Voice Controlled Drone

System on Android

By:

Aldwin Akbar Hermanudin

Rudy Nurhadi

Tomi

ELECTRICAL ENGINEERING DEPARTMENT

FACULTY OF ENGINEERING

UNIVERSITY OF INDONESIA

2015

PREFACE

First, let us send the highest bless to God who always give His mercy to us. Second, we would like to deliver the special thanks to our lecturer, especially Prof. Riri Fitri Sari who have given us advices, motivations, and supports. Third, we say thanks to all the people who gave their support to us, either morally or physically and either directly or implied support. Because without all of them, Voice Controlled Drone on Android will not be perfectly done.

Basically, our project, "Voice Controlled Drone on Android" is a project that integrate a cellular device and a system, to control a drone with human voice and monitor the drone status. The device we use is Smart Phone with Android Operating System.

For creating this Voice Controlled Drone on Android, we worked in team, we do trust our team to finish this system based on each role. We know that in this system we have a lot of mistakes, so we really need users' comments to build better system in the next step of development.

TABLE OF CONTENT

INTRODUCTION

A. Background

In this modern era, we have to admit that technology now could support people's daily work, in every aspects, naming education, health, food, entertainment, energy, and civil. Technology has known could increase value of life. Nevertheless, functionality of technology not just contributes to those areas, technology which is always developing and upgrading, can take part in making humans task easier.

The more futuristic of Industry properties in the world, could cause the importance of property user to adapt with technology, either technology as a need or technology as a 'decoration' of life. Technology as a need for example in security reasons. Drone is technology that was made either for security purpose or monitoring purpose. Drone can be used to do task that humans can't do such as monitor the sky or the ocean. There are different use of drones based on what the drone capabilities is built for.

In the present time, people really need technology, especially technology that integrates with a human voice. The concept of technology entertainment and sophisticated technology can support human beings, in monitoring aspect. The use of Drone as monitoring and commanding with human voice is a technology that can help monitor task easier.

B. Goals

- Implementing and integrating Java Android application
- Take part in drone development technology
- Contributing to support and make human beings' tasks easier.
- Implement google API

C. Objectives

Voice Controlled Drone Based on Android has some objectives. Namely to finish our class project, which is software engineering, and to contribute to our nation with the knowledge we have obtained from University of Indonesia.

D. General Description of Software

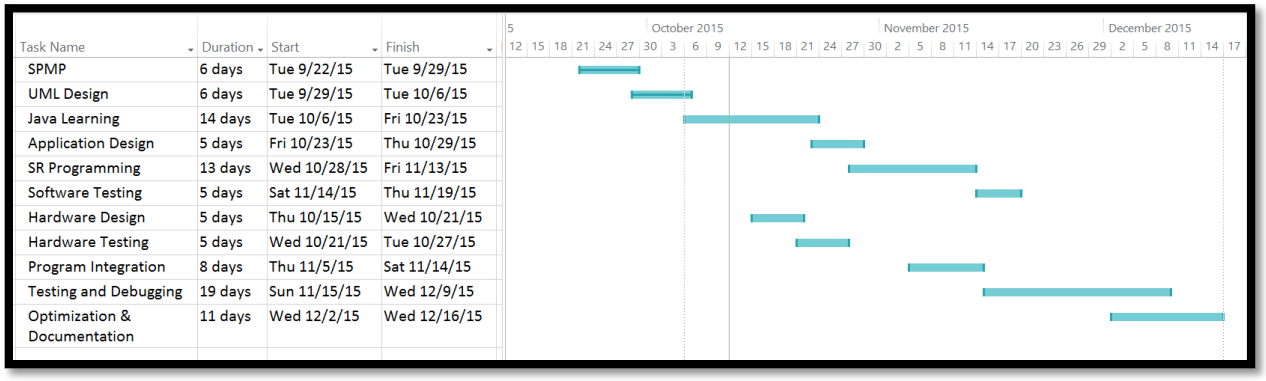
Product details:

The title of this project is “Voice Controlled Drone”. This software is provided for security reasons. In order to make UI a secure environment for student, a scout drone need to be deploy around UI. This drone have the capabilities in scouting a certain area where people cannot reach by foot. University of Indonesia is currently surrounded with a big forest with an uneven road. This is a difficulties for security staff to find out what is going on in the forest during the day. The main reason this drone is made is to help them scout the forest of UI. The drone will have a capabilities of doing missions with voice recognition command and manually override. The voice recognition software platform will be in Android. Using a 3D printer a self made phone docking is used as docking of your phone on your hand, this will be optional. The phone will literally hear your voice and convert it to string and sent it to the drone to do what you ask to. Your phone will show you what your drone camera pointed to and using the voice command to tell your drone to do missions.

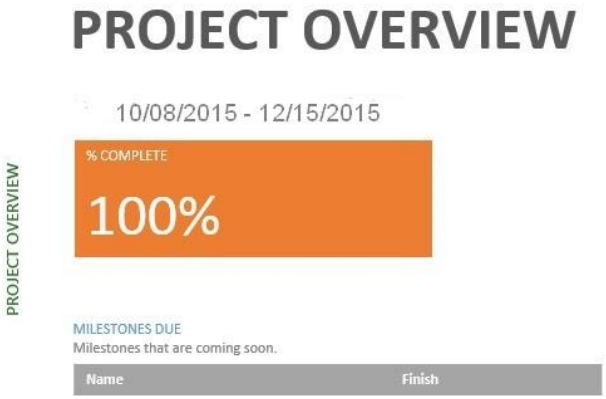
PROJECT MANAGEMENT

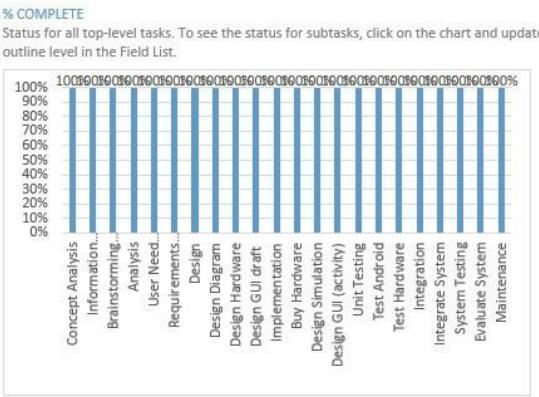
A. Timeline

To finish our project perfectly, we need a guidance and a software manager to make us keep in line and focus as we run the project. We can make a timeline as our guide-plan. The timeline mentioned before was created using Microsoft Office Project Professional 2016. By using this software, each person can participate building this project with much more efficient and structured.



Task Name	Duration	Start	Finish
SPMP	6 days	Tue 9/22/15	Tue 9/29/15
UML Design	6 days	Tue 9/29/15	Tue 10/6/15
Java Learning	14 days	Tue 10/6/15	Fri 10/23/15
Application Design	5 days	Fri 10/23/15	Thu 10/29/15
SR Programming	13 days	Wed 10/28/15	Fri 11/13/15
Software Testing	5 days	Sat 11/14/15	Thu 11/19/15
Hardware Design	5 days	Thu 10/15/15	Wed 10/21/15
Hardware Testing	5 days	Wed 10/21/15	Tue 10/27/15
Program Integration	8 days	Thu 11/5/15	Sat 11/14/15
Testing and Debugging	19 days	Sun 11/15/15	Wed 12/9/15
Optimization & Documentation	11 days	Wed 12/2/15	Wed 12/16/15





B. Task Division

Our team works together to finish our system. Our team, which is consists of three people, divide ourselves into three main tasks, every person has his job and to do lists. We apply task division because in software engineering field, resource management, especially people management is important. So we divide the job based on each person's skill specialization. As a note, we need to emphasize that all stages of the task delegated by these individuals will go back through the process of discussion in group meetings.

Aldwin Akbar Hermanudin:

- Leading the implementation hardware devices.
- Responsible in Hardware Developing
- Creating UML diagrams with the explanation
- Responsible for maintaining device to satisfy consumers' need.

Rudy Nurhadi :

- Leading the implementation phase (programming)
- Creating UML diagrams with the explanation
- Responsible for completing the design needed (application notes).
- Contributing in java programming in Android OS.
- Responsible for evaluating the system to meet the consumer's need.

Tomi:

- Leading the management and documentation
- Responsible in Software designing
- Responsible for maintaining GUI to satisfy consumer's need.
- Contributing in java programming in Android OS.
- Creating UML diagrams with the explanation
- Responsible for making goals, plans, and documentation

C. Risk Management

We encounter problems and obstacles in the process of making voice controlled drone. This section was built to analyze the risk and figure the solutions to be implemented in the system. Those problems we have met are:

1. Time shortage and deadlines

- ✓ Probability: High
- ✓ Prevention: Take care about the schedule and focus while doing it.
- ✓ Correction: When tasks fail to be done in time, there must be a discussion between the team members in order to maintain the efficiency of work.
- ✓ Impact: High

2. Design Errors

- ✓ Probability: Medium
- ✓ Prevention: Critical reviewed for each development or design. Consult the problem with the capable advisor. Take a lot of critics that has positive impacts to this project
- ✓ Correction: Contact the advisor or people that has capacity over this to help us to do some design corrections.
- ✓ Impact: High

3. Miscommunication

- ✓ Probability: Medium
- ✓ Prevention: Every member of this project should be participated to every single internal meeting. After a meeting there must be some self-reviewed between all members about things that they like or dislike. The openness over all members is required. All members should not hesitate to ask and re-ask questions if things seem unclear.
- ✓ Correction: Between team leader and team member or project team must gathered in a meeting that only focus on the solution of the miscommunication problem
- ✓ Impact: High

4. The absence of Project Team Leader

- ✓ Probability: Low

- ✓ Prevention: Choose either team leader or team member that has to come to the meeting if both are not possible to come and asks them to keep alerting about the progress of the project schedule and deadline.
- ✓ Correction: To remind and give the last update over this project
- ✓ Impact: Low

5. Unavailability of the technical advisor when needed

- ✓ Probability: Medium
- ✓ Prevention: Find the most capable advisor during to do this project that focus only on the technical stuff and after that keep in touch with him/her in order to have a good communication and relationship between project team and that advisor.
- ✓ Correction: Contact another advisor that available meanwhile trying to solve the problem by searching it from different source. Due to time is not waiting us and deadline is coming through us.
- ✓ Impact: High

6. Lack of meeting intensity with the customer

- ✓ Probability: Medium
- ✓ Prevention: Meeting with the customer has to be planned well in advance.
- ✓ Correction: Meeting has to be rescheduled.
- ✓ Impact: Medium

7. The customer requirements are not possible

- ✓ Probability: High
- ✓ Prevention: Obviously explained to the customer why or why not to implement things that he/she asked for.
- ✓ Correction: URD has to be analyzed and further discussion are urgently needed before the project is started.
- ✓ Impact: Low

DESIGN

Designing is the process in which the team creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and regarding the constraints. Designing is the step of conceptualizing, planning, and understanding the foundation and process of the projects.

- **UML Diagram**

UML, which stands for Unified Modeling Language, is really important as the pre-process state. This object-oriented system of notation has been evolved from the works of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation. These renowned computer scientists fused their respective technologies into a single, standardized model. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling object oriented programs. UML defines many kind types of diagrams: class (package), use case, sequence, activity, and deployment.

1. Use Case Diagram

This diagram shows the user and what "Voice Controlled Drone" system provides for them. In this case, the application lets users sent data from voice to command the drone. The android system in the diagram below provided several things a user can do with the apps. The apps android will give you the access to monitor drone status the command the user have given to the drone. The monitor status is called the feedback if the command is successful. There are several uses of the apps to control the drone. The apps let you easily command your drone to record video from you drone, capture picture from the drone, command your drone through voice recognition whereas the voice is translated into a string of data and sent to the drone via internet. To anticipate any unwanted incident we develop one more uses to control the drone. A manual override system is intended to control the drone manually with your phone if the drone failed to get command from the voice recognition application.

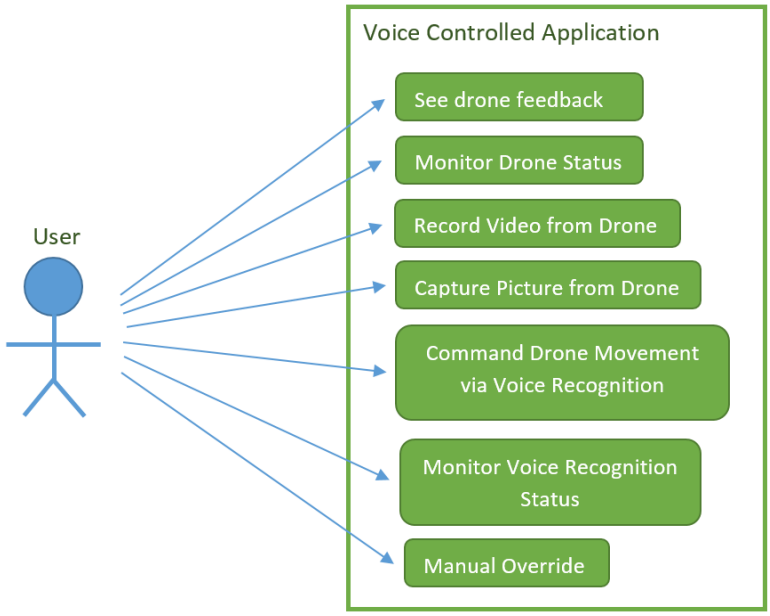


Figure 1. Use Case Diagram

2. Sequence Diagram

Sequence diagram below explains how the operation is performed by the application. Here is how the operation works.

