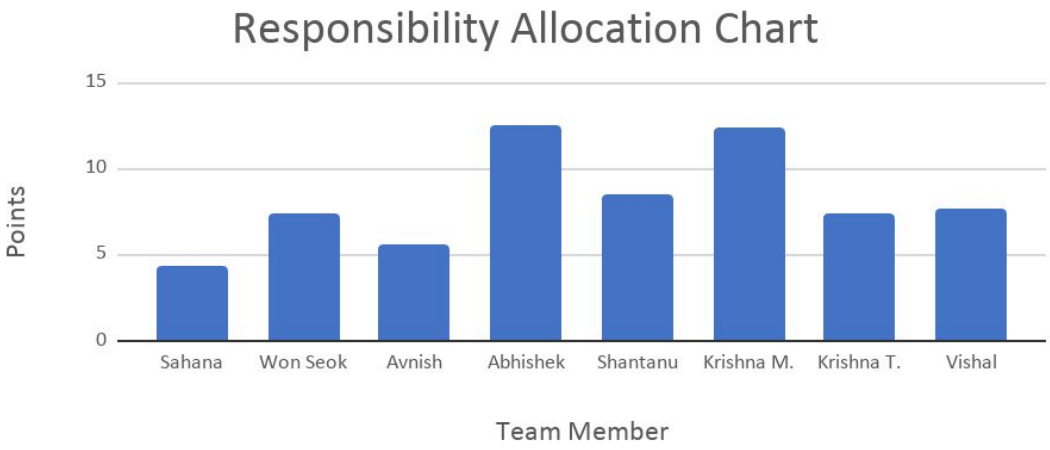
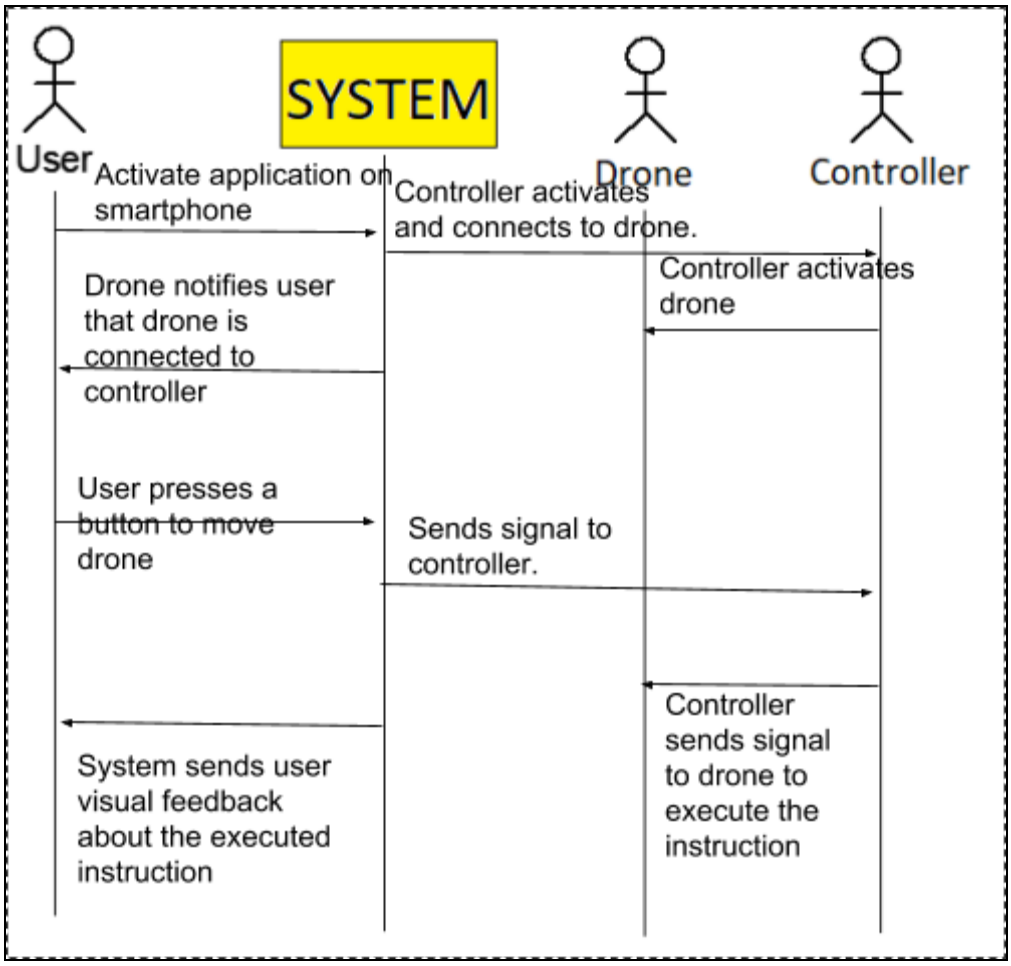


