



# TERM PROJECT

VR Drone Controller

JUNE 6, 2018



Özkan Yumşak	2050565
Orhun Altunlu	2204808
Oğuz Bilgen	2136687

## Table of Contents

1. Introduction.....	3
2. Background .....	3
3. Related Work.....	3
4. Tools and Technologies .....	3
4.1 Tizen Studio.....	3
4.2 Samsung Accessory Protocol and Samsung Accessory SDK .....	4
4.3 Android Studio .....	4
4.4 Parrot Bebop 2 Drone .....	4
4.5 Samsung Note 4 .....	4
4.6 Parrot SDK 3 for Bebop 2 .....	4
4.7 LibGDX & LibGDX Freetype Extension .....	4
4.8 Google GVR SDK .....	4
4.9 OpenGL ES 2.0.....	4
5. Solution Methodology .....	5
5.1 Smartwatch Application Development.....	5
5.2 Android OpenGL ES & Cardboard Implementation .....	5
6. Experiments and Solutions.....	6
6.1. For Tizen Smartwatch.....	6
6.2. Low Pass Filter for Sensor Data and Drone Control.....	8
6. 3. Experiments & Results for VR Display .....	12
6. 4. Experiments & Results for Drone Flight Path .....	16
7. Discussion and Conclusion.....	16
8. References.....	16
9. Repositories: .....	17

## 1. Introduction

In the scope of this Term Project definition, three different modules are developed under the name of VR Drone Controller Application by three participants. A VR application is developed on Android which is receiving live camera feed from a Parrot Bebop 2 Drone and drawing OpenGL elements on the video which is providing an user friendly interface on Google Cardboard's VR experience. Another module is developed on Android to control the drone movements while the Cardboard is worn and VR is experienced by identifying head movements. The last module integrated a smartwatch application on Samsung Gear S3 to the Android application which is identifying the hand gestures to extent to user experience in this environment to a different level in terms of controlling the drone.

## 2. Background

The participants in this project are Computer Engineers who are pursuing their Master's Degree education in Middle East Technical University.

We are experienced and colleagues in defense industry on user interface development area for Avionic Systems. We are familiar with Head Up Display elements and Sensor data management as our job requires.

## 3. Related Work

We decided to conduct this project because we had the opportunity to use our knowledge and apply on a different and enjoyable scope.

We obtained the Parrot Bebop 2 drone which is providing a simple and useful SDK for development. We already had the Samsung Gear S3 smartwatch and Android phones. We wanted to combine these technological tools to create a controller application.

## 4. Tools and Technologies

### 4.1 Tizen Studio

The smartwatch application is developed by using Tizen Studio. It provides application development both in native language and as a web application. The computer and the smartwatch needs to be connected to the same network in order to run and test the application on the real device. It also provides emulators that are never used in this project. All tests are done on real device. It enables debugging the application over a web browser by inspecting the elements.

#### 4.2 Samsung Accessory Protocol and Samsung Accessory SDK

We needed the smartwatch to communicate with the android application which is called as Companion Application. This communication is established via Samsung Accessory Protocol which is implemented on both sides. In this protocol, one side is assigned as provider and the other side is in the role of consumer. In this project, Tizen side application is chosen as Provider since it provides the sensor data and Android side as Consumer since it uses the data provided by the Tizen application. Both sides are agreed on the communication details via an xml file. The communication search for peers via Bluetooth and the communication is established via Bluetooth. Then, Socket communication is established between and data is sent via channels from one side to another.

#### 4.3 Android Studio

Smartphone application is developed by using Android Studio.

#### 4.4 Parrot Bebop 2 Drone

This is the drone that we used to control in this project. We chose this because of its clear and rich documentation and stability.

#### 4.5 Samsung Note 4

Android side application is developed to be compatible with Samsung Note 4 which runs an Android version of 6.0.1

#### 4.6 Parrot SDK 3 for Bebop 2

This SDK enabled us to reach, understand and implement the communication with the drone.

#### 4.7 LibGDX & LibGDX Freetype Extension

Libgdx is an open source cross platform game engine frequently used for 2D game development. Core SDK and SDK for android development are used. Freetype extension enables rendering truetype text.

#### 4.8 Google GVR SDK

SDK developed by Google to create VR applications on Android.

#### 4.9 OpenGL ES 2.0

Rendering library that is enabled by default in Android.

## 5. Solution Methodology

### 5.1 Smartwatch Application Development

One of the application fields of this project is to develop a smartwatch application to pilot drone with hand gestures. For this purpose and in this scope, Samsung Gear S3 Smartwatch is used.