Users will sent a data from his/her phone after user give commands using voice to drone via "Voice Controlled Drone" application, the software will translate the user voice into a length of string and sent it to the drone. Then drone will receive the data via Internet and do what the user request. The feedback from drone is request command the drone gets from user to show if the command is successfully executed.

The manually override system is easier to be deployed whereas the user can control the drone via the user phone. There will be buttons-like controller that the user can touch to control the movement such as moving forward, backward, up, down. The user can also do the control off taking off the ground and landing the drone safely. The manually override is used if user want to control manually or the drone voice recognition is not responding as it should be. This is to anticipate unwanted accident.

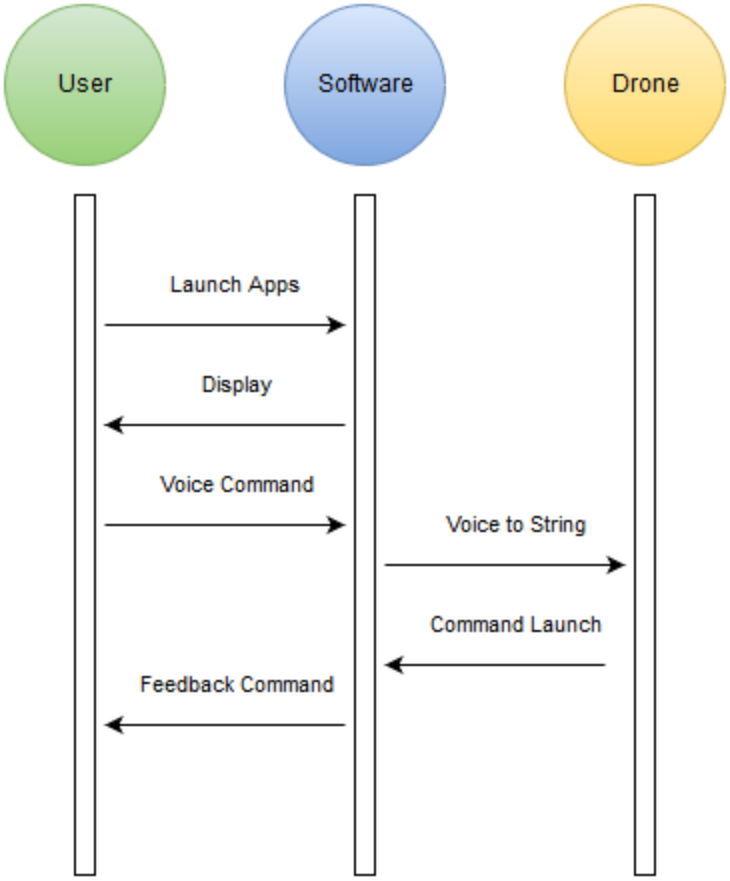


Figure 2. Sequence Diagram

3. Deployment Diagram

Deployment diagram illustrates the physical architecture of the system in terms of the hardware deployed on and the communication links between hardware nodes. By connecting via Internet, it is possible to deliver data from phone to drone. The android apps will be installed in an android IP-enabled device whereas the communication between hardware is through a TCP/IP protocol.

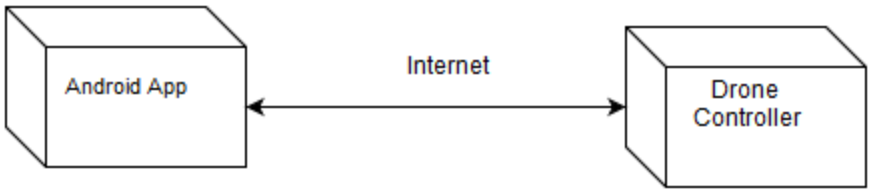


Figure 3. Deployment Diagram

4. Activity Diagram

This diagram describes the flow of activities or tasks. It resembles a flow chart. It has decision points and synchronization bars. The synchronization bars show activities that can happen in either order or even at the same time. “Voice Control Drone” activity diagram below explains the step-by-step activities provide by “Voice Control Drone”. First of all, the software will accept input from the user which is user voice command, and then the system will determine the voice command and translate it to a string. Secondly, the software will sent a data via internet to the drone. If the drone do not give feedback for 5 seconds, the command is consider failed. Otherwise if the command is received by the drone, the drone will sent feedback which is displaying command information that user has given in the beginning. Below is the diagram of Activity Diagram.

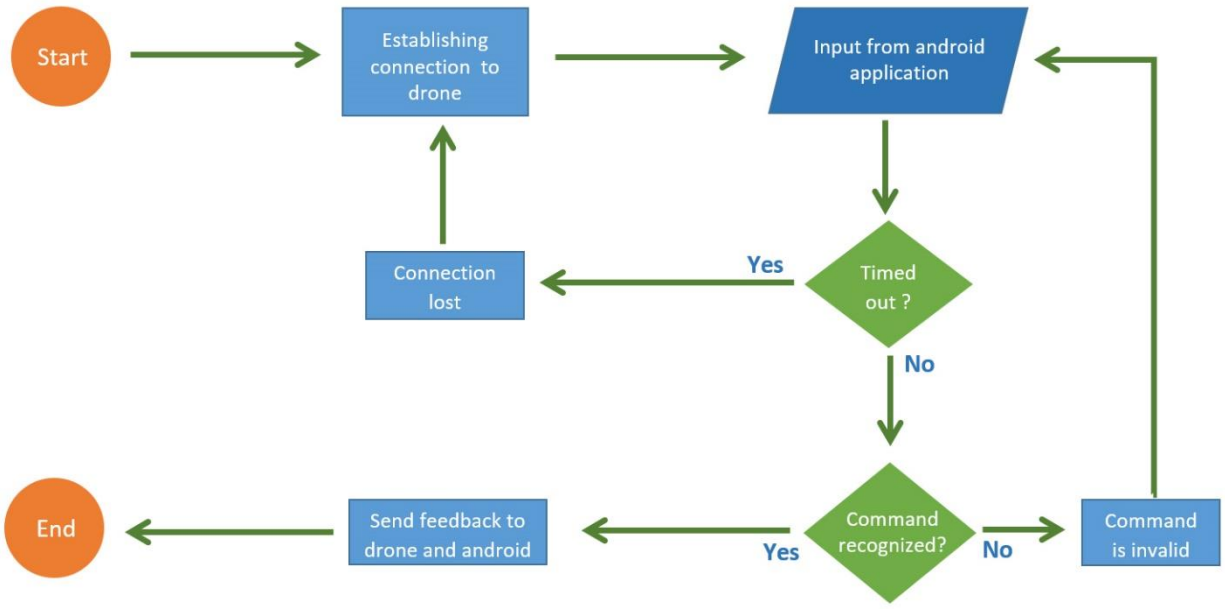


Figure 4. Activity Diagram

5. Class Diagram

This diagram explains the class of the program and describes the interconnection between the classes. These class is implemented in the drone. There are two main class in the system between hardware and software. The software classes is provided in display class. In the display the user can see the status of the drone and the command status. There are three main options in the android. There are video class where you can record video and capture picture if you choose this option, using the voice option let the user to edit/add command and sent command using voice, and the about options give you credits and how to use the application. The hardware drone between application communication use the protocol of TCP/IP. This protocol lets you communicate with your phone to control the drone.

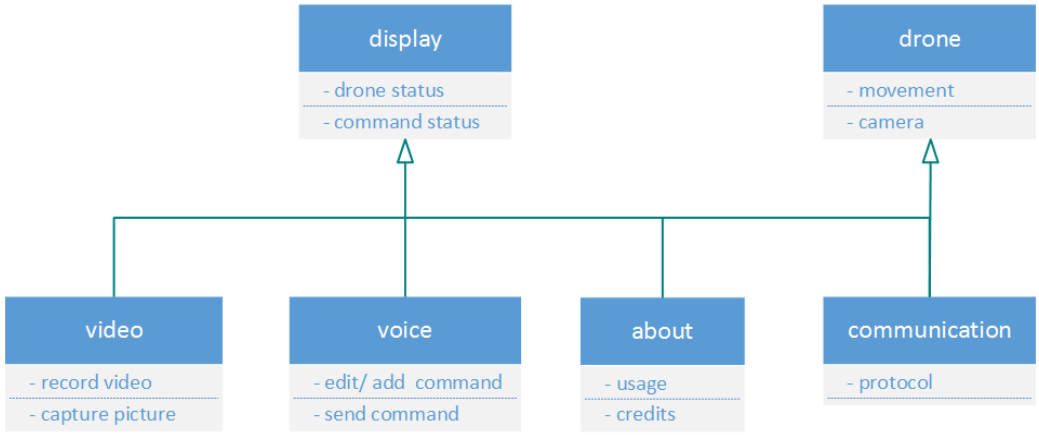


Figure 5. Class Diagram

6. Communication Diagram

Communication diagram was called collaboration diagram in UML. It is similar to sequence diagrams but the focus is on messages passed between objects. The same information can be represented using a sequence diagram and different objects. The diagram provided in communication can be explained the same as sequence diagram where you can find the explanation in the sequence diagram.

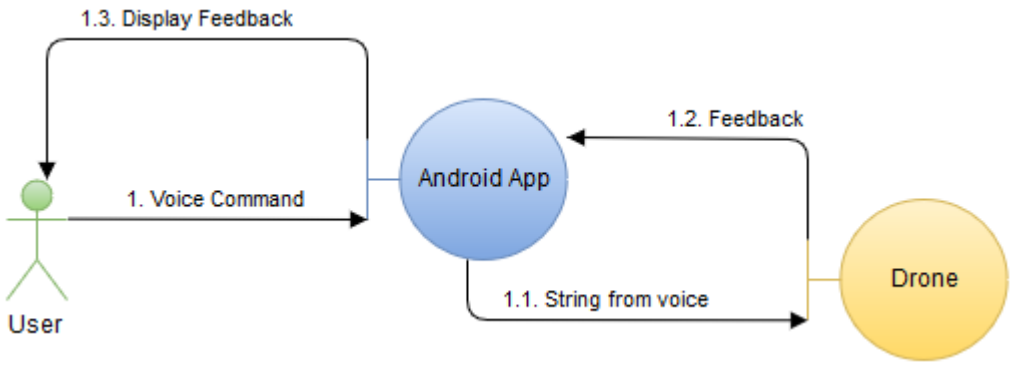


Figure 6. Communication Diagram

7. Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems that has many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. In the diagram provided below show the main android application software system is link to google location API and google speech to text API.

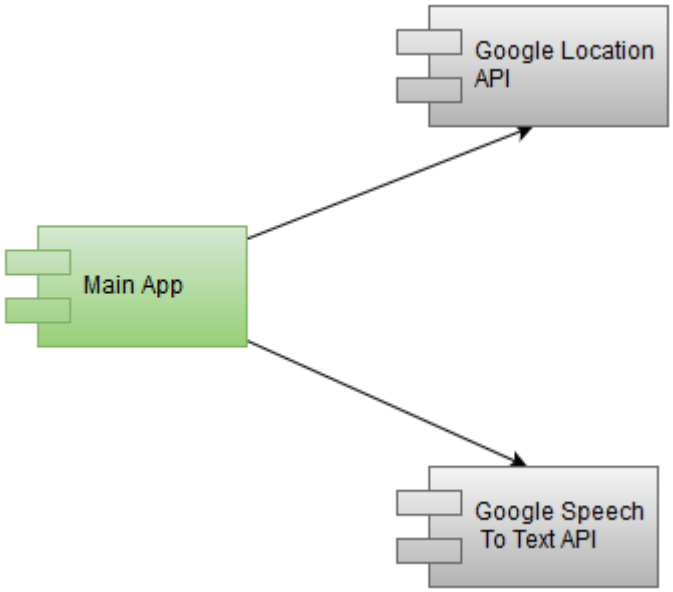


Figure 7. Component Diagram

IMPLEMENTATION

A. Software Specification (Graphical User Interface)

Android Operating System

This software is made for the Android operating system. Here is a quick primer on Android, Android is an operating system developed by the Open Handset Alliance, which is aimed at mobile devices such as, Tablet Computer and Smartphone. In its development, Google took over Android and made the android to be open-source Apache license. Since then it has a lot of communities around the world who participated in the development of android, it is estimated there have been 300,000 applications in the Android Market, which is a third party, where users can download the application free or paid. Android was built from the Linux kernel with a library, API, and middleware, which is programmed with C. In operation, the android framework using Java-Library that implements the Dalvik Virtual Machine, in which the coding of each program will be compiled only when specific coding instruction is executed. This is to improve the efficiency of memory usage. Android is an open-source community-based so that any party may contribute in the development. Therefore, there is one organization that is the Android Open Source Project / AOSP, which is assigned to conduct a routine process of maintenance and android development. AOSP has a goal to improve the experience for end-users in utilizing the mobile device and improve the compatibility of android on every installation. From many versions of Android that has been developed by AOSP, our application targets API level 11-19. API level 11 works on Android 3.0 platform (Honeycomb) until Android 4.4 platform (KitKat).

