# Safe Swarm

## Preemptive and Rapid Response Countermeasures for Public Safety in the 21st Century

Heather Kemp
University of Iowa
Iowa City, Iowa
heather-kemp@uiowa.edu

Theo Linnemann
University of Iowa
Iowa City, Iowa
theo-linnemann@uiowa.edu

Yusuf Sermet
University of Iowa
Iowa City, Iowa
muhammedyusuf-sermet@uiowa.edu

## ABSTRACT

This paper describes the Safe Swarm drone application for Mobile Computing and reviews the method of development and testing for the system.

## CCS CONCEPTS

• **Computer systems organization** → **Robotics**; • **Computing methodologies** → *Robotic planning*; • **Hardware** → *Wireless devices*;

## KEYWORDS

Drones, wireless networking

**ACM Reference Format:**
Heather Kemp, Theo Linnemann, and Yusuf Sermet. 2017. Safe Swarm: Preemptive and Rapid Response Countermeasures for Public Safety in the 21st Century. In *Proceedings of (Mobile Computing).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.475/123_4

## 1 INTRODUCTION

In June 2014, SSH and GfK carried about a national survey of 2,000 people in the USA about harassment as people walked the streets. They found that 65% of women experienced street harassment, 23% had been sexually assaulted, and 20% had been followed, while 25% of men reported to being harassed as well. Enter the Safe Swarm, a fleet of autonomous drones that will deploy to a mobile application users location as their own on-demand, 24/7/365 personal security solution. The Safe Swarm drones, at their current capacity, are able to be stealthily activated with just a simple click of the button. Once the Summon button has been pressed, the app will automatically detect the user's location and provide it to a Safe Swarm drone, which will be deployed immediately to the location. While the drone deploys, it will be providing live video feed, as well as a stored video of its mission to the user. The Safe Swarm drones will facilitate a feeling of safety for the users as they wait for a ride or their loved ones to arrive while decreasing the likelihood of an incident occurring, as perpetrators will be caught through the live video footage.


*Mobile Computing, ,*

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

## 2 PROBLEM FORMULATION

To accomplish the task of automatic drone deployment to a user's app, the following resources were utilized: a database to store feedback information, a client form and deployment form of the Safe Swarm app,the user's GPS location, an idle drone, and a screen recording app. Using these resources and the methodologies described in section 4, the following resources to address the problem were created: a waypoint mission, which is used by the drone to navigate itself to the user, video feed of the user from the drone both live streamed and on file, and a dismissible drone companion for the user.

## 3 CHALLENGES

In accomplishing our goal, we were not without our fair share of challenges. The challenges described below were what we deemed as the most relevant challenges, disregarding the amount of work put towards solving them.

### 3.1 Missing Direct Connection

The DJI drone that we worked with relied on a connection between the drone and a controller which was stationed at the hive. This being said, there was no way to have a connection between the drone and the user, who would be calling for the drone from an undetermined location. With this limitation we still had to perform the following tasks: presenting the drone video and user location feedback in real time; detecting when the user started and ended the mission; and providing consistent location states between the user's app and the drone.

### 3.2 Automated Deployment

In part because of safety concerns, the DJI drone's default operation requires user input for takeoff and piloting. Ideally, we would not have someone stationed at the hive at all times, and so for a successful implementation of this project, we needed the drone's on standby to be able to automatically deploy to a non-static location based on what another device, unconnected to the drone itself, says to do.

### 3.3 Maintaining Position by User

To provide the safety and video footage our design document specified, the drone will be required to not only remain positioned by the user, but also keep its camera focused on the user as well until the drone is told to return home.

## 4 APPROACH

Before addressing the ways in which these challenges were addressed, the software design itself must be discussed first. (Fig. 1) The execution of the software starts with the user on the client side of the application, who presses the Summon button on the app. Doing so will send the user's location, including latitude, longitude, and altitude, along with other information, to the Firebase database, which will then, having detected the change in its dataset, will forward this user location data along to the deployment of the app, which is residing with the drone. This back end constructs a waypoint mission, which contains the midpoint between the drone and the user's location and then the user's location itself. Once the waypoint mission is uploaded to the drone, it deploys to the user's location, where it will remain until the user presses the End Mission button. Once again, this button click will send an updated mission status to the Firebase database, which will then notify the back end app of the mission termination. This update will call a method which initiates the return to home procedure for the drone to land back at the hive.