### 5.2 Android OpenGL ES & Cardboard Implementation

OpenGL ES usage in Android is fairly extensive and there are very differing approaches to set and an display an OpenGL context. For VR GUI, main concern is displaying a video (drone camera feed) together with overlaying display elements commonly used in aircraft HUDs, as seen in figure below:

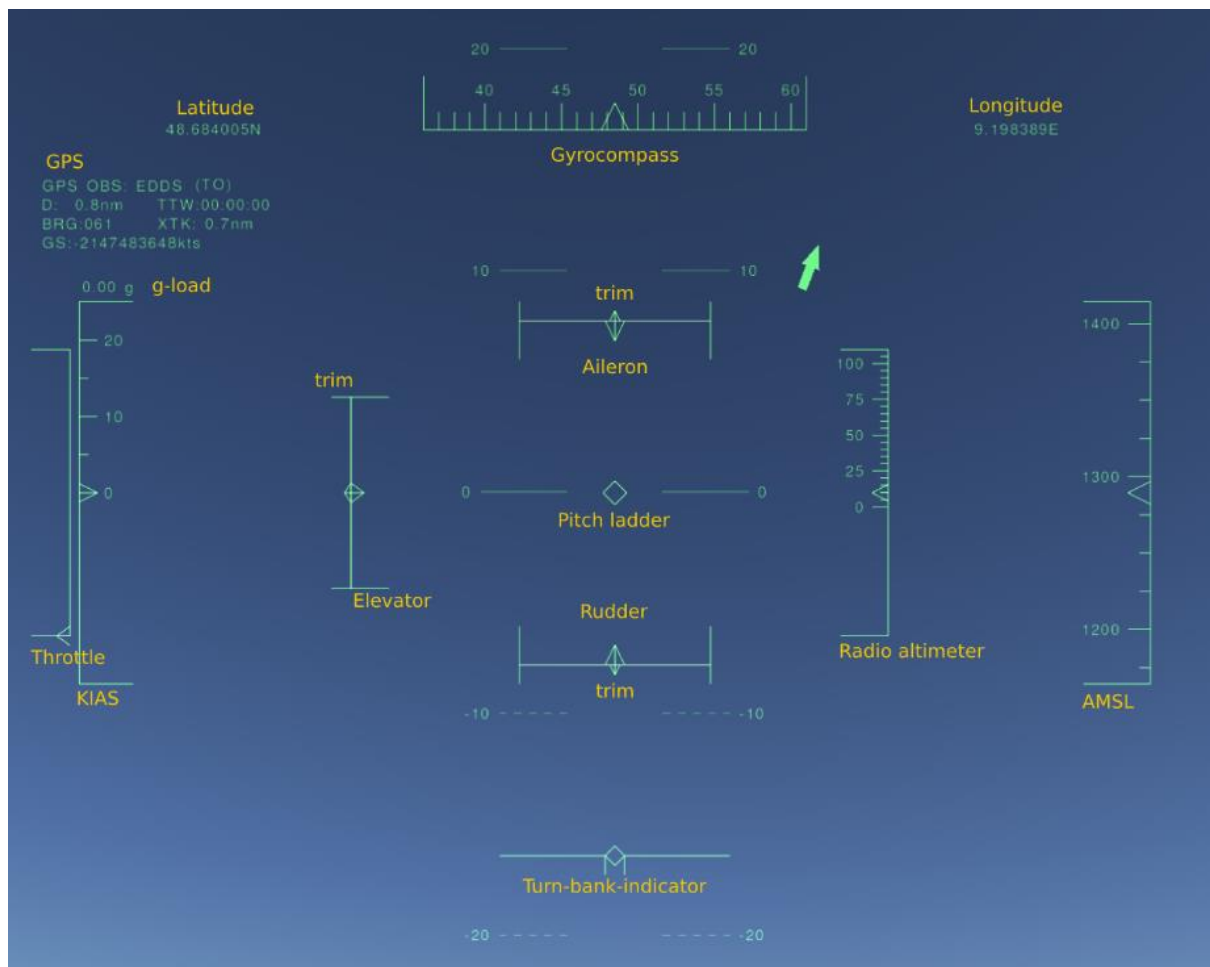


Figure 1: Head Up Display Elements

HUD display elements mainly consist of basic polygonal shapes and lines with dynamic and static texts. Filled shapes are not preferred as overlaying elements should not interfere or obscure the background. Therefore, approach to display a VR application should utilize basic OpenGL polygon rendering functionalities and basic text rendering.