a. Android 3.0/3.1 version (Honeycomb)

Android Honeycomb designed specifically for the tablet. This Android version supports larger screen sizes. User Interface on Honeycomb is also different because it was designed for the tablet. Honeycomb also supports multi-processors and hardware acceleration (hardware) for graphics. The first tablet is made by running Honeycomb is the Motorola Xoom. Tablet devices with the Android 3.0 platform will soon be present in Indonesia. The device is called the Eee Pad Transformer production from Asus. Indonesia planned to enter the market at May 2011.

b. Android version 4.0 (ICS: Ice Cream Sandwich)

Announced on October 19, 2011, bringing Honeycomb features to Smartphones and adds new features including unlocks with face recognition, data network usage monitoring and control, integrated social networking contacts, photographic enhancements, search mail offline, and share information using NFC

c. Android version 4.1 (Jellybean, API level 16)

Google announced Android 4.1 (Jelly Bean) at the Google I/O conference on 27 June 2012. Based on Linux kernel 3.0.31, Jelly Bean was an incremental update with the primary aim of improving the functionality and performance of the user interface. The performance improvement involved "Project Butter", which uses touch anticipation, triple buffering, extended vsync timing and a fixed frame rate of 60 fps to create a fluid and "buttery-smooth" UI. Android 4.1 Jelly Bean was released to the Android Open Source Project on 9 July 2012, and the Nexus 7 tablet, the first device to run Jelly Bean, was released on 13 July 2012.

d. Android version 4.4 (Kitkat, API level 19)

Google announced Android 4.4 KitKat, internally known as Project Svelte,[121] on 3 September 2013. The release had long been expected to be numbered 5.0 and called 'Key Lime Pie'. KitKat debuted on Google's Nexus 5, and has been optimised to run on a greater range of devices, having 512 MB of RAM as a recommended minimum.

B. Programming Language

We used Java and XML programming language to build our Android application. Typically, Java will configure each application function and XML will gives the application interface. Here is the code for our applications:

1. Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="studio.android.art.artdrone3" >

    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
```

```
    </uses-permission>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.default" />
            </intent-filter>
        </activity>
        <activity
            android:name=".SecondActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
        </activity>
        <activity
            android:name=".ThirdActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
        </activity>
        <activity android:name=".Splash">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

    </application>

</manifest>
```

2. MainActivity

```
package studio.android.art.artdrone3;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.widget.Toolbar;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

import java.util.Objects;


public class MainActivity extends AppCompatActivity {
```

```
    // Declaring Your View and Variables
    Toolbar toolbar;
    public static ViewPager pager;
    ViewPagerAdapter adapter;
    SlidingTabLayout tabs;
    CharSequence Titles[] = {"Speech", "Video", "Status"};
    int Numboftabs = 3;

    public static ConnectTask connectTCP = null;
    public static TCPClient tcpClient = null;
    public static ARTDroneStatus droneStatus = null;
    static Bitmap bitmapCameraDrone;
    Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Creating The Toolbar and setting it as the Toolbar for the activity
        toolbar = (Toolbar) findViewById(R.id.tool_bar);
        setSupportActionBar(toolbar);

        // Creating The ViewPagerAdapter and Passing Fragment Manager, Titles
fot the Tabs and Number Of Tabs.
        adapter = new ViewPagerAdapter(getSupportFragmentManager(), Titles,
Numboftabs);

        // Assigning ViewPager View and setting the adapter
        pager = (ViewPager) findViewById(R.id.pager);
        pager.setAdapter(adapter);

        // Assiging the Sliding Tab Layout View
        tabs = (SlidingTabLayout) findViewById(R.id.tabs);
        tabs.setDistributeEvenly(true); // To make the Tabs Fixed set this
true, This makes the tabs Space Evenly in Available width

        // Setting Custom Color for the Scroll bar indicator of the Tab View
        tabs.setCustomTabColorizer(new SlidingTabLayout.TabColorizer() {
            @Override
            public int getIndicatorColor(int position) {
                return getResources().getColor(R.color.colorAccent);
            }
        });

        // Setting the ViewPager For the SlidingTabsLayout
        tabs.setViewPager(pager);

        droneStatus = new ARTDroneStatus();

        handler = new Handler();
        handler.post(new Runnable() {
            @Override
            public void run() {
                if (pager.getCurrentItem() == 1) {
                    if (tcpClient != null) {
                        tcpClient.sendMessage("vf");
                    }
                    handler.postDelayed(this, 35);
                } else if (pager.getCurrentItem() == 2) {
                    if (tcpClient != null) {
```

```
                        tcpClient.sendMessage("ds");
                    }
                    handler.postDelayed(this, 250);
                } else {
                    handler.postDelayed(this, 500); // set time here to refresh
textView
                }
            }
        });
    }


    public static class ARTDroneStatus {
        public static boolean armMode;
        public static String flightMode;
        public static float altitude;
        public static float compass;
        public static float velocityX;
        public static float velocityY;
        public static float velocityZ;
        public static float airTemperature;
        public static float airPressure;
        public static float battery;

        public ARTDroneStatus() {
            armMode = false;
            flightMode = "Land";
            altitude = 0;
            compass = 0;
            velocityX = 0;
            velocityY = 0;
            velocityZ = 0;
            airTemperature = 0;
            airPressure = 0;
            battery = 0;
        }
    }

    public static class ConnectTask extends AsyncTask<String, byte[],
TCPClient> {
        @Override
        protected TCPClient doInBackground(String... message) {
            tcpClient = new TCPClient(new TCPClient.OnMessageReceived() {

                @Override
                public void messageReceived(byte[] message, byte[] isImage) {
                    publishProgress(message, isImage);
                }
            }, Tab1.ipAddressEditText.getText().toString(),
Integer.parseInt(Tab1.portAddressEditText.getText().toString()));
            tcpClient.run();
            return null;
        }

        @Override
        protected void onProgressUpdate(byte[]... values) {
            super.onProgressUpdate(values);
            //receivedTextView.setText(values[0].length + "");
            if (Objects.equals(new String(values[1]), "I")) {
                bitmapCameraDrone = BitmapFactory.decodeByteArray(values[0], 0,
values[0].length);
                if (bitmapCameraDrone != null) {
```

```
                    Tab2.cameraDrone.setImageBitmap(bitmapCameraDrone);
                    bitmapCameraDrone = null;
                }
            }
            if (Objects.equals(new String(values[1]), "S") && values[0].length
>= 2) {
                if ((values[0][0] == (byte) 's' && values[0][1] == (byte) 'c'
&& values[0][2] == (byte) ':')) {
                    Tab1.receivedTextView.setText(new String(values[0], 3,
values[0].length - 3));
                } else if (values[0][0] == (byte) 'd' && values[0][1] == (byte)
's' && values[0][2] == (byte) ':') {
                    Log.e("ds", "ds");
                    int j = 3;
                    int k = 0;

                    for (int i = 3; i < values[0].length; i++) {
                        if (values[0][i] == (byte) ';') {
                            Log.e("ds", ";");
                            if (k == 0) {
                                droneStatus.armMode = (values[0][j] != (byte)
'0');
                                if (droneStatus.armMode) {
                                    Tab3.armMode.setText("Arm");
                                } else {
                                    Tab3.armMode.setText("Disarm");
                                }
                            } else if (k == 1) {
                                droneStatus.flightMode = new String(values[0],
j, i - j);

Tab3.flightMode.setText(droneStatus.flightMode);
                            } else if (k == 2) {
                                droneStatus.altitude = Float.parseFloat(new
String(values[0], j, i - j));
                                Tab3.altitude.setText(droneStatus.altitude +
"");
                            } else if (k == 3) {
                                droneStatus.compass = Float.parseFloat(new
String(values[0], j, i - j));
                                Tab3.compass.setText(droneStatus.compass + "");
                            } else if (k == 4) {
                                droneStatus.velocityX = Float.parseFloat(new
String(values[0], j, i - j));
                                Tab3.velocityX.setText(droneStatus.velocityX +
"");
                            } else if (k == 5) {
                                droneStatus.velocityY = Float.parseFloat(new
String(values[0], j, i - j));
                                Tab3.velocityY.setText(droneStatus.velocityY +
"");
                            } else if (k == 6) {
                                droneStatus.velocityZ = Float.parseFloat(new
String(values[0], j, i - j));
                                Tab3.velocityZ.setText(droneStatus.velocityZ +
"");
                            } else if (k == 7) {
                                droneStatus.airTemperature =
Float.parseFloat(new String(values[0], j, i - j));

Tab3.airTemperature.setText(droneStatus.airTemperature + "");
                            } else if (k == 8) {
```

```
                                droneStatus.airPressure = Float.parseFloat(new
String(values[0], j, i - j));

Tab3.airPressure.setText(droneStatus.airPressure + "");
                            } else if (k == 9) {
                                droneStatus.battery = Float.parseFloat(new
String(values[0], j, i - j));
                                Tab3.battery.setText(droneStatus.battery + "");
                                break;
                            }
                            k++;
                            j = i + 1;
                        }
                    }
                }
            }
        }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    boolean doubleBackToExitPressedOnce = false;

    @Override
    public void onBackPressed() {
        if (doubleBackToExitPressedOnce) {
            super.onBackPressed();
            return;
        }

        this.doubleBackToExitPressedOnce = true;
        Toast.makeText(this, "Please click BACK again to exit",
Toast.LENGTH_SHORT).show();

        new Handler().postDelayed(new Runnable() {

            @Override
            public void run() {
                doubleBackToExitPressedOnce = false;
            }
        }, 2000);
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        switch (item.getItemId()) {
            case R.id.action_settings:
                //noinspection SimplifiableIfStatement
                Intent startActivity = new Intent(this, ThirdActivity.class);
                startActivity(startActivity);

                return true;
            default:

                return super.onOptionsItemSelected(item);
```

```
        }
    }
}
```

3. SecondActivity

```
package studio.android.art.artdrone3;

import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;


public class SecondActivity extends AppCompatActivity {

    Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

    }
}
```

4. ThirdActivity

```
package studio.android.art.artdrone3;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

/**
 * Created by Tomi Lebrero on 11/30/2015.
 */
public class ThirdActivity extends AppCompatActivity {

    Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_third);
        toolbar = (Toolbar) findViewById(R.id.tool_bar);
        setSupportActionBar(toolbar);
    }
}
```

5. Tab 1

```
package studio.android.art.artdrone3;

/**
 * Created by hp on 25/11/15.
```

```
 */
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.PopupWindow;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;

/**
 * Created by hp1 on 21-01-2015.
 */
public class Tab1 extends Fragment {
    ImageButton speakButton;
    TextView commandTextView;
    public static TextView receivedTextView;
    public static EditText ipAddressEditText;
    public static EditText portAddressEditText;
    EditText languageEditText;
    EditText commandEdit;
    ImageButton sendCommandButton;
    Button connectButton;
    Button stopButton;
    String voiceCommand;
    ImageButton btnOpenPopup;

    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.tab_1,container,false);
        speakButton = (ImageButton) view.findViewById(R.id.speakButton);
        commandTextView = (TextView) view.findViewById(R.id.commandTextView);
        receivedTextView = (TextView) view.findViewById(R.id.receivedTextView);
        languageEditText = (EditText) view.findViewById(R.id.languageEditText);
        ipAddressEditText = (EditText)
view.findViewById(R.id.ipAddressEditText);
        portAddressEditText = (EditText)
view.findViewById(R.id.portAddressEditText);
        commandEdit = (EditText) view.findViewById(R.id.commandEdit);
        sendCommandButton = (ImageButton)
view.findViewById(R.id.sendCommandButton);
        connectButton =  (Button) view.findViewById(R.id.connectButton);
        stopButton =  (Button) view.findViewById(R.id.stopButton);
        btnOpenPopup = (ImageButton) view.findViewById(R.id.imageButton);

        speakButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
```

```
                intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

                String languageUsed = languageEditText.getText().toString();

                intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, languageUsed);

                try {
                    startActivityForResult(intent, 1);

                    commandTextView.setText("");
                    receivedTextView.setText("");

                } catch (ActivityNotFoundException a) {
                    Toast.makeText(getActivity().getApplicationContext(),
"Perangkat Tidak Mendukung", Toast.LENGTH_SHORT).show();
                }

            }
        });

        sendCommandButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String commandEditString = commandEdit.getText().toString();
                commandTextView.setText(commandEditString);
                receivedTextView.setText("");
                if(MainActivity.tcpClient != null) {
                    MainActivity.tcpClient.sendMessage(commandEditString);
                }
                else {
                    receivedTextView.setText("Not connected");
                }
            }
        });

        connectButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(MainActivity.tcpClient != null) {
                    MainActivity.tcpClient.stopClient();
                    MainActivity.tcpClient = null;
                    MainActivity.connectTCP = null;
                }
                if (MainActivity.connectTCP == null) {
                    MainActivity.connectTCP = new MainActivity.ConnectTask();
                    MainActivity.connectTCP.execute("");
                }
            }
        });

        stopButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(MainActivity.tcpClient != null) {
                    MainActivity.tcpClient.stopClient();
                    MainActivity.tcpClient = null;
                    MainActivity.connectTCP = null;
                }
            }
        });
```