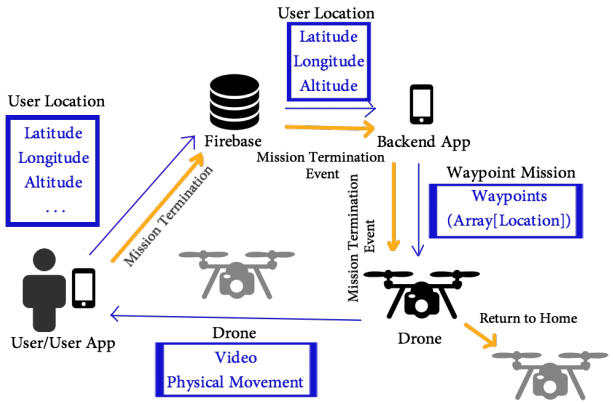


**Figure 1: High Level Software Diagram**

### 4.1 Missing Direct Connection

Two separate tools were used to compensate for the lack of a direct connection. The first tool is the Firebase database. This database would store the drone's status, user location, and other variables for consistent retrieval and storage across devices. Furthermore, when the database was updated with things such as a new mission or a mission's termination, it then was able to call a method on the hive back end application, which then either deployed a drone or called a drone back respectively. For the recording requirement, AZ Screen Recorder was used on the back end application. This was a design choice made due to the fact that the back end application temporarily will go to sleep to allow for the drone to catch up with the code, and we thus didn't want the video stream to splutter with the sleeping of the app itself. Thankfully, AZ Screen Recorder not only stored the drone operator footage as a file, but it also functioned as a streaming service for this video to the user's phone directly. Together, these two tools accomplished everything that required a direct connection between the user's phone and the drone, and then more.

### 4.2 Automated Deployment

To accomplish the automated deployment, two separate tools were used once again. First, we have the DJI SDK, which provided much of the backbone for the development of this feature through their provided methods. Furthermore, it provided things like obstacle avoidance and flight controls already built in. To pilot the drone, we took the location provided by the second tool, the Firebase database, for the user's current location, and built a waypoint mission which consisted of the midway point between the deployment location and the user's location, as well as the destination itself. This waypoint mission, which was then uploaded to the drone, allowed it to navigate easily to the user to prepare for monitoring.

### 4.3 Maintaining Position by User

The task of maintaining focus on the user was by far the most difficult to accomplish, which is why, given the time allotted for this project, we were not able to fully complete it, despite investigating several ways to implement this feature. Before going over those ways, it should be first stated that the drone's WiFi capabilities are disabled in R/C mode, which is used for the mission execution. It became apparent that WiFi is required for any significant range for the drone, beyond the additional constraints. The first mission we attempted to program was a "Follow Me" mission, which allows the drone to follow a GPS signal, but as we learned this signal is limited to the one from the R/C controller itself, and so this mission was not practical for our purposes. The second mission attempted was the waypoint mission, which we were already using to navigate to the user. However, waypoints can't be added in real time during flight to account for the user's movement, meaning we would have to cancel the mission, send the drone back to the hive, and then resend the drone back to the user's new location, which isn't a feasible execution plan for our purposes. The third and final mission considered was the "Active Track" mission. This mission wouldn't require GPS signals, but it would require selecting the user on the deployment device's screen or detecting user actions on the camera. This could have been overcome with more time, but would likely require intercepting and interpreting the raw drone stream data. Given more time to accomplish this task, we would have pursued the "Active Track" mission more thoroughly to allow us to follow the user even if they were moving at higher speeds than basic walking.

## 5 EXPERIMENT SETUP

System evaluation was based on the completion of the drones' primary objective in 3 different scenarios.

### 5.1 Primary Objective Definition

The primary objective was defined as the drone, in an automated manner, successfully deployed upon request, positioned itself within the goal cylinder, and tracked the user until either the user sent the dismiss command or the drone's energy supply was depleted. A successful rapid deployment was defined by the drone taking no longer than the expected travel time to the location plus the average time to target lock with a tolerance of 10%. The goal cylinder was defined as a right circular cylinder described by the location of the user where the user's longitudinal axis defined the cylinder's

central vertical axis. The goal cylinder consists of 3 parameters, the radius from the user, the maximum altitude bound, and minimum altitude bound. The three scenarios we consider are described in sections 5.2, 5.3, and 5.4.

## 5.2 Static Target Scenario

In the static target scenario, we considered the unmoving user standing at a distance of 100 meters from the point of deployment. Both the user and deployment point were at approximately the same elevation. There were no obstacles within the shortest path between the drone and the user. In this scenario, the goal cylinder was defined by a radius of 1.85 meters, a maximum bound of 8 meters and minimum bound of 6 meters. We evaluated the drone's performance based on its successful completion of the primary objective as defined in section 5.1 and the drone's ability to maintain its position for up to 60 seconds.

## 5.3 Low Velocity Dynamic Target Scenario

In the low velocity dynamic target scenario, we considered the moving user at an average adult walking pace of approximately 5.0 kilometers per hour. Initial conditions were similar to the static scenario in 5.2 in that the altitude was approximately the same as the user's upon deployment and there were no obstacles between the user and the drone. We defined the goal cylinder to have a radius of 2 meters, a maximum bound of 9 meters and minimum of 6 meters. A successful execution of this scenario required completion of the primary objective, in addition to the following. The drone must track the target continuously for a minimum of 60 seconds while the target started and stopped moving and took a minimum of eight 90 degree turns, with at least two left turns and right turns.