Table of Contents

Interaction Diagrams and Design Principles

Use Case 1 - Move Drone

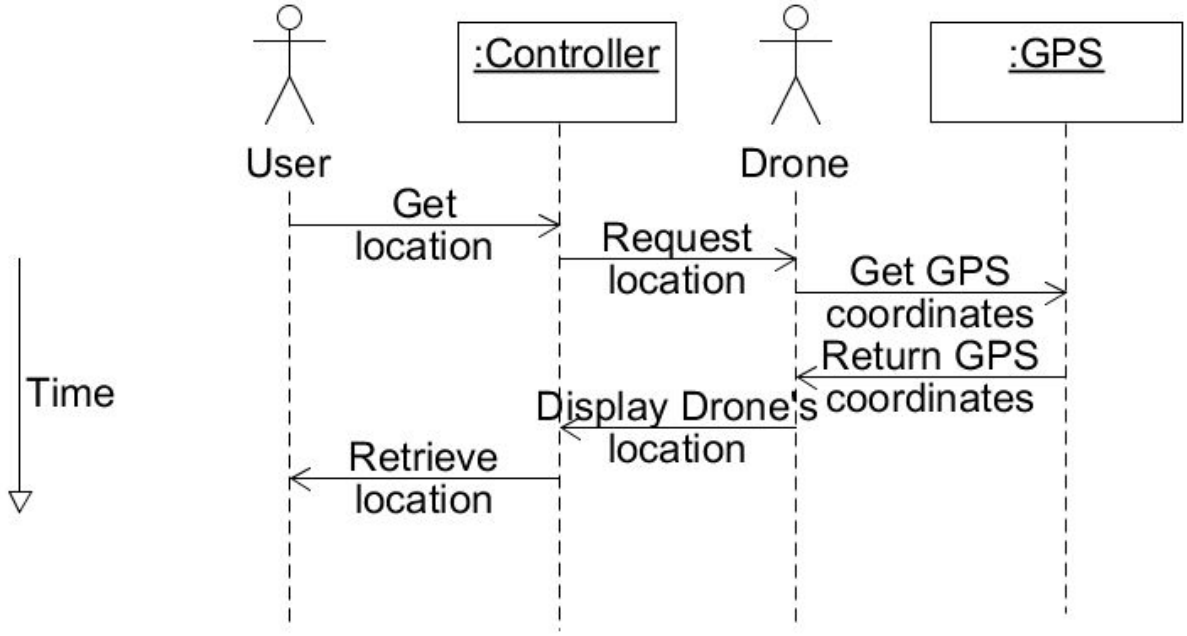


The diagram for the first use case is displayed above. In this case, the User will first activate the system in order to gain access to the controller. From there the User is able to use the controller to move the drone. The User will be able to see visual feedback from the drone.

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.