```
        btnOpenPopup.setOnClickListener(new Button.OnClickListener() {

            @Override
            public void onClick(View arg0) {
                LayoutInflater layoutInflater = (LayoutInflater)
getActivity().getBaseContext()

.getSystemService(getContext().LAYOUT_INFLATER_SERVICE);
                View popupView =
layoutInflater.inflate(R.layout.activity_second,
                        null);
                final PopupWindow popupWindow = new PopupWindow(popupView,
                        ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);

                ImageButton btnDismiss = (ImageButton) popupView
                        .findViewById(R.id.dismiss);
                btnDismiss.setOnClickListener(new Button.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                        // TODO Auto-generated method stub
                        popupWindow.dismiss();
                    }
                });

                popupWindow.showAtLocation(btnOpenPopup, Gravity.CENTER, 0, 0);

            }
        });

        return view;
    }

    @Override
    public void onActivityResult (int requestCode, int resultCode, Intent data)
{
        super.onActivityResult(requestCode, resultCode, data);

        switch (requestCode) {
            case 1: {
                if (resultCode == getActivity().RESULT_OK && data != null) {

                    ArrayList<String> text =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

                    voiceCommand = text.get(0);
                    commandTextView.setText(voiceCommand);

                    if(MainActivity.tcpClient != null) {
                        MainActivity.tcpClient.sendMessage("sc" +
voiceCommand);
                    }
                    else {
                        receivedTextView.setText("Not connected");
                    }

                }
                break;
            }
        }
    }
```

```
}
```

6. Tab 2

```
package studio.android.art.artdrone3;

/**
 * Created by hp on 25/11/15.
 */
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

/**
 * Created by hp1 on 21-01-2015.
 */
public class Tab2 extends Fragment {
    public static ImageView cameraDrone;

    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.tab_2,container,false);
        cameraDrone = (ImageView) v.findViewById(R.id.cameraDrone);

        return v;
    }
}
```

7. Tab 3

```
package studio.android.art.artdrone3;

/**
 * Created by hp on 25/11/15.
 */
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

/**
 * Created by hp1 on 21-01-2015.
 */
public class Tab3 extends Fragment {
    public static TextView armMode;
    public static TextView flightMode;
    public static TextView altitude;
    public static TextView compass;
    public static TextView velocityX;
```

```
    public static TextView velocityY;
    public static TextView velocityZ;
    public static TextView airTemperature;
    public static TextView airPressure;
    public static TextView battery;

    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.tab_3,container,false);

        armMode = (TextView) v.findViewById(R.id.armMode);
        flightMode = (TextView) v.findViewById(R.id.flightMode);
        altitude = (TextView) v.findViewById(R.id.altitude);
        compass = (TextView) v.findViewById(R.id.compass);
        velocityX = (TextView) v.findViewById(R.id.velocityX);
        velocityY = (TextView) v.findViewById(R.id.velocityY);
        velocityZ = (TextView) v.findViewById(R.id.velocityZ);
        airTemperature = (TextView) v.findViewById(R.id.airTemperature);
        airPressure = (TextView) v.findViewById(R.id.airPressure);
        battery = (TextView) v.findViewById(R.id.battery);

        return v;
    }

}
```

8. Splash Screen

```
package studio.android.art.artdrone3;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

/**
 * Created by filip on 2/24/2015.
 */
public class Splash extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splashh);

        final ImageView iv = (ImageView) findViewById(R.id.imageView);
        final Animation an = AnimationUtils.loadAnimation(getBaseContext(),
R.anim.rotate);
        final Animation an2 = AnimationUtils.loadAnimation(getBaseContext(),
R.anim.abc_fade_out);

        iv.startAnimation(an);
        an.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {

            }
```

```
            @Override
            public void onAnimationEnd(Animation animation) {
                iv.startAnimation(an2);
                finish();
                Intent i = new Intent(getBaseContext(),MainActivity.class);
                startActivity(i);
            }

            @Override
            public void onAnimationRepeat(Animation animation) {

            }
        });
    }
}
```

9. TCP Client

```
package studio.android.art.artdrone3;

import android.util.Log;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Objects;

/**
 * Description
 *
 * @author Catalin Prata
 *         Date: 2/12/13
 */
public class TCPClient {

    public static String SERVER_IP; //your computer IP address
    public static int SERVER_PORT;
    // message to send to the server
    private String mServerMessage;
    // sends message received notifications
    private OnMessageReceived mMessageListener = null;
    // while this is true, the server will continue running
    private boolean mRun = false;
    // used to send messages
    private PrintWriter mBufferOut;
    // used to read messages from the server
    private BufferedReader mBufferIn;

    /**
     * Constructor of the class. OnMessagedReceived listens for the messages
received from server
     */
    public TCPClient(OnMessageReceived listener, String ipAddress, int
```

```
portAddress) {
        mMessageListener = listener;
        SERVER_IP = ipAddress;
        SERVER_PORT = portAddress;
    }

    /**
     * Sends the message entered by client to the server
     *
     * @param message text entered by client
     */
    public void sendMessage(String message) {
        if (mBufferOut != null && !mBufferOut.checkError()) {
            mBufferOut.println(message);
            mBufferOut.flush();
        }
    }

    /**
     * Close the connection and release the members
     */
    public void stopClient() {
        Log.i("Debug", "stopClient");

        // send mesage that we are closing the connection
        //sendMessage(Constants.CLOSED_CONNECTION + "Kazy");

        mRun = false;

        if (mBufferOut != null) {
            mBufferOut.flush();
            mBufferOut.close();
        }

        mMessageListener = null;
        mBufferIn = null;
        mBufferOut = null;
        mServerMessage = null;
    }

    public void run() {

        mRun = true;

        try {
            //here you must put your computer's IP address.
            InetAddress serverAddr = InetAddress.getByName(SERVER_IP);

            Log.e("TCP Client", "C: Connecting...");

            //create a socket to make the connection with the server
            Socket socket = new Socket(serverAddr, SERVER_PORT);

            try {
                Log.i("Debug", "inside try catch");
                //sends the message to the server
                mBufferOut = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())), true);

                //receives the message which the server sends back
                InputStream mInputStream = socket.getInputStream();
                DataInputStream mDataInputStream = new
```

```
DataInputStream(mInputStream);
                mBufferIn = new BufferedReader(new
InputStreamReader(mInputStream));
                byte[] data;
                int len;
                byte[] isImage = "I".getBytes();
                byte[] isString = "S".getBytes();

                while (mRun) {
                    mServerMessage = mBufferIn.readLine();
                    //Log.e("Debug1", mServerMessage);
                    if (Objects.equals(mServerMessage, "*AI")) {
                        mServerMessage = "";
                        mServerMessage = mBufferIn.readLine();
                        len = Integer.parseInt(mServerMessage);
                        if (len > 0) {
                            data = new byte[len];

                            //Log.e("Debug2", len + "");
                            mDataInputStream.readFully(data, 0, len);

                            if (mMessageListener != null) {
                                //call the method messageReceived from
MyActivity class

                                mMessageListener.messageReceived(data,
isImage);

                            }
                        }
                    }
                    else if (mServerMessage != null && mMessageListener !=
null) {
                        //call the method messageReceived from MyActivity class

mMessageListener.messageReceived(mServerMessage.getBytes(), isString);
                    }

                }
                //Log.e("RESPONSE FROM SERVER", "S: Received Message: '" +
mServerMessage + "'");

            } catch (Exception e) {
                Log.e("TCP", "S: Error", e);
            } finally {
                //the socket must be closed. It is not possible to reconnect to
this socket
                // after it is closed, which means a new socket instance has to
be created.
                socket.close();
            }

        } catch (Exception e) {

            Log.e("TCP", "C: Error", e);

        }

    }

    //Declare the interface. The method messageReceived(String message) will
must be implemented in the MyActivity
    //class at on asynckTask doInBackground
    public interface OnMessageReceived {
```

```
        public void messageReceived(byte[] message, byte[] type);
    }
}
```

10. Tab 1,2, & 3 xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#FAFAFA"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center">
        <ImageButton
            android:id="@+id/speakButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/mic"
            android:background="@android:color/transparent" />
    </LinearLayout>

    <Space
        android:layout_width="match_parent"
        android:layout_height="17dp"
        android:layout_gravity="right" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:id="@+id/commandTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight=".20"
            android:ems="10"
            android:gravity="center"
            android:layout_gravity="center_vertical">
        </TextView>
        <TextView
            android:id="@+id/receivedTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight=".20"
            android:ems="10"
            android:gravity="center|center_horizontal"
            android:layout_gravity="center_vertical">
        </TextView>
    </LinearLayout>
```

```
<Space
    android:layout_width="match_parent"
    android:layout_height="16dp" />
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ipAddressLabel"
        android:text="IP Address "
        android:textStyle="italic"
        android:layout_weight=".50" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/portAddressLabel"
        android:layout_weight=".50"
        android:text="Port Address"
        android:textStyle="italic" />
</LinearLayout>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/ipAddressEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight=".50"
        android:text="10.5.76.245" >
    </EditText>

    <EditText
        android:id="@+id/portAddressEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight=".50"
        android:inputType="number"
        android:text="50005" >
    </EditText>
</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="17dp" />

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/languageLabel"
        android:text="Language"
        android:textStyle="italic"
        android:layout_weight=".50" />

    <TextView
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/commandLabel"
        android:text="Manual Command"
        android:textStyle="italic"
        android:layout_weight=".50" />
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/languageEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight=".40"
        android:text="id-ID" >
    </EditText>
    <EditText
        android:id="@+id/commandEdit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="type command"
        android:layout_gravity="bottom"
        android:layout_weight=".50">
    </EditText>
    <ImageButton
        android:id="@+id/sendCommandButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/sent"
        android:background="@android:color/transparent"
        android:layout_gravity="right" />
</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="15dp" />

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right" >



</LinearLayout>

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText4"
    android:text="Connect or Disconnect Drone"
    android:textStyle="italic" />

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:weightSum="1">

    <Button
        android:id="@+id/connectButton"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Connect" />

        <Button
            android:id="@+id/stopButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Stop" />
    </LinearLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <ImageButton
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:src="@drawable/info"
            android:background="@android:color/transparent"
            android:id="@+id/imageButton"
            android:layout_gravity="end|bottom"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentEnd="true" />
    </RelativeLayout>

</LinearLayout>


<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:background="#FAFAFA"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/cameraDrone"
        android:layout_below="@+id/videoLabel"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/videoLabel"
        android:layout_weight="0.18"
        android:text="Video Feed Drone"
        android:textStyle="bold|italic"
        android:gravity="center_horizontal"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="31dp"
        android:textSize="20sp" />
</RelativeLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:background="#FAFAFA"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Drone Status"
        android:id="@+id/textView"
        android:layout_marginTop="29dp"
        android:textStyle="bold|italic"
        android:textSize="20sp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Disarm"
        android:id="@+id/armMode"
        android:textIsSelectable="true"
        android:layout_below="@+id/textView2"
        android:layout_alignLeft="@+id/textView2"
        android:layout_alignStart="@+id/textView2"
        android:layout_alignRight="@+id/textView7"
        android:layout_alignEnd="@+id/textView7"
        android:gravity="center_horizontal" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Land"
        android:id="@+id/flightMode"
        android:textIsSelectable="true"
        android:layout_below="@+id/textView2"
        android:layout_toRightOf="@+id/textView"
        android:gravity="center_horizontal"
        android:layout_alignRight="@+id/textView5"
        android:layout_alignEnd="@+id/textView5" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/altitude"
        android:textIsSelectable="true"
        android:layout_marginRight="15dp"
        android:layout_marginEnd="15dp"
        android:layout_below="@+id/textView4"
        android:layout_alignRight="@+id/textView7"
        android:layout_alignEnd="@+id/textView7" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/compass"
        android:textIsSelectable="true"
        android:layout_alignTop="@+id/altitude"
```

```
        android:layout_alignLeft="@+id/textView5"
        android:layout_alignStart="@+id/textView5"
        android:layout_marginLeft="24dp"
        android:layout_marginStart="24dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/velocityX"
        android:textIsSelectable="true"
        android:layout_below="@+id/textView7"
        android:layout_toLeftOf="@+id/altitude"
        android:layout_toStartOf="@+id/altitude" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/velocityY"
        android:textIsSelectable="true"
        android:layout_alignTop="@+id/velocityX"
        android:layout_alignLeft="@+id/compass"
        android:layout_alignStart="@+id/compass" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/velocityZ"
        android:textIsSelectable="true"
        android:layout_below="@+id/textView9"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/airTemperature"
        android:textIsSelectable="true"
        android:layout_below="@+id/textView10"
        android:layout_toRightOf="@+id/velocityX"
        android:layout_toEndOf="@+id/velocityX"
        android:layout_marginTop="12dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/airPressure"
        android:textIsSelectable="true"
        android:layout_above="@+id/battery"
        android:layout_alignLeft="@+id/compass"
        android:layout_alignStart="@+id/compass" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/battery"
        android:textIsSelectable="true"
        android:layout_below="@+id/airTemperature"
```