## 5.4 High Velocity Dynamic Target Scenario

In the high velocity dynamic target scenario, we considered the moving user at an average adult running pace of approximately 16 kilometers per hour. Initial conditions were again similar to scenarios 5.2 and 5.3, however the user was in movement at the time of deployment. In this scenario, we defined the goal cylinder to have a radius of 2.5 meters, a maximum bound of 10 meters and a minimum bound of 6 meters. In addition to the primary objective, the drone must have tracked the target continuously from the time of initial target lock for 60 seconds. The user's path contained no less than four 90 degree turns consisting of at least one left and right turn.

## 5.5 Target Metrics

During the three testing scenarios, we used a custom scoring rubric out of 80 points defined as follows.

(1) One point for each second that the drone maintained target lock within the goal cylinder, up to sixty points.
(2) Ten points for a successful rapid deployment.
(3) Ten points for a successful deployment termination, which is defined as returning to the point of deployment when its primary objective is complete.

## 5.6 Experiment Results

The final tests for this project were done at Iowa City Aerohawks corresponding to the criteria described in section 5. The results are as followed.

**Table 1: Static Target Testing Results**

| | |
|---|---|
| Deployment | 10 |
| Retrieval | 10 |
| Goal Cylinder (Out of 60 seconds) | 60 |
| **Total (Out of 80)** | **80** |

**Table 2: Low Velocity Dynamic Target Testing Results**

| | |
|---|---|
| Deployment | 10 |
| Retrieval | 10 |
| Goal Cylinder (Out of 60 seconds) | 20 |
| **Total (Out of 80)** | 40 |

**Table 3: High Velocity Dynamic Target Testing Results**

| | |
|---|---|
| Deployment | 10 |
| Retrieval | 10 |
| Goal Cylinder (Out of 60 seconds) | 10 |
| **Total (Out of 80)** | 30 |

While these results in the low velocity dynamic target (Table 2) and high velocity dynamic target (Table 3) scenarios were less than we anticipated, we can attribute this to, as previously stated, a lack of time to properly implement the active track feature. In the static target scenario (Table 1), the drone gave a perfect performance, managing to keep focused on the user due to the lack of movement. As a result, in cases where the user would be waiting where they are, such as waiting for a ride to arrive, the SafeSwarm drones would successfully meet their requirements.

## 5.7 Demonstration

For the sake of documenting the results of these experiments, video footage was taken throughout the process, focusing on the static target scenario in section 5.2. This footage can be viewed on the playlist provided:
youtube.com/playlist?list=PLRLtuTuDXS0wuWsrjO7J-dApJiVMe8Ff4

# 6 LESSONS LEARNED

While we were not successful in all of our goals, we believe that we learned many lessons that will help us grow as computer scientists.

## 6.1 Asynchronous Programming

Due to the nature of the project, we were attempting to write an imperative program, but the methods in the proprietary SDK required exclusively asynchronous function calls, while also being a bit buggy themselves. After trying many times to brute force

our way through this way of programming, we realized that we had to generalize the problem, which allowed us to implement a "call-and-wait" style of execution for nondeterministic functions. In this way, while it was not strictly imperative, we could guarantee that the required parts of the code were executed before a future method was called.

### 6.2 Logistics and Tooling

In order to properly work with this project, an adequate amount of work was done in familiarizing ourselves with different ways for testing. For example, non-trivial amounts of time were spent on figuring out how to get wireless debugging working, as we were unable to keep the phone plugged in to the laptop for console logging. Furthermore, when it came to the design of the actual tests themselves, our flight time was primarily constrained by the early winter sunsets, and by having only one person with an LTE enabled Android device, we would have to perform each part of the summoning step separately rather than simultaneously. In the future, as we now know these types of problems that will occur, we will be able to take precautionary actions ahead of time to prevent losing valuable project time.

### 6.3 Documentation Clarity

When it came to the documentation for automatic drone deployment and user tracking, information was sparse at best. We came to realize that we wouldn't find resources for exactly what we wanted to know, and so we ended up assimilating parts from almost every DJI tutorial manually to automate flight control. By doing this task, we learned not only how to compile resources for a difficult task, but also that sometimes you need to work with many different sources before you find the information that you will need, especially for an uncommon topic in academia like drone deployment.

### 6.4 Version Compatibility Issues

The basic DJI SDK integration for the drone was challenging by itself, as it was not kept up to date perfectly with the latest version of the Android SDK. To accommodate for this fact, we had to learn some advanced techniques for managing and forcing the use of correct dependencies to even proceed with development.

## 7 CONCLUSION

At the end of this project, it became apparent that our goal, while simple in theory, was much loftier than anticipated. The scope of the challenges were more significant and there were additional constraints that we had not previously recognized. By working through the challenges documented, we learned an immense amount and successfully implemented a majority of our functional components.