UC-3: GetLocation

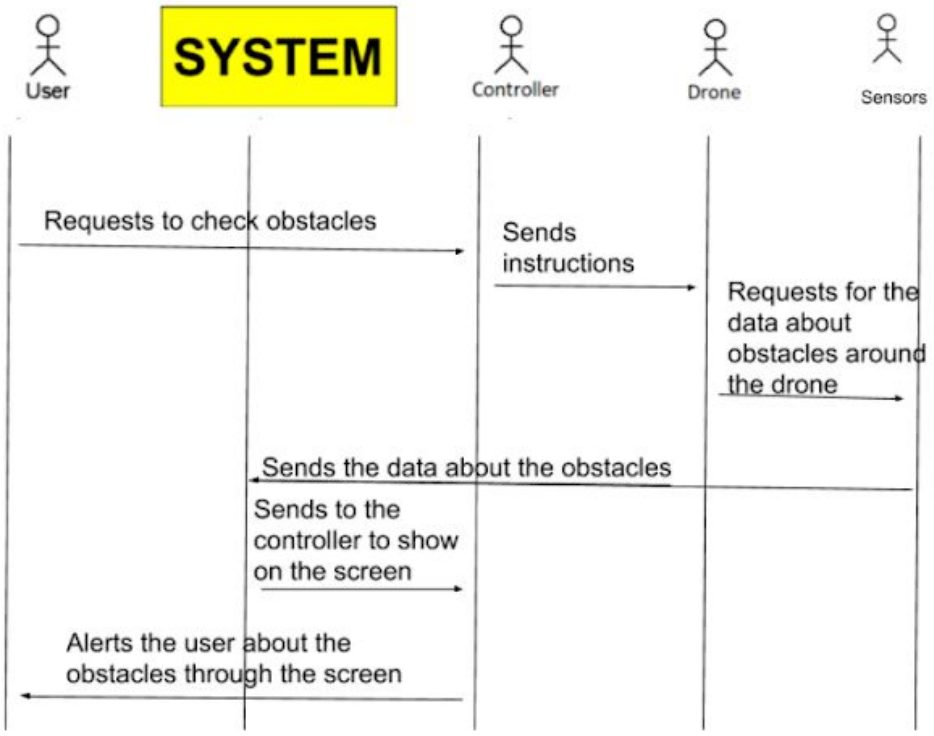


The diagram above demonstrates the interactions between classes in UC-3: get the location. Once the user has control of the drone and being able to maneuver around obstacles. First, the user sends a request to get the location of the drone to the controller which then gets requests it to the drone. The 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.