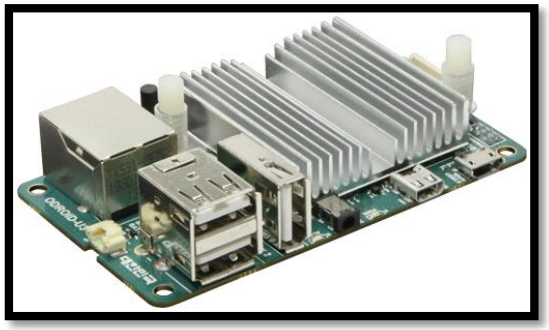
```
        android:layout_toRightOf="@+id/velocityZ"
        android:layout_toEndOf="@+id/velocityZ"
        android:layout_marginTop="52dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Arm Mode"
        android:id="@+id/textView2"
        android:layout_marginTop="28dp"
        android:textStyle="bold|italic"
        android:layout_below="@+id/textView"
        android:layout_toLeftOf="@+id/textView"
        android:layout_toStartOf="@+id/textView" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Flight Mode"
        android:id="@+id/textView3"
        android:textStyle="bold|italic"
        android:layout_alignTop="@+id/textView2"
        android:layout_alignLeft="@+id/flightMode"
        android:layout_alignStart="@+id/flightMode" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Altitude"
        android:id="@+id/textView4"
        android:textStyle="bold|italic"
        android:layout_centerVertical="true"
        android:layout_alignLeft="@+id/textView2"
        android:layout_alignStart="@+id/textView2"
        android:layout_toLeftOf="@+id/textView6"
        android:layout_toStartOf="@+id/textView6"
        android:gravity="center_horizontal" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Compass"
        android:id="@+id/textView5"
        android:textStyle="bold|italic"
        android:layout_alignTop="@+id/textView4"
        android:layout_toRightOf="@+id/textView"
        android:layout_toEndOf="@+id/textView" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Velocity"
        android:id="@+id/textView6"
        android:textStyle="bold|italic"
        android:textSize="20sp"
        android:layout_below="@+id/flightMode"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="18dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="Velocity X"
        android:id="@+id/textView7"
        android:textStyle="bold|italic"
        android:layout_below="@+id/flightMode"
        android:layout_alignLeft="@+id/textView2"
        android:layout_alignStart="@+id/textView2"
        android:layout_marginTop="68dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Velocity Y"
        android:id="@+id/textView8"
        android:textStyle="bold|italic"
        android:layout_alignTop="@+id/textView7"
        android:layout_alignRight="@+id/flightMode"
        android:layout_alignEnd="@+id/flightMode" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Velocity Z"
        android:id="@+id/textView9"
        android:textStyle="bold|italic"
        android:layout_below="@+id/velocityY"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Temperature"
        android:id="@+id/textView10"
        android:textStyle="bold|italic"
        android:layout_marginTop="35dp"
        android:layout_below="@+id/altitude"
        android:layout_alignLeft="@+id/textView4"
        android:layout_alignStart="@+id/textView4" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pressure"
        android:id="@+id/textView11"
        android:textStyle="bold|italic"
        android:layout_alignTop="@+id/textView10"
        android:layout_alignRight="@+id/textView5"
        android:layout_alignEnd="@+id/textView5" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Battery"
        android:id="@+id/textView12"
        android:textStyle="bold|italic"
        android:layout_below="@+id/airPressure"
        android:layout_alignRight="@+id/textView9"
        android:layout_alignEnd="@+id/textView9"
        android:layout_marginTop="25dp" />

</RelativeLayout>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:background="#F9EDDF"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".Splash">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/buton"
        android:id="@+id/imageView"
        android:layout_alignBottom="@+id/imageView2"
        android:layout_centerHorizontal="true" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/art"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:id="@+id/imageView2" />

</RelativeLayout>
```

1. Hardware Specification

This section is to describe hardware description we use to make the ARTDrone system. What we need is Odroid U3 in drone and a minimum requirement smartphone:



Components	Detail
CPU	1.7 Ghz QuadCore
Memory	2GB LPDDR2
DC Input	5V/ 2A
USB Host	3 Ports
Storage	uSD and eMMC

Components	Detail
CPU	1.2 GHz
Memory	1GB RAM
OS	Android Kitkat
Connection	3G Data
Storage	8GB

2. Robot Operating System

ROS is an open-source, meta-operating system for robots. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. The primary goal of ROS is to support code reuse in robotics research and development. ROS is a distributed framework of processes (aka Nodes) that enables executables to be individually designed and loosely coupled at runtime. These processes can be grouped into Packages, which can be easily shared and distributed. ROS also supports a federated system of code Repositories that enable collaboration to be distributed as well. This design, from the filesystem level to the community level, enables independent decisions about development and implementation, but all can be brought together with ROS infrastructure tools.
In support of this primary goal of sharing and collaboration, there are several other goals of the ROS framework:

- **Thin:** ROS is designed to be as thin as possible -- we won't wrap your main() -- so that code written for ROS can be used with other robot software frameworks.
- **ROS-agnostic libraries**: the preferred development model is to write ROS-agnostic libraries with clean functional interfaces.
- **Language independence**: the ROS framework is easy to implement in any modern programming language. We have already implemented it in Python, C++, and Lisp, and we have experimental libraries in Java and Lua.
- **Easy testing**: ROS has a built-in unit/integration test framework called rostest that makes it easy to bring up and tear down test fixtures.
- **Scaling**: ROS is appropriate for large runtime systems and for large development processes

Drone Source Code

On the drone, there attached a single board computer which run the Robot Operating System under Ubuntu 14.04. In the computer, runs five main program that controls the drone flight controller unit, which control the drone movement. These five nodes are using a wifi adapter to communicate to the android application. These five nodes are :

1. **TCP Bridge**

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string>


std::string incoming_reply;
int incoming_reply_size;
void incomingReply(const std_msgs::String& data_reply);

int main(int argc, char **argv){


	std_msgs::String voice_cmd;
	const char * message_reply;

	ros::init(argc, argv, "tcp_bridge");
	ros::NodeHandle n;
	ros::Publisher pub_voice = n.advertise<std_msgs::String>("art_vrd/voice_data", 1024);
	ros::Subscriber sub_incoming_reply = n.subscribe("art_vrd/incoming_reply", 1024,
incomingReply);
	ROS_INFO("Starting TCP Bridge.");
	// ################## Intializing Socket #################
   int socket_desc , client_sock , c , read_size;
   struct sockaddr_in server , client;
   char client_message[512];
   int port_number = 50001;

   socket_desc = socket(AF_INET , SOCK_STREAM , 0);
   if (socket_desc == -1){
		ROS_ERROR_STREAM("[TB] Could not create socket");
   }
   server.sin_family = AF_INET;
   server.sin_addr.s_addr = INADDR_ANY;
   server.sin_port = htons(port_number);
   if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) < 0){
		ROS_ERROR_STREAM("[TB] bind failed. Error: " << strerror(errno));
	  return 1;
   }
   listen(socket_desc , 1);
   c = sizeof(struct sockaddr_in);

	// recv timeout configuration
   struct timeval tv;
	tv.tv_sec = 30;  /* 30 Secs Timeout */
	tv.tv_usec = 0;  // Not init'ing this can cause strange errors
	// ################## Intializing Socket #################
```

```
    while(ros::ok()){
        ROS_INFO_STREAM("[TB] Waiting for incoming connections...");

        client_sock = accept(socket_desc, (struct sockaddr *)&client, (socklen_t*)&c);
        if (client_sock < 0){
            ROS_ERROR_STREAM("[TB] accept failed. Error: " << strerror(errno));
            return 1;
        }
        ROS_INFO_STREAM("[TB] Connection accepted");
        setsockopt(client_sock, SOL_SOCKET, SO_RCVTIMEO, &tv,sizeof(tv));
    while(ros::ok()){
            errno = 0;      // reset errno
            read_size = recv(client_sock , client_message , 512 , 0);
            if (errno != EAGAIN){
                client_message[read_size] = '\0';
                ROS_INFO_STREAM( "[TB] Voice Data : " << client_message);
                voice_cmd.data = client_message;
                pub_voice.publish(voice_cmd);
                incoming_reply = "NULL";

                ROS_INFO_STREAM("[TB] Waiting reply from art_vrd nodes");
                while(read_size != -1 && read_size != 0 && incoming_reply == "NULL" &&
ros::ok()){
                    ros::spinOnce();
                }
                ROS_INFO_STREAM("[TB] Reply accepted from art_vrd nodes");

                message_reply = incoming_reply.c_str();
                incoming_reply_size = incoming_reply.size();

                // ################## Command Reply Header Data
########################
                int32_t byteBuffer = htonl(incoming_reply_size);
                if(send(client_sock, &byteBuffer, sizeof(byteBuffer), 0) < 0) {
                    ROS_ERROR_STREAM("[TB] send failed. Error: " << strerror(errno));
                    break;
                }
                // ################## Command Reply Header Data
########################

                if( send(client_sock , message_reply , incoming_reply_size , 0) < 0){
                    ROS_ERROR_STREAM("[TB] send failed. Error: " << strerror(errno));
                    break;
                }

                if(read_size == 0){
                    ROS_WARN_STREAM("[TB] Client disconnected");
                    break;
                }
                else if(read_size == -1){
                    ROS_ERROR_STREAM("[TB] recv failed. Error: " << strerror(errno));
                    break;
                }
                else if(incoming_reply == "NULL"){
                    ROS_WARN_STREAM("[TB] Client disconnected with NULL message");;
                    break;
                }
            }
```

```
                else if (errno == EAGAIN){
                        ROS_ERROR_STREAM("[TB] recv timed out. Error: " << strerror(errno));
                        break;
                }

                ros::spinOnce();
            }
            close(client_sock);
        sleep(1);
       }
       close(socket_desc);

    return 0;
}

void incomingReply(const std_msgs::String& data_reply)
{
      incoming_reply = data_reply.data;
}
```

2. System Monitor

```
#include "ros/ros.h"
#include "geometry_msgs/Vector3Stamped.h"
#include "sensor_msgs/Temperature.h"
#include "sensor_msgs/FluidPressure.h"
#include "std_msgs/Float64.h"
#include "std_msgs/String.h"
#include "mavros_msgs/State.h"
#include "mavros_msgs/BatteryStatus.h"
#include "mavros_msgs/OverrideRCIn.h"
#include "mavros_msgs/RCIn.h"
#include <string>
#include <string.h>
#include <iostream>
#include <stdlib.h>
#include <unistd.h>

using namespace std;

bool arm_state = 0;
string flight_mode = "AUTO";
float rel_alt = 0;
float compass = 0;
float vel_x = 0;
float vel_y = 0;
float vel_z = 0;
float temperature = 0;
float pressure = 0;
float battery = 0;
int rc_failsafe_override_flag = 0;
int rc_in_data_channel[8];
int channel_7_mid = 1500;
// channel_7_off = 987 | channel_7_on = 2010
```

```
void debugging(string drone_status_debug);
void rcinReceiver(const mavros_msgs::RCIn& rc_in_data);
void altReceiver(const std_msgs::Float64& alt_recv);
void compassReceiver(const std_msgs::Float64& compass_recv);
void velocityReceiver(const geometry_msgs::Vector3Stamped& velocity_recv);
void temperatureReceiver(const sensor_msgs::Temperature& temperature_recv);
void pressureReceiver(const sensor_msgs::FluidPressure& pressure_recv);
void stateReceiver(const mavros_msgs::State& state_recv);
void batteryReceiver(const mavros_msgs::BatteryStatus& battery_recv);
void vDataMonitor(const std_msgs::String& vData);
ros::Publisher pub_rc_override;
ros::Publisher pub_incoming_reply;
mavros_msgs::OverrideRCIn rc_override_data;

int main(int argc, char **argv)
{

      ros::init(argc, argv, "system_monitor");
      ros::NodeHandle sys_mon;
      ros::Subscriber sub_state = sys_mon.subscribe("mavros/state", 100, stateReceiver);
      ros::Subscriber sub_rel_alt = sys_mon.subscribe("/mavros/global_position/rel_alt", 1,
altReceiver );
      ros::Subscriber sub_compass = sys_mon.subscribe("/mavros/global_position/compass_hdg", 10,
compassReceiver );
      ros::Subscriber sub_vel = sys_mon.subscribe("/mavros/global_position/gp_vel", 10,
velocityReceiver );
      ros::Subscriber sub_temperature = sys_mon.subscribe("/mavros/imu/temperature", 10,
temperatureReceiver );
      ros::Subscriber sub_pressure = sys_mon.subscribe("/mavros/imu/atm_pressure", 10,
pressureReceiver );
      ros::Subscriber sub_battery = sys_mon.subscribe("/mavros/battery", 10, batteryReceiver );
      ros::Subscriber rc_in_sub      = sys_mon.subscribe("/mavros/rc/in", 100, rcinReceiver);
      ros::Subscriber sub_vd = sys_mon.subscribe("art_vrd/voice_data", 10, vDataMonitor);
      pub_incoming_reply = sys_mon.advertise<std_msgs::String>("art_vrd/incoming_reply", 100);
      pub_rc_override = sys_mon.advertise<mavros_msgs::OverrideRCIn>("/mavros/rc/override", 1,
true);
      ROS_INFO("Starting System Monitor.");
      // set initial state for rc override take over.
      if(rc_in_data_channel[6] < channel_7_mid){
                  rc_failsafe_override_flag = 0;
      }
      else if (rc_in_data_channel[6] > channel_7_mid){
                  rc_failsafe_override_flag = 1;
      }

      ros::spin();

  return 0;
}


void stateReceiver(const mavros_msgs::State& state_recv){

      flight_mode = state_recv.mode;
      arm_state = state_recv.armed;
}

void rcinReceiver(const mavros_msgs::RCIn& rc_in_data){
```

```
    int x;
    for (x = 0; x<8;x++){
        rc_in_data_channel[x] = rc_in_data.channels[x];
    }
    if(rc_in_data_channel[6] < channel_7_mid && rc_failsafe_override_flag == 0){
        for(int i=0; i < 8; i++) rc_override_data.channels[i] = 0;
        pub_rc_override.publish(rc_override_data);
        rc_failsafe_override_flag = 1;
        ROS_ERROR_STREAM( "[SM] RC is now taking over!") ;
    }
    else if (rc_in_data_channel[6] > channel_7_mid && rc_failsafe_override_flag == 1){
        rc_failsafe_override_flag = 0;
        ROS_ERROR_STREAM( "[SM] Drone is now taking over") ;
    }