VR display differs from traditional HUDs as a way to render the underlying video is needed. For traditional HUDs display is drawn on a transparent glass, and therefore such concerns are nonexistent. At the end, all of the symbologies should be multiplied for left and right eyes for the display to be VR-ready.

General methodology to meet all these requirements is to divide the work to three parts: displaying video, displaying symbologies and make required changes to create a stereo display (two displays for each eye). Performance improvements should be focused in first two parts, as third part is more or less done with Android Gvr SDK built-in functionalities.

## 6. Experiments and Solutions

### 6.1. For Tizen Smartwatch

The operating system the Gear runs is Tizen. Tizen supports application development in both its native language and as a web project. The incapability of drone SDK over native application and our familiarity with web project development, we preferred it as our solution methodology.

We experienced both communicating drone with the watch directly and using the android application in the middle to communicate, but we preferred the android application. Because we already have a study field in developing android application to control drone which already implements drone sdk. And also its incompatibility on smartwatch lead us to go with such solution. In order to communicate with the drone, we used Android application as the mediator. Communication is established with Android application over Bluetooth first and delivered the drone via Wi-Fi connection.

Tizen calls it as Companion application where the watch communicate with an android application. The communication is done via Samsung Accessory Protocol(SAP). It is implemented on both sides. Samsung Accessory SDK is installed on android side. Communication is done via Socket implementation over channels. Our solution requires collecting data from sensors and send events from the watch to drone. We tried to achieve that via sending data over specific channels. But we couldn't achieve that and couldn't find a solution. Then we changed the data structure to be sent and all data is sent via the same and only channel. Each data is tagged with a unique identifier and this tag is parsed on android side to handle different data transmission.

We also experienced reading data from different sensors that the device provides but were unable to detect and identify the movement that facilitates the drone movement. We finally reduced its scope to read gravity sensor data only and define movements with the help of that sensor only. The Low Pass Filter algorithm explained in the following sections is also applied to the data received from the watch.

We performed our tests with the smartwatch worn on the left wrist. The left arm is located straight and stretched parallel to ground. With its movement to up and down, the drone is moved forward and backward. With the arms rolled on left and right, the drone is moved to left and right. With the rotary detection on the smartwatch, the drone is either rotated or moved up and down.



Figure 2: SmartWatch Application Screens

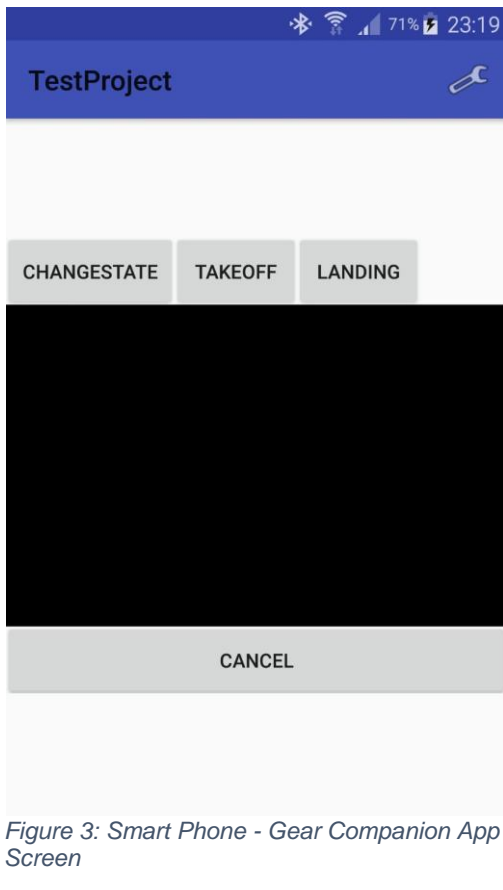


Figure 3: Smart Phone - Gear Companion App Screen

Smartwatch application consists of three screens which are shown above. First screen is the main screen that is providing access to Settings and Fly screens. In the Settings screens, a list of settings can be found such as Maximum Altitude, Maximum Vertical Speed. When FLY button is clicked, the settings are sent and the third screen is accessed. It has a Take Off button which turns to Land button with a red background once the drone is taken off. The sensor data is sent while this screen is displayed and movement is performed as defined above. Some demonstration videos can be found in the Repositories section.

Android side companion application screen is added on the left. Black part consists of the video received from drone camera. It is open in Camera LookUp mode in which camera angle can be arranged with arm movements. Change state changes mode to piloting where the real piloting experience takes place. It has other control inputs as well in case of an emergency situation if needed.