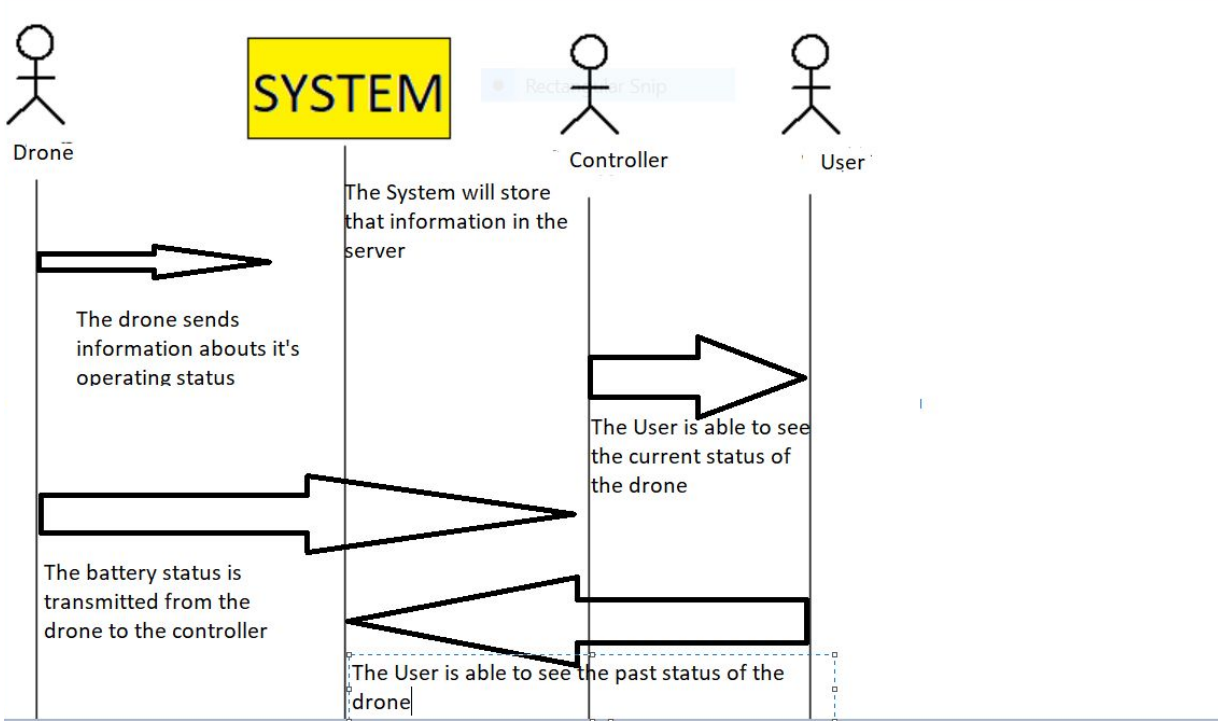
Use Case 4 : Check Obstacles



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 -Get Status

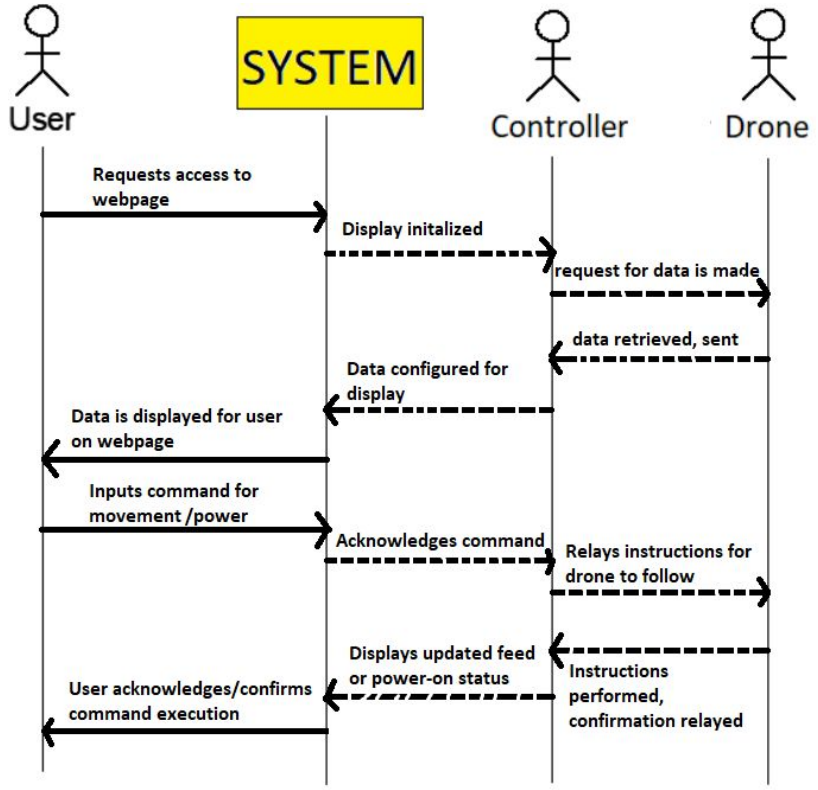


The interaction diagram for use case 6 is displayed above. The drone basically sends a signal to the system which the user can see the result of through the controller. The part of this use case is to let the user know of the operation status of the drone, in particular, the battery level. The user is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

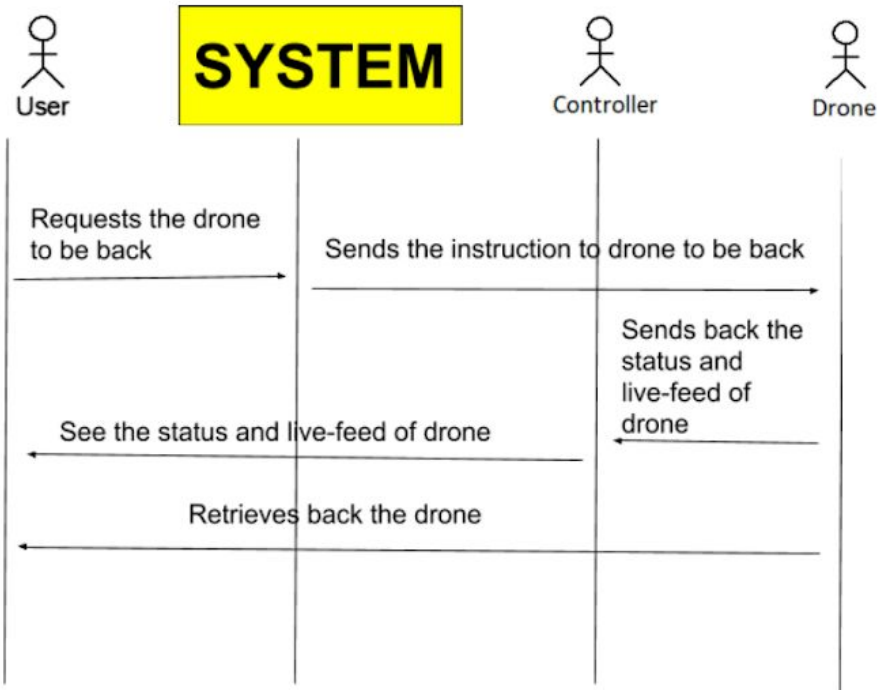


The diagram above demonstrates how the user, controller, and drone interact with each other to show necessary data of drone to the user, so the user can control the drone. When the system needs data, the drone sends the data that is saved on it to the controller upon the request by the controller. When 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.