}

void altReceiver(const std_msgs::Float64& alt_recv){

    rel_alt = alt_recv.data;
}

void compassReceiver(const std_msgs::Float64& compass_recv){

    compass = compass_recv.data;
}

void velocityReceiver(const geometry_msgs::Vector3Stamped& velocity_recv){

    vel_x = velocity_recv.vector.x;
    vel_y = velocity_recv.vector.y;
    vel_z = velocity_recv.vector.z;

}

void temperatureReceiver(const sensor_msgs::Temperature& temperature_recv){

    temperature = temperature_recv.temperature;

}

void pressureReceiver(const sensor_msgs::FluidPressure& pressure_recv){

    pressure = pressure_recv.fluid_pressure;
}

void batteryReceiver(const mavros_msgs::BatteryStatus& battery_recv){

    battery = battery_recv.voltage;

}
void vDataMonitor(const std_msgs::String& vData){
    std_msgs::String incoming_reply;
    if (vData.data[0] == 's' && vData.data[1] == 'c'){
        ROS_INFO_STREAM( "[SM] It's a sc command") ;
    }

    else if(vData.data[0] == 'd' && vData.data[1] == 's'){
```

```
            ROS_INFO_STREAM( "[SM] It's a ds command") ;
      }

      else if (vData.data[0] == 'v' && vData.data[1] == 'f'){
            ROS_INFO_STREAM( "[SM] It's a vf command") ;
      }

      else{

            incoming_reply.data = "gi:error_unkown_command";
            pub_incoming_reply.publish(incoming_reply);
            ROS_ERROR_STREAM( "[SM] It's not a ds, sc, or vf command");
      }
}

void debugging(string drone_status_debug){

            //Debugging Purpose
            cout << "\n\nDrone Status String" << endl;
            cout << drone_status_debug << endl;
            cout << endl;

            cout << "== Drone Status ==\n" << endl;

            cout << "Arm Mode\t= " << arm_state << endl;
            cout << "Flight Mode\t= " << flight_mode << endl;
            cout << "Altitude\t= " << rel_alt << endl;
            cout << "Compass\t\t= " << compass << endl;
            cout << "Velocity X\t= " << vel_x << endl;
            cout << "Velocity Y\t= " << vel_y << endl;
            cout << "Velocity Z\t= " << vel_z << endl;
            cout << "Temperature\t= " << temperature << endl;
            cout << "Pressure\t= " << pressure << endl;
            cout << "Battery\t\t= " << battery << endl;

}
```

3. Movement Controller

```
#include "../include/art_vrd/movement_controller.h"

#define YES 1
#define NO 0
#define RC NO

int main(int argc, char **argv)
{
      ros::init(argc, argv, "movement_controller");
      ros::NodeHandle n;
      ros::Subscriber vd_sub         = n.subscribe("art_vrd/voice_data", 10, mainMovementController);
      ros::Subscriber rc_in_sub      = n.subscribe("/mavros/rc/in", 100, rcinReceiver);
      ros::Subscriber sub_rel_alt = n.subscribe("/mavros/global_position/rel_alt", 1, altReceiver );
      ros::Subscriber sub_pos        = n.subscribe("/mavros/local_position/local", 100,
positionReceiver );
      ros::Subscriber sub_state      = n.subscribe("/mavros/state", 100, stateReceiver);
```

```
    client                              =
n.serviceClient<mavros_msgs::SetMode>("/mavros/set_mode");
    pub_rc_override                     =
n.advertise<mavros_msgs::OverrideRCIn>("/mavros/rc/override", 1, true);
    pub_quad_pos                        =
n.advertise<geometry_msgs::PoseStamped>("mavros/setpoint_position/local", 1000);
    pub_incoming_reply                   = n.advertise<std_msgs::String>("art_vrd/incoming_reply",
100);
  ROS_INFO_STREAM("Starting Movement Controller.");

  known_command[0] = "naik";
    known_command[1] = "turun";
    known_command[2] = "maju";
    known_command[3] = "mundur";
    known_command[4] = "kiri";
    known_command[5] = "kanan";
    known_command[6] = "pulang ke pangkalan";
    known_command[7] = "mendarat di sini";
    known_command[8] = "lepas landas";

    #if RC == NO

    // ############# comment this if you have a rc #############
    usleep(2000000);

    rc_override_data.channels[6] = 1000;
    pub_rc_override.publish(rc_override_data);

    usleep(2000000);

    rc_override_data.channels[6] = 2000;
    pub_rc_override.publish(rc_override_data);

    // ############# comment this if you have a rc #############

    #endif


    ros::spin();
    return 0;
}


void mainMovementController(const std_msgs::String& vData)
{
    int x;
    int distance_vdata = 0;

    if ( vData.data[0] == 's' && vData.data[1] == 'c' ){

        for(x=0; x < command_list;x++){
            command_search = strstr(vData.data.c_str(), known_command[x]);
            if (command_search != NULL){
                command_detected = x;
                break;                              // exit loop if found a known_command
            }

            else{
```

```
                command_detected = -1;          // set to invalid if known_command not found
            }
        }

        if(rc_in_channel_7 > channel_7_mid){
            if(command_detected == 0 && flight_mode == "GUIDED" && arm_state){

                // naik command
                distance_vdata = strtol
(command_search+strlen(known_command[command_detected]),NULL,10);
                moveDrone('z',pos_z+distance_vdata);
            }

            else if(command_detected == 1 && flight_mode == "GUIDED" && arm_state){

                // turun command
                distance_vdata = strtol
(command_search+strlen(known_command[command_detected]),NULL,10);
                moveDrone('z',pos_z-distance_vdata);

            }

            else if(command_detected == 2 && flight_mode == "GUIDED" && arm_state){

                // maju command
                distance_vdata = strtol
(command_search+strlen(known_command[command_detected]),NULL,10);
                moveDrone('y',pos_y+distance_vdata);
            }

            else if(command_detected == 3 && flight_mode == "GUIDED" && arm_state){

                // mundur command
                distance_vdata = strtol
(command_search+strlen(known_command[command_detected]),NULL,10);
                moveDrone('y',pos_y-distance_vdata);

            }

            else if(command_detected == 4 && flight_mode == "GUIDED" && arm_state){

                // kiri command
                distance_vdata = strtol
(command_search+strlen(known_command[command_detected]),NULL,10);
                moveDrone('x',pos_x-distance_vdata);

            }

            else if(command_detected == 5 && flight_mode == "GUIDED" && arm_state){

                // kanan command
                distance_vdata = strtol
(command_search+strlen(known_command[command_detected]),NULL,10);
                moveDrone('x',pos_x+distance_vdata);

            }

            else if(command_detected == 6 ){
```

```
        if (flight_mode == "RTL"){
                command_detected = -2;          // send error reply to android
        }
        else {
                if(!changeFlightMode("RTL")){
                        command_detected = -2;          // send error reply to android
                }
        }
}

else if(command_detected == 7 ){

        if (flight_mode == "LAND"){
                command_detected = -2;          // send error reply to android
        }
        else {
                if(!changeFlightMode("LAND")){
                        command_detected = -2;          // send error reply to android
                }
        }
}

else if(command_detected == 8 && !arm_state){

        changeFlightMode("STABILIZE");
        usleep(3000000);

        system("rosrun mavros mavsafety arm");
        ROS_INFO_STREAM( "Wait 5 seconds" ) ;
        usleep(5000000);

        for(int i=0; i < 8; i++) rc_override_data.channels[i] = 65535;
        rc_override_data.channels[2] = 1185;
        pub_rc_override.publish(rc_override_data);
        usleep(5000000);
        changeFlightMode("AUTO");
        usleep(13000000);
        changeFlightMode("GUIDED");

        for(int i=0; i < 8; i++) rc_override_data.channels[i] = 65535;
        rc_override_data.channels[2] = 0;
        pub_rc_override.publish(rc_override_data);


}

else {
        command_detected = -1;
}

if(command_detected == -3 ){
        sendReply("sc:drone_movement_is_zero\n");
}

else if(command_detected == -2 ){
        sendReply("sc:can't_change_fm\n");
}
```

```
                else if(command_detected != -1 ){
                        sendReply("sc:ok\n");
                }

                else{
                        sendReply("sc:invalid_or_error\n");
                }
        }

        else{
                sendReply("sc:rc_takeover\n");
                ROS_ERROR_STREAM( "[MC] RC Take Over!" ) ;
        }
    }
}

void rcinReceiver(const mavros_msgs::RCIn& rc_in_data){

    rc_in_channel_7 = rc_in_data.channels[6];
}

void altReceiver(const std_msgs::Float64& alt_msg){

    rel_alt = alt_msg.data;
    pos_z   = rel_alt;
}

void positionReceiver(const geometry_msgs::PoseStamped& local_recv){

    pos_x = local_recv.pose.position.x;
    pos_y = local_recv.pose.position.y;
}

void stateReceiver(const mavros_msgs::State& state_recv){
    arm_state = state_recv.armed;
    flight_mode = state_recv.mode;
}

void moveDrone(char axis, int location){
    int movement_distance;
    if (axis == 'x'){
                movement_distance = abs(pos_x-location);
        }

    else if (axis == 'y'){
                movement_distance = abs(pos_y-location);
        }

    else if (axis == 'z'){
                movement_distance = abs(pos_z-location);
        }

    else {
                movement_distance = DISTANCELIMIT+1;                // always never sent to topic if
axis input invalid
        }
```

```
        if (movement_distance < DISTANCELIMIT && movement_distance != 0){
                quad_pos.header.stamp = ros::Time::now();
                quad_pos.header.frame_id = "1";

                if (axis == 'x'){
                        quad_pos.pose.position.x = location;
                        quad_pos.pose.position.y = pos_y;
                        quad_pos.pose.position.z = pos_z;
                }

                else if (axis == 'y'){
                        quad_pos.pose.position.x = pos_x;
                        quad_pos.pose.position.y = location;
                        quad_pos.pose.position.z = pos_z;
                }

                else if (axis == 'z'){
                        quad_pos.pose.position.x = pos_x;
                        quad_pos.pose.position.y = pos_y;
                        quad_pos.pose.position.z = location;
                }

                else {
                        command_detected = -1;
                }

                pub_quad_pos.publish(quad_pos);
        }

        else if(movement_distance == 0){
                command_detected = -3;
        }

        else {
                command_detected = -1;
        }

}

void sendReply(const char* reply_message){

        incoming_reply.data = reply_message;
        pub_incoming_reply.publish(incoming_reply);
}

int changeFlightMode(const char* mode){
        bool success;
        flight.request.base_mode = 0;                   //Set to 0 to use custom_mode
        flight.request.custom_mode = mode;                  //Set to '' to use base_mode
        success = client.call(flight);

        if(success){
                ROS_INFO_STREAM( "[MC] Flight Mode changed to "<< flight.request.custom_mode ) ;
        }
        else {
                ROS_ERROR_STREAM( "[MC] Failed to change." ) ;
        }
```