## 6.2. Low Pass Filter for Sensor Data and Drone Control

Capability of controlling drone with sensors is one of the requirement of the project. It is not a new thing that reading sensor data of smartwatch or smartphone but using these sensors to control the drone is new for most developers. As we expected to control our drone safely and accurately we should interpret these sensor values as 3 dimensional directives. To achieve this we use low pass filter.

Low Pass Filter Formula used for drone controlling is provided above:

$$\text{history}[\text{index}] = \text{coeff} * \text{history}[\text{index}] + (1f - \text{coeff}) * \text{newValue};$$

where coeff can be changed by preferences. This formula takes sum of old and new value and creates an output. Since this output depends on old value, new sensor value does not dominate and does not abruptly rise change the speed, tilt, yaw etc. of drone.

### Experiment 1:

Reading Smartphone Sensor Data: Reading sensors data through SensorEventListener and controlling drone with these raw sensor data. To achieve this new activity is created and drone connection established.

### Experiment 2:

Low Pass Filtering Sensor Data: Reading sensors data through SensorEventListener before filtering them with use of Low Pass Filtering and controlling the drone with these data. To achieve this new activity created and drone connection established.

**Results:** These two experiments are conducted for determining how low pass filtering affect drone control so results of these experiments are piece together as one result. Our results shows that Experiment 1 data change as above when android phone placed landscape mode parallel to user and rotated until parallel to sky user in a given time T:

Real Sensor Data	Experiment 1	Experiment2
0	0	0
1	1	0.2
2	2	0,56
3	3	1,048
4	4	1,6384
5	5	2,31072
6	6	3,048576



7	7	3,8388608
8	8	4,67108864
9	9	5,536870912

*Table 1: Low Pass Filter Experiment*

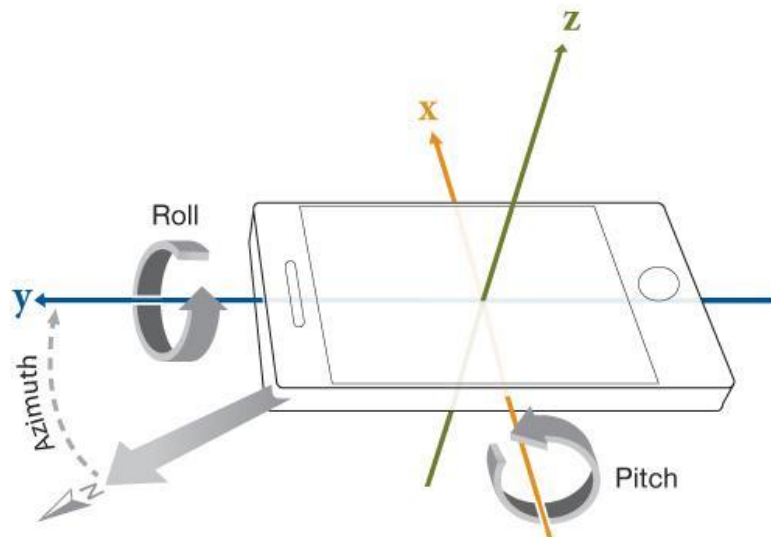
Above table display values for where coefficient equals to 0.8.

These results shows that in a given time real sensor data makes drone control difficult because it cannot eliminate unintentional movement of user. It is directly set drone movement angle whenever Real Sensor Data changed. But in low pass filter case we can eliminate short movement and natural reactions of human by filtering the data. If user really insist on particular movement at some point old data will be changed in a way that user intended to.

These results are tested on drone and safe and accurate piloting is achieved.

With use of these filtering and smartphone sensors Drone successfully controlled. Androids TYPE\_ACCELEROMETER sensor is used for this purpose which measures the acceleration force in  $m/s^2$  that is applied to a device on all three physical axes (x, y, and z), including the force of gravity. This sensor is a hardware sensor so it is independent from OS version of device which is an advantage. Most of the android devices provides this hardware sensors which makes this controller compatible with almost all the devices can be found in the market.

How controlling can be done is simply described below:



*Figure 4: Phone Sensor axis*

**Drone Move Forward and Backward:** To move drone forward or backward z axis of accelerometer sensor of device is used which provides data in range [-9.0, 9.0]. After applying low pass filter to obtained data from device, interpolation is applied to data hence drone requires data in range [-100, 100]. This interpolated and filtered data is send to drone and drone moves.

**Drone Move Left and Right:** To move drone left or right y axis of accelerometer sensor of device is used which provides data in range [-9.0, 9.0]. After applying low pass filter to obtained data from device, interpolation is applied to data hence drone requires data in range [-100, 100]. This interpolated and filtered data is send to drone and drone moves.