Use Case 8 :Return to Home

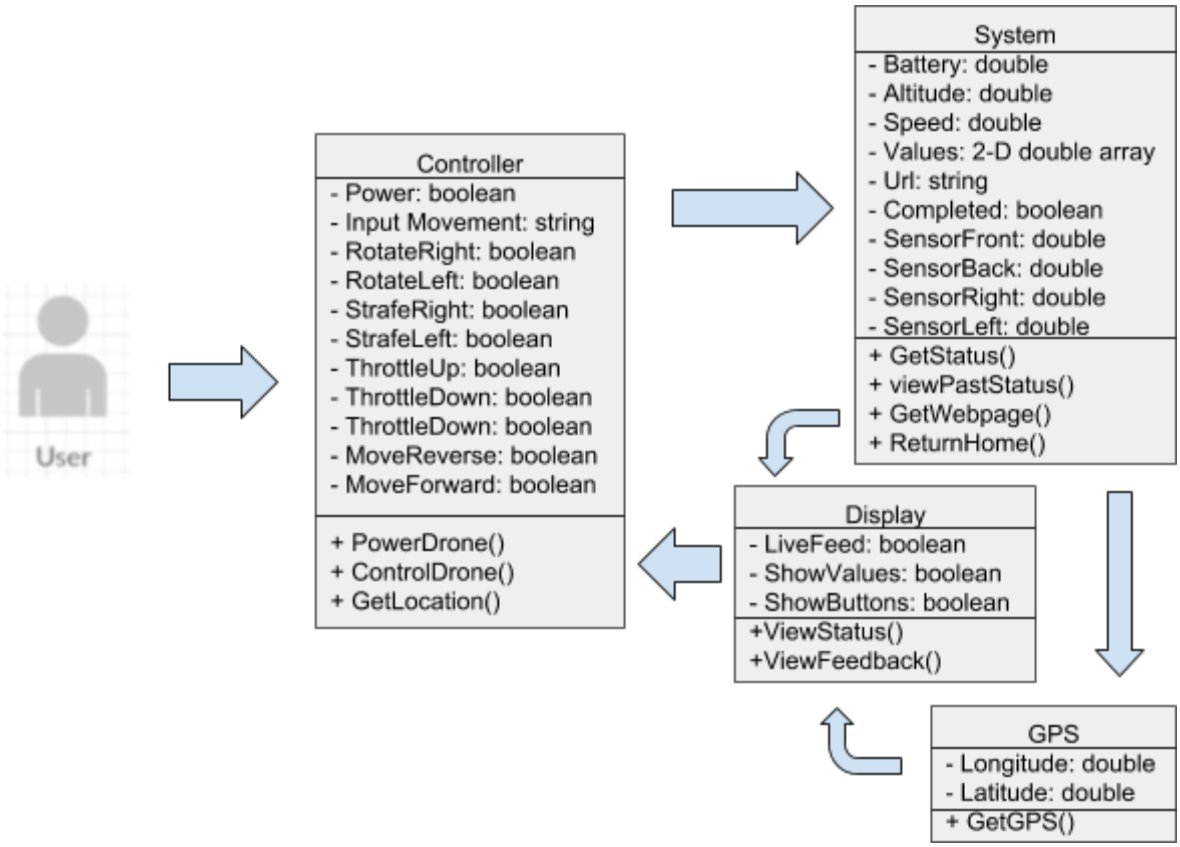


Design Principles:

The design principle for this use case is both Expert Doer Principle and Low Coupling Principle. It is important that this object reports when the drone needs to be returned to the original location and to be able to recognize when its supposed to return to the original location. It is also implementing Low Coupling Principle because it should only be focusing on data and communication relating to returning home.