```
    return success;
}
```

4. Drone Status

```
#include "ros/ros.h"
#include "geometry_msgs/Vector3Stamped.h"
#include "sensor_msgs/Temperature.h"
#include "sensor_msgs/FluidPressure.h"
#include "std_msgs/Float64.h"
#include "std_msgs/String.h"
#include "mavros_msgs/State.h"
#include "mavros_msgs/BatteryStatus.h"
#include <string>
#include <string.h>
#include <iostream>
#include <stdlib.h>
#include <unistd.h>

using namespace std;

bool arm_state = 0;
string flight_mode = "AUTO";
float rel_alt = 0;
float compass = 0;
float vel_x = 0;
float vel_y = 0;
float vel_z = 0;
float temperature = 0;
float pressure = 0;
float battery = 0;


void debugging(string drone_status_debug);
void altReceiver(const std_msgs::Float64& alt_recv);
void compassReceiver(const std_msgs::Float64& compass_recv);
void velocityReceiver(const geometry_msgs::Vector3Stamped& velocity_recv);
void temperatureReceiver(const sensor_msgs::Temperature& temperature_recv);
void pressureReceiver(const sensor_msgs::FluidPressure& pressure_recv);
void stateReceiver(const mavros_msgs::State& state_recv);
void batteryReceiver(const mavros_msgs::BatteryStatus& battery_recv);
void mainStatus(const std_msgs::String& vData);
ros::Publisher pub_incoming_reply;

int main(int argc, char **argv)
{

    ros::init(argc, argv, "drone_status");
    ros::NodeHandle status;
    ros::Subscriber sub_state          = status.subscribe("mavros/state", 100, stateReceiver);
    ros::Subscriber sub_rel_alt    = status.subscribe("/mavros/global_position/rel_alt", 1,
altReceiver );
    ros::Subscriber sub_compass          =
status.subscribe("/mavros/global_position/compass_hdg", 10, compassReceiver );
    ros::Subscriber sub_vel              = status.subscribe("/mavros/global_position/gp_vel", 10,
velocityReceiver );
    ros::Subscriber sub_temperature = status.subscribe("/mavros/imu/temperature", 10,
temperatureReceiver );
```

```
    ros::Subscriber sub_pressure    = status.subscribe("/mavros/imu/atm_pressure", 10,
pressureReceiver );
    ros::Subscriber sub_battery     = status.subscribe("/mavros/battery", 10, batteryReceiver );
    ros::Subscriber sub_vd              = status.subscribe("art_vrd/voice_data", 10, mainStatus);
    pub_incoming_reply                      =
status.advertise<std_msgs::String>("art_vrd/incoming_reply", 100);
    ros::spin();

  return 0;
}

void stateReceiver(const mavros_msgs::State& state_recv){

    flight_mode = state_recv.mode;
    arm_state = state_recv.armed;
}

void altReceiver(const std_msgs::Float64& alt_recv){

    rel_alt = alt_recv.data;
}

void compassReceiver(const std_msgs::Float64& compass_recv){

    compass = compass_recv.data;
}

void velocityReceiver(const geometry_msgs::Vector3Stamped& velocity_recv){

    vel_x = velocity_recv.vector.x;
    vel_y = velocity_recv.vector.y;
    vel_z = velocity_recv.vector.z;

}

void temperatureReceiver(const sensor_msgs::Temperature& temperature_recv){

    temperature = temperature_recv.temperature;

}

void pressureReceiver(const sensor_msgs::FluidPressure& pressure_recv){

    pressure = pressure_recv.fluid_pressure;
}

void batteryReceiver(const mavros_msgs::BatteryStatus& battery_recv){

    battery = battery_recv.voltage;

}


void mainStatus(const std_msgs::String& vData){
    string s_arm_state      = static_cast<ostringstream*>( &(ostringstream() << arm_state) )-
>str();
    string s_rel_alt        = static_cast<ostringstream*>( &(ostringstream() << rel_alt) )->str();
```

```
    string s_compass       = static_cast<ostringstream*>( &(ostringstream() << compass) )->str();
    string s_vel_x         = static_cast<ostringstream*>( &(ostringstream() << vel_x) )->str();
    string s_vel_y         = static_cast<ostringstream*>( &(ostringstream() << vel_y) )->str();
    string s_vel_z         = static_cast<ostringstream*>( &(ostringstream() << vel_z) )->str();
    string s_temperature = static_cast<ostringstream*>( &(ostringstream() << temperature) )->str();
    string s_pressure      = static_cast<ostringstream*>( &(ostringstream() << pressure) )->str();
    string s_battery       = static_cast<ostringstream*>( &(ostringstream() << battery) )->str();

    std_msgs::String incoming_reply;
    if ( vData.data[0] == 'd' && vData.data[1] == 's' ){

        incoming_reply.data = "ds:"+s_arm_state+";"+flight_mode+";"+s_rel_alt+";"+s_compass+";"+s_vel_x+";"+s_vel_y+";"+s_vel_z+";"+s_temperature+";"+s_pressure+";"+s_battery+";\n";
        //debugging(incoming_reply.data);
        pub_incoming_reply.publish(incoming_reply);
    }
}

void debugging(string drone_status_debug){

        //Debugging Purpose
        cout << "\n\nDrone Status String" << endl;
        cout << drone_status_debug << endl;
        cout << endl;
        cout << "== Drone Status ==\n" << endl;
        cout << "Arm Mode\t= " << arm_state << endl;
        cout << "Flight Mode\t= " << flight_mode << endl;
        cout << "Altitude\t= " << rel_alt << endl;
        cout << "Compass\t\t= " << compass << endl;
        cout << "Velocity X\t= " << vel_x << endl;
        cout << "Velocity Y\t= " << vel_y << endl;
        cout << "Velocity Z\t= " << vel_z << endl;
        cout << "Temperature\t= " << temperature << endl;
        cout << "Pressure\t= " << pressure << endl;
        cout << "Battery\t\t= " << battery << endl;

}
```

5. **Camera Feed**

```
#include <iostream>
#include <string>
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <unistd.h>
#include <stdlib.h>
#include <iostream>
#include <vector>
#include <stdio.h>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
```

```
using namespace std;
using namespace cv;
void mainCameraFeed(const std_msgs::String& vData);
ros::Publisher pub_incoming_reply;
// ############ OpenCV Variable ############
char video_data[100000];
vector<uchar> buff;
vector<int> param = vector<int>(2);
Mat image_capture;
VideoCapture cap(0);                    // Membuka Kamera
// ############ OpenCV Variable ############

int main(int argc, char **argv){

      ros::init(argc, argv, "camera_feed");
      ros::NodeHandle n;
      ros::Subscriber sub_vd = n.subscribe("art_vrd/voice_data", 10, mainCameraFeed);
      pub_incoming_reply = n.advertise<std_msgs::String>("art_vrd/incoming_reply", 100);
      ROS_INFO("Starting Camera Feed.");

      // ############ Video Init ############
   cap.set(CV_CAP_PROP_FRAME_WIDTH, 640);          // Set Lebar gambar
      cap.set(CV_CAP_PROP_FRAME_HEIGHT, 360);      // Set tinggi Gambar
   param[0] = CV_IMWRITE_JPEG_QUALITY;                  // set tipe encoding
      param[1] = 30;                                    // set kualitas encoding

      if(!cap.isOpened()){
            ROS_ERROR_STREAM("[CF] error opening camera") ;
            return -1;
      }
      // ############ Video Init ############

      ros::spin();
      return 0;
}

void mainCameraFeed(const std_msgs::String& vData){
      std_msgs::String incoming_reply;
      int x;

      if ( vData.data[0] == 'v' && vData.data[1] == 'f' ){

            cap >> image_capture;                                       // mengambil gambar
dari kamera
            cvtColor(image_capture, image_capture, CV_8U);

            imencode(".jpeg",image_capture, buff,param);          // meng-encode gambar

            for (x = 0; x < buff.size(); x++){                          // merubah data gambar
dari tipe vector ke array
                  video_data[x] = buff[x];
            }
            std::string video_string(video_data,buff.size());
            incoming_reply.data = video_string;
            pub_incoming_reply.publish(incoming_reply);
      }

}
```

TESTING AND ANALYSIS

A. Testing Form

ARTDrone

Nama :

Tanggal Pengisian Form :

1. Testing System

Sistem harus dapat memberikan informasi kepada user tentang apapun yang sedang terjadi pada sistem melalui respon dan waktu yang tepat.

No	Testing System	Yes	No	Komentar
1	Apakah main interface memberikan informasi yang sesuai?			
2	Apakah antar activity pada aplikasi berhasil difungsikan?			
3	Apakah tiap tombol menampilkan tampilan yang sesuai?			
4	Apakah fungsi speech recognition berjalan dengan baik?			
5	Apakah fungsi video berfungsi sesuai?			
6	Apakah fungsi status berfungsi dengan baik?			
7	Apakah koneksi antar device dengan drone berjalan dengan baik?			
8	Apakah fungsi notifikasi kesalahan masukkan berjalan dengan baik?			
9	Apakah drone melakukan sesuai dengan input speech recognition?			
10	Apakah aplikasi ini bersifat user-			

	friendly?			

2. Consistency and Standard

Sistem seharusnya menggunakan penamaan yang standard dan konsisten pada keseluruhan sistem, agar tidak menimbulkan ambiguitas ketika digunakan.

No	Consistensy Testing	Yes	No	Komentar
1	Apakah setiap halaman memiliki judul?			
2	Apakah anda mengerti dengan hanya membaca judul disetiap halamannya?			
3	Apakah setiap tombol berfungsi sama memiliki konsistensi yang baik?			

3. User Interface

Desain yang dibuat harus mudah dimengerti dan menarik perhatian pengguna.

No	User Interface	Yes	No	Komentar
1	Apakah tampilan aplikasi menarik?			
2	Apakah tiap tombol pada tiap fitur kontrol mudah dimengerti?			
3	Apakah jenis huruf dan ukurannya nyaman untuk dibaca?			
4	Apakah logo utama merepresentasikan aplikasi?			

B. Testing Result and Analysis

Testing Result

This table explains about the form result we gave before, and the value on the table is from our alpha testing which done to preview our application. We have 10 users who tested our application.

No. Soal	Pertanyaan	Nilai Jawaban User ke-										Total	Rata-Rata
		1	2	3	4	5	6	7	8	9	10		
1.1	Apakah main interface memberikan informasi yang sesuai?	1	1	1	1	1	1	1	-1	-1	-1	4	
1.2	Apakah antar activity pada aplikasi berhasil difungsikan?	1	1	1	1	1	1	1	1	1	1	10	
1.3	Apakah tiap tombol menampilkan tampilan yang sesuai?	1	1	1	1	1	1	-1	1	1	1	8	
1.4	Apakah fungsi speech recognition berjalan dengan baik?	1	1	1	1	1	1	1	1	1	1	10	
1.5	Apakah fungsi video berfungsi sesuai?	1	1	1	1	1	1	1	1	1	1	10	
1.6	Apakah fungsi status berfungsi dengan baik?	1	1	1	1	1	1	1	1	1	1	10	**9**
1.7	Apakah koneksi antar device dengan drone berjalan dengan baik?	1	1	1	1	1	1	1	1	1	1	10	
1.8	Apakah fungsi notifikasi kesalahan masukkan berjalan dengan baik?	1	1	1	1	1	1	1	1	1	1	10	
1.9	Apakah drone melakukan sesuai	1	1	1	1	1	1	1	1	1	1	10	

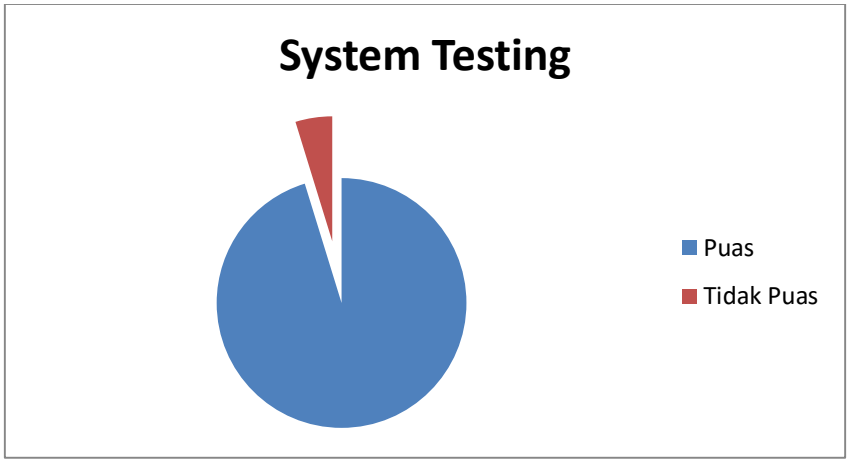
	dengan input speech recognition?												
1.10	Apakah aplikasi ini bersifat user-friendly?	1	1	1	1	1	1	1	1	-1	1	8	
2.1	Apakah setiap halaman memiliki judul?	1	1	1	1	1	1	1	1	1	1	10	
2.2	Apakah anda mengerti dengan hanya membaca judul disetiap halamannya?	1	1	1	1	-1	1	1	-1	1	1	6	**8.6**
2.3	Apakah setiap tombol gambar berfungsi sama memiliki konsistensi yang baik?	1	1	1	1	1	1	1	1	1	1	10	
3.1	Apakah tampilan aplikasi menarik?	1	1	1	1	1	1	1	1	1	1	10	
3.2	Apakah tiap tombol pada tiap fitur kontrol mudah dimengerti?	1	1	1	1	1	1	1	1	1	1	10	**10**
3.3	Apakah jenis huruf dan ukurannya nyaman untuk dibaca?	1	1	1	1	1	1	1	1	1	1	10	
3.4	Apakah logo utama merepresentasikan aplikasi?	1	1	1	1	1	1	1	1	1	1	10	

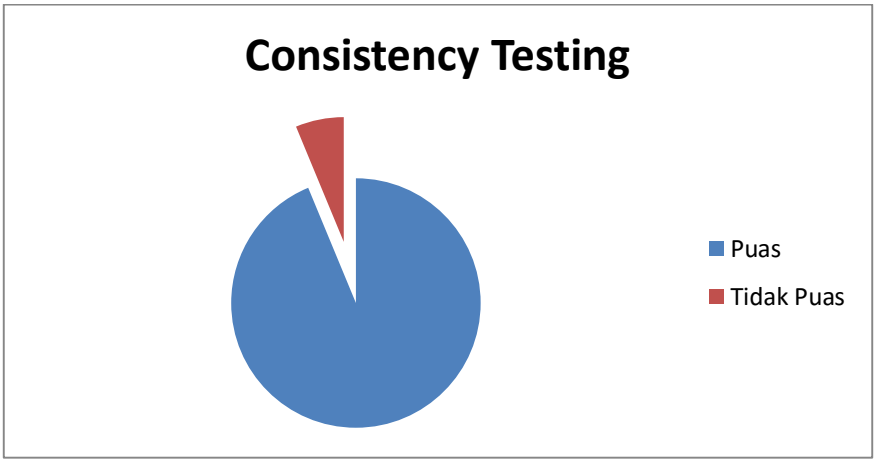
Average = 9.2

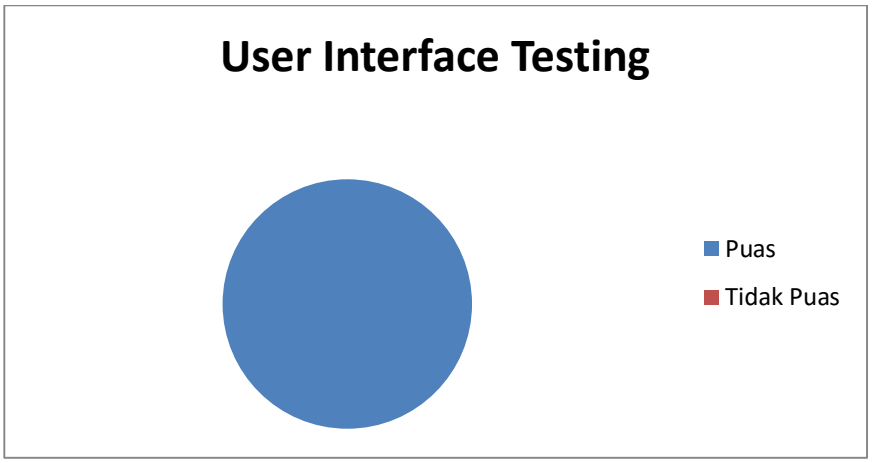
1 = YES -1 = NO

Testing Analysis

Generally, the testing results show that ARTDrone performance is good. From the average point on each testing system, the standard consistency, and user interface we were acknowledged that our application has not perform well in the consistency, since some people still cannot understand some of our activity by its name only. The lowest mark is given by the first question on the first section. This testing were done after some enhancement. Here are some bars to show out users' satisfaction :






User Interface Testing
Puas
Tidak Puas

USER MANUAL

A. Installation

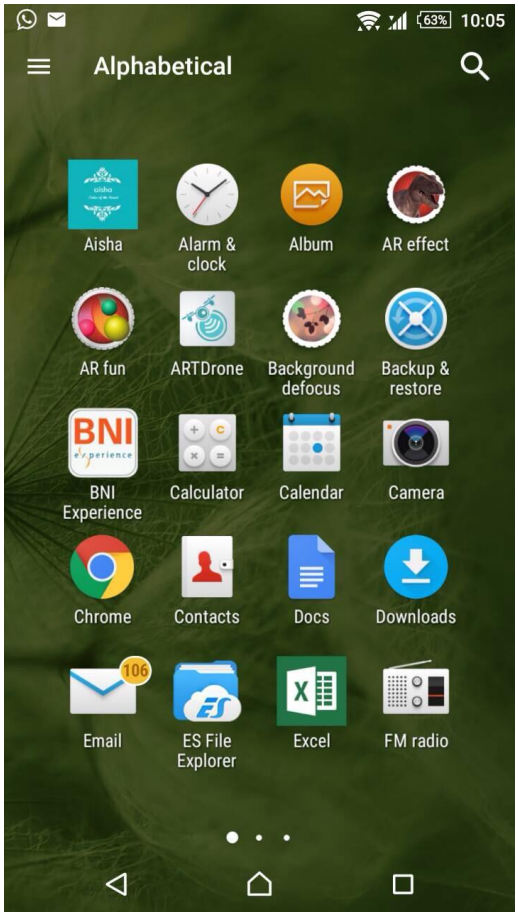
ARTDrone application is an Android-base application and designed to run in full compatibility with gadgets that featured:

- Touch screen support
- OS: Android 3.0 until Android 6.0
- SD card support

To install this application, simply open the installer file and choose option install and when it finish, ARTDrone icon will automatically popped up in gadget's menu.

B. Application Launch

Click on ARTDrone icon which present in your gadget:

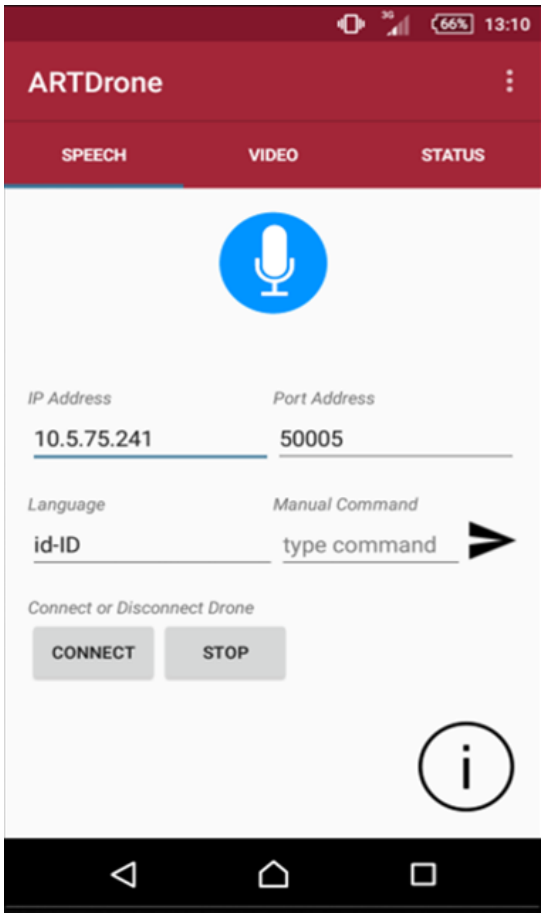


The icon of the application is made simple and look sophisticated to represent a drone communicating over internet:

C. Menu

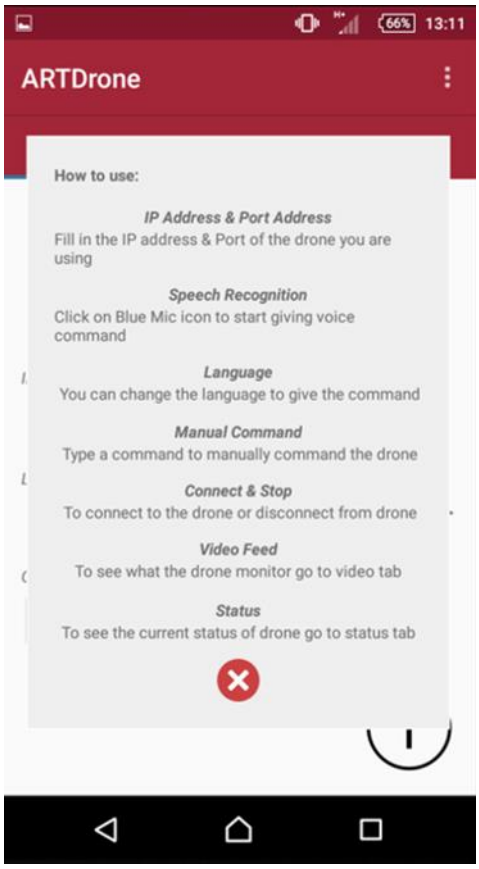
First interface that shows up is the splash and follow by first tab preface.

After the splash screen finish to load its activity, the first activity will commence which is the first layout tab will be present.

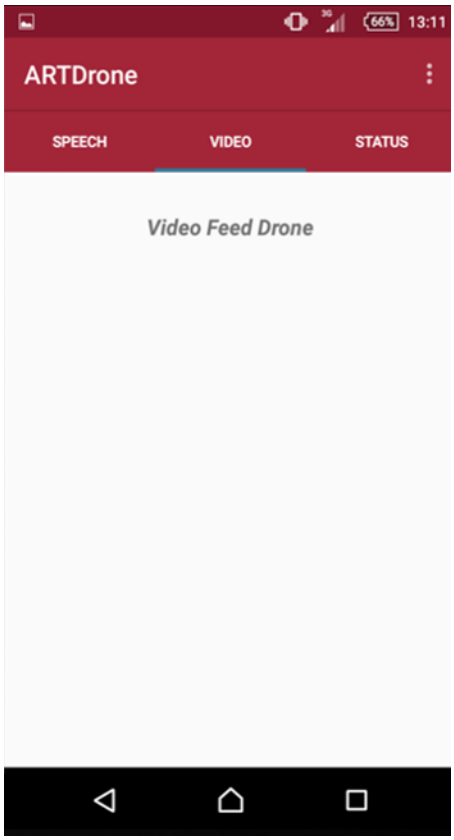


In this interface, you will see there are three tabs. Each tabs represent a certain task for the drone. The first tab is called speech which have the task to give commands through voice. Click the blue microphone button to start speaking by giving the command. The IP address and Port address is used to connect to the correct drone. You can change the language to give command in another language such as english or chinese. To give a better resulting command, user can type a command and manually sent it to the drone by clicking the black airplane icon. The information icon is the user manual on how to operate the application.

The I icon means information, can be click to show a pop up on how to use the ARTDrone android application. The popup message can be close by clicking the cross button.



The next tab is called video. This tab represent a function to let the user see what the drone see with the camera install on the drone. As the user connect to the drone it will automatically run the video. In the image below there are no image being shown from the drone because the application is not connected to the drone. If it is connected the size of the video will be a quarter size of the middle of the tab video. The video run on 20fps to make sure the video data being receive from the drone will be smooth and get a good pixel.



The last tab is called status. This tab lets the user monitor the status of the drone as shown in the picture below. The drone status will change when the application connects to the drone.

The Status tab provides you with several key information of the drone such as Arm mode, Flight mode, Altitude, Compass, Velocity x, y, and z, Temperature, Pressure and Battery. Each status represents the status of its name :

- ✓ Arm mode status show the user if a drone is arm or in a state of on or disarm or in the state of off.
- ✓ Flight mode status show the user the current mode of the drone whether it is in the air or on the land as landing.
- ✓ Altitude status show the height of the drone calculated from the ground.

- ✓ Compass status show you the current direction of the drone
- ✓ Velocity is the same as speed. Velocity represents the speed of the drone. There are three planes that is cover by the drone which is the velocity of x coordinate, velocity of y coordinate, and velocity of z coordinate.
- ✓ Temperature status show the heat of the drone status. This status shows if the drone is overheating or not. It can be use to know the surroundings temperature.
- ✓ Pressure shows the drone pressure status.
- ✓ Battery status show the length of battery life left in the drone power. This indicate how many powers left on the drone.

The action settings in the application provides an activity of the information of the developer of the application and the version of the application.

Below is the list of the commands that the Drone will recognize,

List of Commands:

- ✓ Maju x Meter → Move Forward
- ✓ Mundur x Meter → Move Backward
- ✓ Naik x Meter → Ascending
- ✓ Turun x Meter → Descending
- ✓ Kiri x Meter → Go left
- ✓ Kanan x Meter → Go right
- ✓ Mendarat Disini → Land
- ✓ Pulang ke Pangkalan → Back to starting point
- ✓ Lepas Landas → Take off

PS. x is the number of meter metric

Each command can be added words before or after the command sentences above, for Example:

- ✓ Recognize : ***Saatnya naik 5 meter sekarang***
- ✓ Un-recognize : ***Naik dong 5 meter sekarang***

In this version of application, the drone can only receive and recognize command in Indonesian language. Further improvement in various languages is in development.

REFERENCES

Spiral Model. http://www.princeton.edu/ accessed on 29th September 2013

The Eclipse Foundation open source community website. www.eclipse.org/ accessed periodically

Getting Started as Android Developer. developer.android.com/training/ accessed periodically

Android-related code. http://stackoverflow.com/ accessed periodically

Github collaboration tools. github.com/crcdor/voice-controlled-drone accessed periodically

Smart Home Technologies & Home Automation Control. http://www.savantsystems.com/ accessed on 16th December 2013

Design Electronics. http://www.designelectronics.net/ accessed on 15th November 2013

Huda, Arif Akbarul. *24 Jam pintar pemrograman Android*. 2012. Penerbit Andi : Yogyakarta.