In application after selecting Vr controlling option below page showed to pilot.

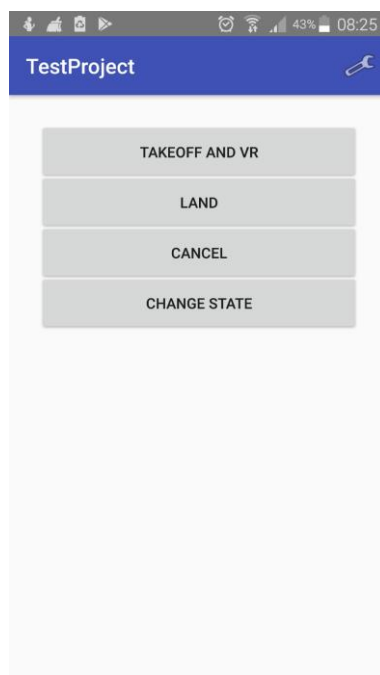


Figure 5: VR Controller Main Screen

After selecting Takeoff and VR button VR experience will be started and VR screen provided to pilot.



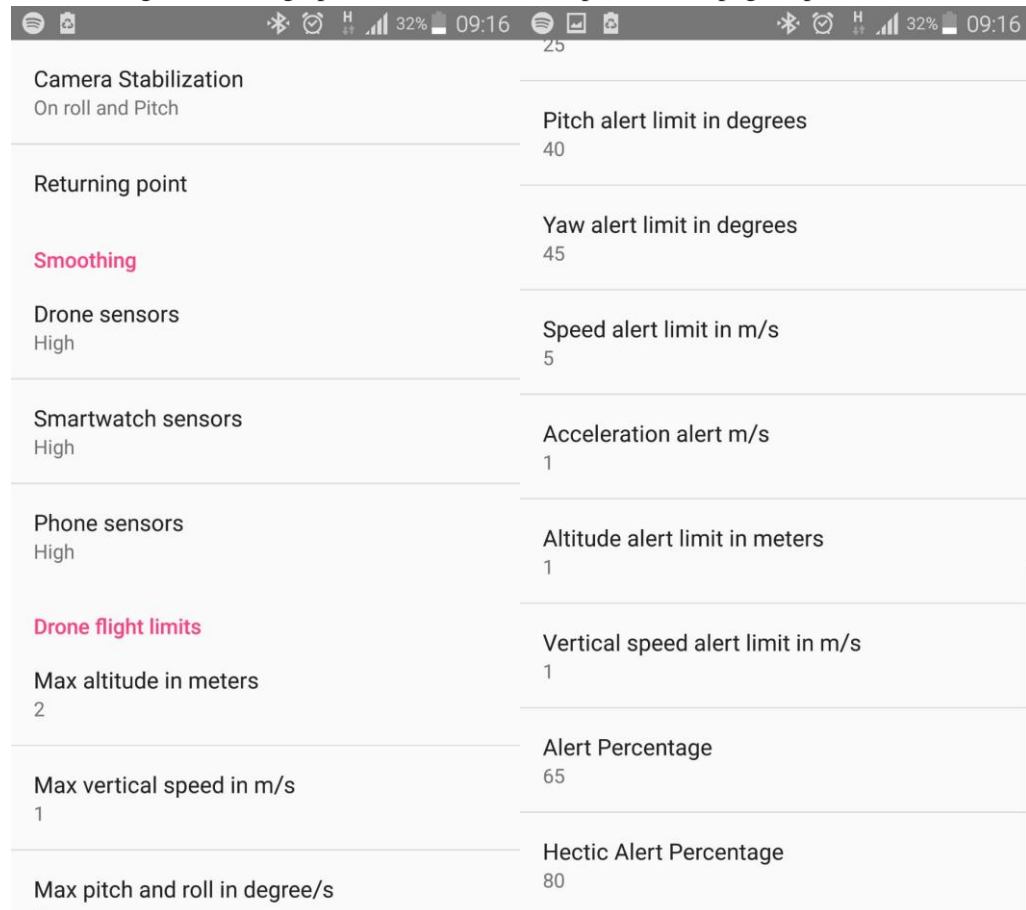
Figure 6: Drone Camera View on VR

## Controlling:

- Now Pilot can control Drone with head movements with use of motion sensors as described.
- By pressing volume up piloting mode and camera mode can be switched between. By pressing
- By pressing volume down takeoff and landing commands can be sent to drone

## Controlling and Drone Options:

In order to change controlling options and drone limits preferences page is provided to user.