Class Diagram and Interface Specification

Class Diagram



Data Types and Operation Signatures

1. **Controller:**The main role of this class is to use the help of the other classes to maneuver and utilize the drone.

   Operations

   PowerDrone()-Boolean: This function is responsible for turn the drone on and off. It will have a value of true if the drone is on and false otherwise.
   ControlDrone()-Void: The purpose of this function is to maneuver the drone in a certain direction based on the parameters and it will not return anything. The parameters are type of movement (String). The acceptable string inputs are Rotation, Forward, Reverse, throttle, Strafe.
   GetLocation()- Double Array: This uses the GPS to get the location of the drone in the form of latitude and longitude.

   Attributes

   1. Power: boolean -It has a value of true or false depending on if the drone is on or not.
   2. InputMovement: string-This is the type of movement the user wants to do.
   3. RotateRight: boolean- Has a value of true if the user wants to get the drone to move to the right
   4. RotateLeft: boolean - Has a value of true if the user wants to get the drone to rotate to the left
   5. StrafeRight: boolean - Has a value of true if the user wants to get the drone to move to the right
   6. StrafeLeft: boolean - Has a value of true if the user wants to get the drone to move to the left
   7. ThrottleUp:boolean-Has a value of true if the user wants to get the drone to move to the up.

8. ThrottleDown:boolean-Has a value of true if the user wants to get the drone to move to the down.
9. MoveForward: boolean-Has a value of true if the user wants to get the drone to move to the forward.
10. MoveReverse: boolean - Has a value of true or false that will depend on whether the drone is moving backwards or not.

2. **System:** The system class mainly deals with the inner mechanisms of the drone and its operational status.

Operations

1. GetStatus()-Double Array: The main role of this function is to gather the information of the drone. Some of the possible data it will gather will be the battery life, altitude, and speed along with any other needed information.
2. ViewPastStatus()-2-D Double Array: The values returned from GetStatus will be stored in a 2-d double array. If this function is called it will return the mentioned array.
3. getWebPage(): String: This function will output the url of the user interface in the form of a string.
4. ReturnHome(): Boolean: Once the function is executed and has completed/ failed its task it will either output true or false respectively.

Attributes

1. Battery: double- The amount of battery life left on the drone.
2. Altitude: double-The height of the drone based on the sensors.

3. Speed: double-The speed of the drone in mph.
4. Values: 2-d double array-Contains battery life, altitude, speed and any other required values in a given instant. Each row represents one instance.
5. Url: string-The url of the display website.
6. Completed: boolean - Has a value of of 1 or true if the drone has managed to return home.
7. SensorFront: double - The time it takes for the front sensor signal to be transmitted and returned to the drone.
8. SensorBack: double - The time it takes for the back sensor signal to be transmitted and return to the drone.
9. SensorRight: double - The time it takes for the right sensor signal to be transmitted and return to the drone.
10. SensorLeft: double - The time it takes for the left sensor signal to be transmitted and returned to the drone.

3. **Display:** This class is mainly responsible for showing and updating the graphical user interface.

Operations

1. ViewStatus()- Boolean: This function takes the values of GetStatus and displays them in the appropriate place in the GUI. Upon completion it will either return true or false.
2. ViewFeedback()-Void: This is responsible for displaying the videofeed of the drone. Since this is happening constantly, it will have a return type of void.

Attributes

1. LiveFeed: boolean-Has a value of true if the camera feed is being displayed on the controller.
2. ShowButtons: boolean-Has a value of true if the buttons are being displayed on the controller.
3. ShowValues: boolean - Has a value of true if there are values such as speed, battery life, altitude are being displayed on the controller.

4. **GPS:** The class GPS is used to implement the location component of the drone. The functions for this class are displayed below.

Operations

1. GetGPS(): Boolean: This function is there to initiate the connections required for GetLocation to work. Upon completion it will either return true or false.

Attributes

1. Latitude: double - Returns the current latitude of the drone.
2. Longitude: double - Returns the current longitude of the drone.

Traceability Matrix

Domain Concepts/ Software Class	Controller	System	Display	GPS
Controller	X			
Interface		X	X	
Connector	X	X		X
Page Maker			X	
Dynamic Data		X		X
Notifier		X	X	
Calibrator	X	X		

The software classes were developed from the domain concepts based on the required functionality of each domain concept. The core functions of the drone involved movement, operational status, user interface, and location. Each of these core functions represents a software class. So the traceability matrix above represents how each of the domain concepts correspond to the core functions of the drone i.e. the software classes.