Camera Stabilization On roll and Pitch	Pitch alert limit in degrees 40
Returning point	Yaw alert limit in degrees 45
Smoothing	Speed alert limit in m/s 5
Drone sensors High	Acceleration alert m/s 1
Smartwatch sensors High	Altitude alert limit in meters 1
Phone sensors High	Vertical speed alert limit in m/s 1
Drone flight limits	Alert Percentage 65
Max altitude in meters 2	Hectic Alert Percentage 80
Max vertical speed in m/s 1	
Max pitch and roll in degree/s	

Figure 7: Settings Menu

All the options provided can be set by pilot.

### 6. 3. Experiments & Results for VR Display

Android Gvr SDK is used for creating a stereo VR display, and there is no alternative currently available. Therefore, usage of this library is the main constraint. Gvr SDK directly uses OpenGL ES rendering approach: it prepares an OpenGL context and provides an entry for the developer to make custom opengl based rendering. At first glance, this is sufficient for drawing simple polygons but since opengl does not provide a way to display fonts directly, alternative approaches are needed. It is, however, is sufficient (and very efficient) for displaying drone camera feed. All received video frame can be directly binded to an opengl texture.

There are two performance bottlenecks for displaying the camera feed: Resolution of the source video and resolution of the texture on phone screen. Resolution on the phone screen cannot be changed, as it covers the whole display area for achieving an immersive experience. Therefore, focus to speed video rendering up is on efficiently handling source feed. To that end, following approaches are utilized:

1. Video stream settings on Parrot Bebop drone SDK are altered to receive a video in a low resolution.
2. Recording the video from the output buffer (send by the drone) is written directly to the opengl texture without any extra copying
3. Writing on the opengl texture is done in a separate thread, while rendering is done on main opengl rendering thread.

At first, third approach resulted in corrupting of the texture opengl thread tried to display the texture while video decoder thread wrote onto it. This is fixed by making the texture synchronized between the threads.

A very visible performance hit was observed while displaying the overlaying GUI symbologies:

First approach for displaying the GUI was to use native android GUI library consisting of "views". Views range from lists to buttons or labels: common symbologies seen in many android applications. Since android uses a separate OpenGL context (and thread) to draw views, an injection to this process is required.

One way to achieve this is to draw contents of views to an OpenGL texture instead of phone screen. After this process, texture can be used like any other texture in OpenGL and drawn on top of background video. This process, however, results in a heavy performance hit as even drawing a single pixel with this process results in a texture with the full screen size; all transparent save that single pixel. Therefore, after drawing a full screen sized camera feed, another full screen sized GUI was drawn.

Android native GUI views are flexible, fully supported and extensive. It is also very easy and relatively less time consuming to create a GUI using these elements (With android studio drag & drop functionality). However, since they are not efficient to use for this case, they were dropped.

Second approach to use OpenGL ES 2.0 directly. This approach is very efficient, but since library is very low level and does not directly have text rendering support, this approach was dropped as well.

Third approach is to use Unity for Android. Unity is a mostly 3D game engine, widely known and used both for smartphone game development and for PCs. Unity also have plugins for displaying VR content. Google GVR library is also supported. However, this approach is not usable for the project as it requires integration of multiple very different libraries (Tizen for smartwatches and Parrot Drone SDK). Unity projects tend to be heavily centered on Unity environment.

Fourth approach is to use LibGDX, an open source game engine for 2D gaming development. LibGDX is very efficient especially for drawing 2D texture heavy applications. It can also be easily integrated to an existing android project. For integrating this library to an android cardboard application, Libgdx-CardBoard-Extension by yangweibh is used (see Repositories section). LibGDX cannot be used for displaying the camera feed as it is a cross-platform library. For displaying the camera feed, an external texture should be defined. External textures are platform-specific.

After testing out different approaches, using libGDX for GUI rendering and OpenGL ES 2.0 for displaying background camera feed.

Final product is an android activity given below (display for one eye is added, screen displays two such screens for each eye):

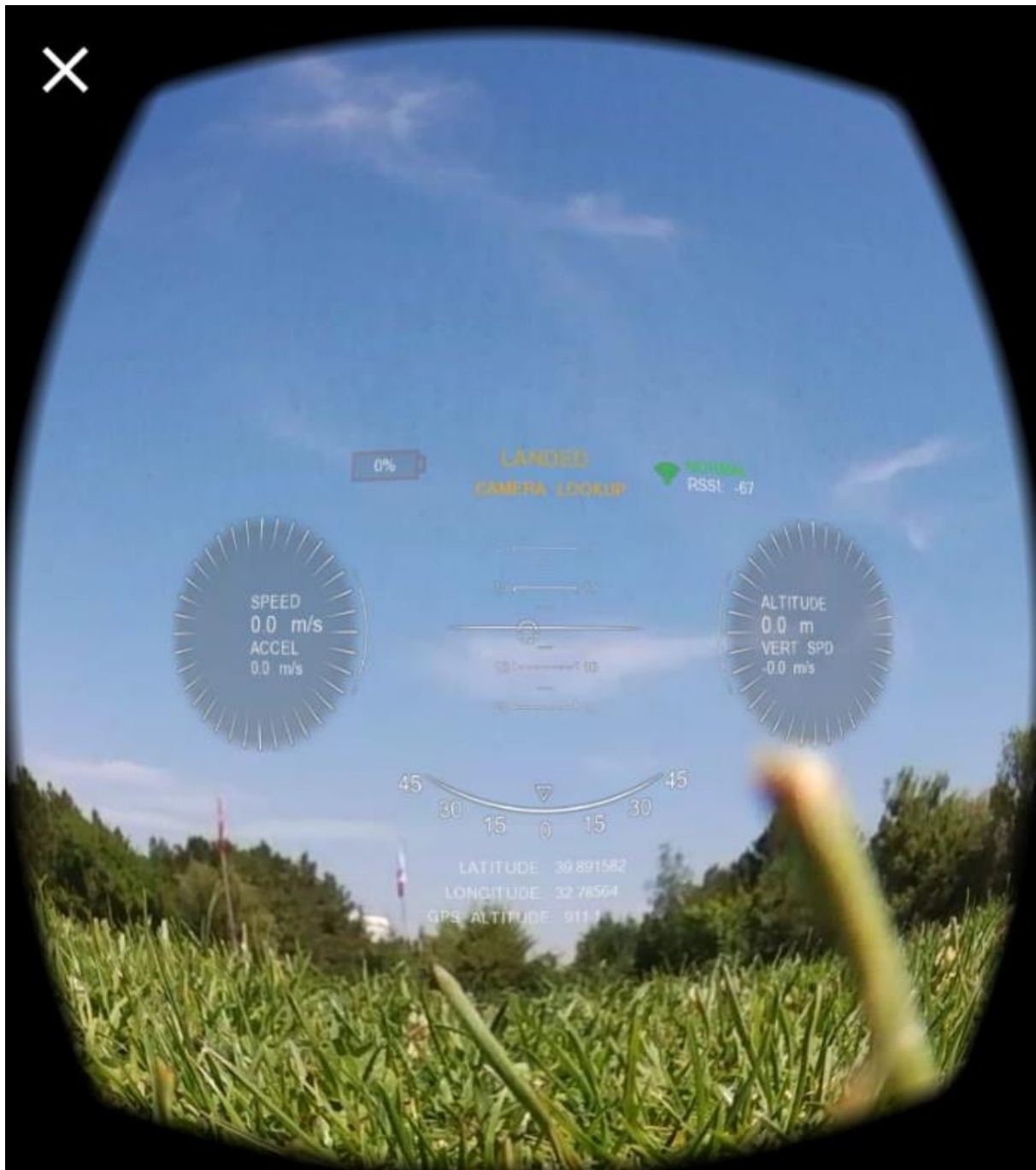


Figure 8: HUD View on VR

User's point of view roughly consists of the bounding box of all the drawn symbologies. Symbologies include a speed ring (left). Detailed overview of the symbologies is given below:





Figure 9: HUD Symbology Items

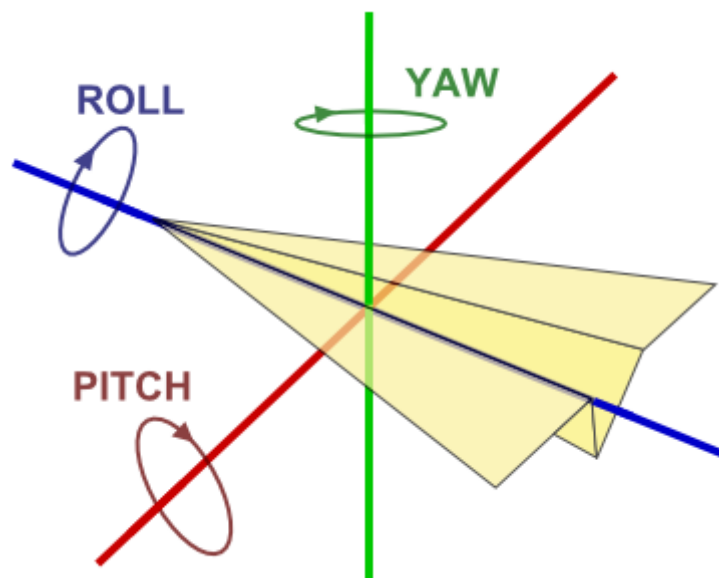


Figure 10: Directions on Drone Illustration

1. Pitch ladder. Displays degree of the drone nose according to horizon. Ranges from -90 degrees to 90 degrees with an horizon bar at 0 degree. There are bars at every 10 degrees and small bars for every other 10 degrees.
2. Roll: The triangular pointer rotates according to drone's roll value.
3. Yaw pointer: Changes according to yaw value. When it is on left and right borders, yaw value is -90 degrees and 90 degrees, respectively.
4. Current speed in m/s. Uses x and y values of speed received from the drone.
5. Current acceleration using 4.
6. Speed rings. These rotate according to speed (higher the speed increase, faster they rotate).
7. Acceleration pointer. Moves at the ring border (between the topmost and bottommost points of the circle) according to 5.

8. Altitude and vertical speed values. Similar to 4, 5, 6 and 7.
9. Battery level percentage. Blinks when battery is low. Color coded with green/orange/red according to battery level.
10. Current flight state. Ranges between EMERGENCY LANDING, EMERGENCY, USER TAKE OFF, TAKING OFF, MOTOR RAMPING, LANDING, LANDED, HOVERING, FLYING.
11. Current control state. PILOTING or CAMERA LOOKUP. On piloting mode, user input moves the drone. In camera lookup mode, user input moves the camera while drone stays still.
12. Current wifi signal strength. Color coded as green/orange/red according to connection strength. RSSI value is shown as well as labels: DISCONNECTED, POOR, NOT GOOD, NORMAL, VERY GOOD, AMAZING.
13. Coordinates and elevation received from GPS and barometric altitude sensor.

Elements 1, 2, 3, 4, 5, 6, 7, 8 have alert states: NO ALERT, ALERT and HECTIC ALERT. When in NO ALERT mode, elements are drawn normally.

In alert states, elements turn red, blink and zoom in/zoom out. When in hectic alert state, element blinks between 20% to 100% transparency and resizes between 90% to 110% of its original size within a 0.5 second period. In alert state, transparency is between %40 to %100 and resize is between %95 to %105 in 0.75 second period. Hectic alert turns the element color completely red while alert turns it to a red/orange tint.

For each element, there is a config in settings menu that sets the a related limit value as well as general settings for entering hectic alert and alert states percentages after which alerts are shown. For example, user may set speed limit as 10 m/s and then set alert limit percentage to 65% and hectic alert limit to 80%. When drone speed reaches 6.5 m/s, speed element enters alert state. When speed reaches 8 m/s, it enters hectic alert state.

#### 6. 4. Experiments & Results for Drone Flight Path

Although is not a part of the final application, research on making drone follow a predefined path is also done. It is found that Parrot drones are compatible with MavLink, a messaging library defined for drones and similar small aircraft. It defines a file-based message sequence in which each line corresponds to a command (going to a latitude / longitude or a relative position, approaching an altitude, rotating for a given time interval etc).

An android application can be created to present user a menu to populate the MavLink file. This file then can be loaded to the drone to make it follow the given commands.

## 7. Discussion and Conclusion

At the end of this project, we have experienced many technologies and tools which are both new to us and/or we are already familiar with.

## 8. References

1. <https://developer.samsung.com/gear/develop>
2. <https://developer.samsung.com/gear/develop/creating-your-first-app/web-companion/setup-sdk>



3. <http://qgroundcontrol.org/mavlink/start>

## 9. Repositories:

1. <https://github.com/obilgen35/tizendronecontroller>
2. <https://github.com/orhuncng/Bebop-Ui-Test>
3. [https://github.com/obilgen35/tizendronecontroller/blob/master/drone\\_watch\\_controller\\_1.mp4](https://github.com/obilgen35/tizendronecontroller/blob/master/drone_watch_controller_1.mp4)
4. [https://github.com/obilgen35/tizendronecontroller/blob/master/drone\\_watch\\_controller\\_2.mp4](https://github.com/obilgen35/tizendronecontroller/blob/master/drone_watch_controller_2.mp4)

Third party repositories used:

1. <https://github.com/libgdx/libgdx>
2. <https://github.com/yangweighbh/Libgdx-CardBoard-Extension>