System Architecture and System Design

Architectural Styles

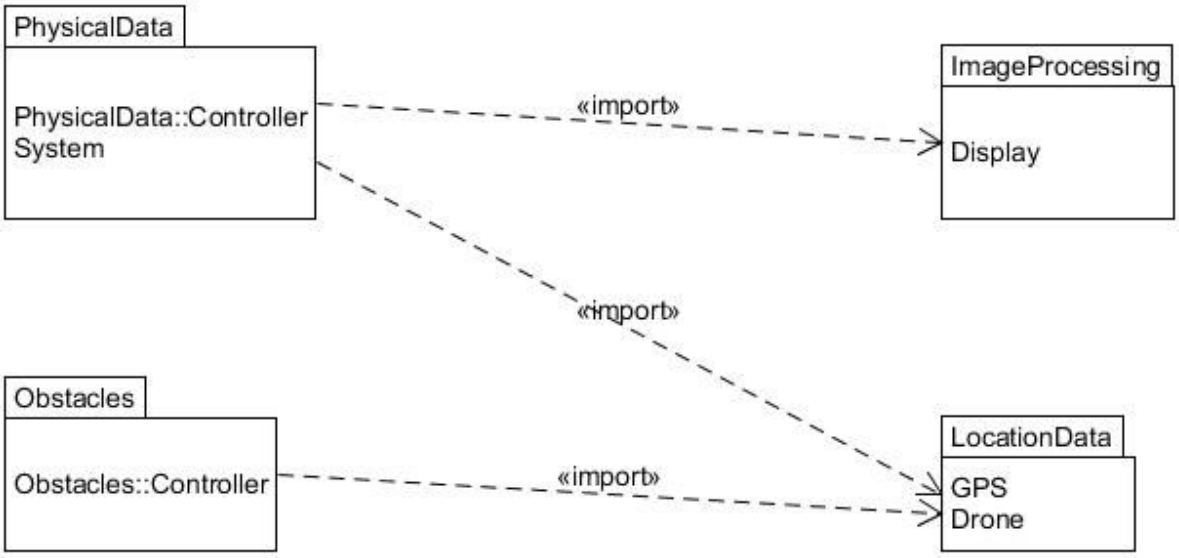
REST: Since all the information about the drone is displayed on a website using HTML, it complies with a RESTful API. The video feed, physical drone data, and controller inputs are gathered from servers controlled by other subsystems and represented as hypertext.

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.

Project Management and Plan of 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 Alfred and IPwebcam. Both 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. Currently, we are trying to manipulate the given video feed from the application, so that it is possible to transfer the video feed to our user interface. This is being done by using the HTML code of the webpage version of the application since the camera feed is currently being displayed there. 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. The next step is to transmit this information from the android device to the user interface.

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. 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, AvoidObstacles, and ReturnToHome. Similar to the physical data component, this section will mainly be in effect once the drone is working. So as mentioned before, we are all working to complete the hardware and software parts of the drone.

Plan 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
    - *Onboard Data Management/Transmission:* Determining the operational status of the drone from real-time data/status of equipment, signal, etc. This needs to be successfully relayed back to the operator's control device.
        - Date of Completion: April 1st, 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
    - *"Crash-testing":* Success of Flight testing to determine drone's survivability
        - Date of completion: April 8th, 2019
    - *Obstacle avoidance:* In addition to recognizing objects in its' way, the drone will react to maneuver out of harm's way/avoid pathing into roadblocks
        - Date of completion: April 15th, 2019

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

https://www.dronesense.com/?gclid=EAIaIQobChMIqo-45_Gx4AIVwoCfCh2CbA0QEAAYASAAEgKMu_D_BwE

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

Drone Statistics
https://www.aopa.org/news-and-media/all-news/2018/october/01/drone-study-reveals-potential-and-limits

Coast Guard Table:
https://www.dco.uscg.mil/Portals/9/CG-5R/SARfactsInfo/SAR%20Sum%20Stats%2064-16.pdf

Obstacle Sensor(HC-SR04) Information:
https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/
https://pimylifeup.com/raspberry-pi-distance-sensor/
https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi