INTERNATIONAL UNIVERSITY LIAISON INDONESIA

Study of Air-Traffic Volume Based on ADS-B Data

Presented to the Faculty of
Engineering

In Partial Fulfilment
Of the Requirements for the Degree
Bachelor of Sciences
In
Aviation Engineering

By
I GEDE Suryadharma Susila

January 19, 2020

"If the facts do not fit the theory, change the facts."

Albert Einstein

INTERNATIONAL UNIVERSITY LIAISON INDONESIA

Abstract

Faculty of Engineering
Department of Aviation Engineering

Bachelor of Engineering

Study of Air-Traffic Volume Based on ADS-B Data

by I GEDE Suryadharma Susila

The purpose of Air-Traffic Density is to identify the traffic flow and visualized the behavior, because air traffic is foreseen to be double by 2030. The objective of this study is to collect the ADS-B messages using the low-cost receiver system and used the ADS-B data to visualize and determine the air-traffic volume. To sustain and further improve safety standard, modern visualizations and analysis tools have to be created. To do so, the approach is going to be using the Automatic Dependent Surveillance-Broadcast or known as ADS-B. ADS-B is an alternative way of supplying position data to aircraft where aircraft continuously transmit information about their identification and position. Such signals can be received with the ADS-B receiver. In this research, the low-cost ADS-B receiver is acquired from the OpenSky Network. The ADS-B message that was collected by using a stream parser is going to be filtered through categorizing the altitude into 6 categories. Each category was named by their flight level, FL1 is categorize for the flight level from 0 to 1000 feet, FL2 starts from 1001 to 2000 feet, FL3 starts from 2001 to 3000 feet, FL4 starts from 3001 to 4000 feet, FL5 starts from 4001 to 5000 feet, and the last group starts from 5001 feet and above. Not only the altitude that was filtered but also the latitude and longitude also grouped by their flight level. By way of illustration, the filtering process was held by using Python as the programming language. In conclusion, the ADS-B data was able to be collected using the low-cost ADS-B receiver system. From the collected data, the air traffic can be visualized and determined from the message that have been gathered using the receiver system as it is. After the air traffic is visualized, the average number of aircraft in Flight Level and the standard deviation can be determined. The highest average and the standard deviation is at FL6 and the lowest average is at FL5.

Keyword: *Automatic Dependent Surveillance-Broadcast (ADS-B); Air-Traffic Volume*

Statement by The Author

I hereby declare that this submission is my own work and to the best of my knowledge, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at any educational institution, except where due acknowledgment is made in the thesis.

I Gede Suryadharma Susila _____ _____

Student Date

Committee Approval

Study of Air-Traffic Volume Based on ADS-B Data

I GEDE Suryadharma Susila

January 19, 2020

Faculty of Engineering

Triwanto Simanjuntak, Ph.D
Advisor, Department of Aviation Engineering _____ Date

Dr. Ir. Prianggada Indra Tanaya, M.M.E.
Dean of Engineering _____ Date

Acknowledgements

Thank you God Almighty for the strength, knowledge, ability and opportunity to undertake and complete this research study. Without his presence, this wonderful achievement would not be possible.

Throughout the writing process of this thesis report I have received a great amount of supports and assistances. I would like to thank my beloved thesis supervisor and advisor, Triwanto Simanjuntak, PhD.

I would like to thank Michael Earley for the help and all of the inputs as my thesis proofreader.

I would also like to thank my parents for the wise counsel, support and sympathetic ear that will always be there for me.

Special mention to Lutfi Muzzaki Khairullah, Yazfan Tabah Tahta Bagaskara and all my family member from aviation engineering batch 2015 for all the great support as well as providing happy distractions to rest my mind throughout the whole writing process.

And finally, last but by no means least, for Fidelis Ardani Emiati as my great extra-support system and for everyone that I have not mentioned yet.

Thank you for all the encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

ATS	**A**ir **T**raffic **S**ervice
ADS-B	**A**utomatic **D**ependent **S**urveillance **B**roadcast
ATC	**A**ir **T**raffic **C**ontrol
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
PSR	**P**rimary **S**urveillance **R**adar
SSR	**S**econdary **S**urveillance **R**adar
ES	**E**xtended **S**quitters
UAT	**U**niversal **A**ccess **T**ransceiver
FL	**F**light **L**evel
ASR	**A**irport **S**urveillance **R**adar
FIS-B	**F**light **I**nformation **S**ervices **B**roadcast
TIS-B	**T**raffic **I**nformation **S**ervices **B**roadcast
NOOBS	**N**ew **O**ut **O**f the **B**ox **S**oftware
CDTI	**C**ockpit **D**isplay of **T**raffic **I**nformation

For all the people I love

Chapter 1

Introduction

1.1 Introduction

The Air Traffic Control's (ATC) goal is to optimize all protection and capacity. Such that, all aircraft are approved without compromising passenger's existence with delays. Traffic density is a valuable information to assess the safety and efficiency of any transport system in terms of traffic flow. On Figure 1.1, air-traffic density is a study about the movement of several aircraft that fly within a given time span in a certain region of airspace [2].

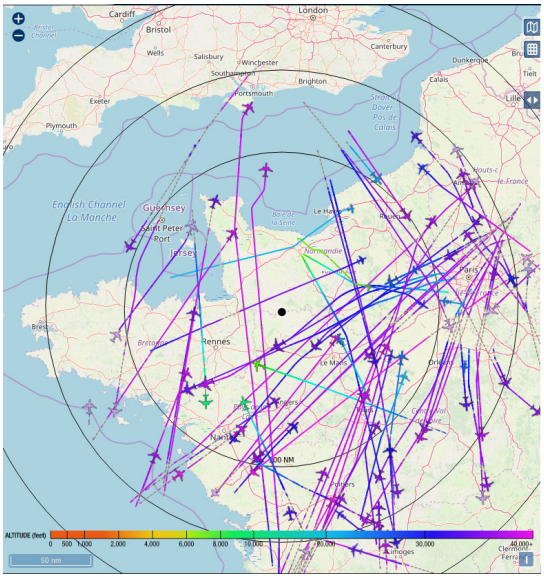

FIGURE 1.1: Air-Traffic Volume In a Certain Region

As it is estimated that air traffic will double by 2030 [1], this will cause a higher density of the traffic which means accident probability will also increase. One of the busiest airport in Indonesia is Bandara Soekarno-Hatta International Airport. There are around 1,200 to 1,300 aircraft takes off or landing in a day at Bandara Soekarno-Hatta International Airport, and also have handled around 63 million of passenger a year [3]. To maintain and further improvement of safety and efficiency, modern visualizations and analysis tools must be developed.

Air-traffic density can be defined in many ways. In this study, air-traffic density can be imagined from the flight trajectories and flight path from all gathered data such as the flight ID, latitude, longitude, and altitude information.

1.2 Background

Air traffic service providers and regulators from all over the world are expanding into airspace and flight operations to allow traffic flow, capacity, security, and protection more versatile and adaptable.

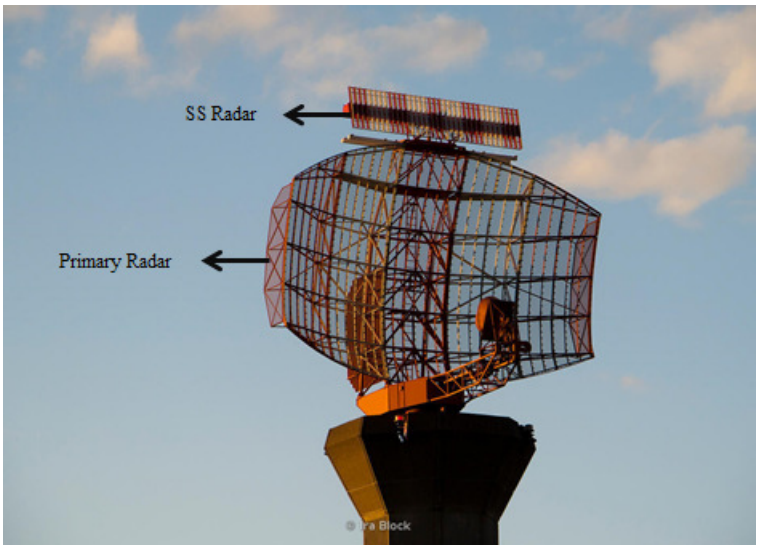

FIGURE 1.2: Airport Surveillance Radar (ASR)

Airport Surveillance Radar (ASR) consist of 2 surveillance radar. Primary Surveillance Radar (PSR) and Secondary Surveillance Radar (SSR). These types of radar have their own weaknesses and their weaknesses are considered poor in providing complete information about the aircraft. For example, PSR does not provide altitude and identity. PSR systems are also very expensive to install and maintain compared to SSR. Although SSR allows communication of identity and altitude, SSR still has possibilities to report false position or report false altitude [4]. SSR still considered expensive to install and maintain. The switch from radar surveillance to Automatic Dependent Surveillance-Broadcast (ADS-B) to monitoring aircraft more effectively and efficiently on the ground and during flight. ADS-B coverage also improved compare with ASR which can only controls traffic within a radius of 60 miles (96 km) to 200 Nm (370.4 km).

ADS-B is a device that enhances safety and effectiveness and helps aircraft, controllers airports, airlines, and people directly. The ADS-B is focused on switching from ground radar and navigation aids to precise tracking of satellite indicators. The system involves an aircraft with ADS-B determining its position using the GPS.

A suitable transmitter then broadcasts that position at rapid intervals, along with identity, altitude, velocity, and other data.

The ADS-B monitoring systems consist of 2 large-scale systems, 1090 Extended Squitter (1090ES), which operate on the 1090 MHz base frequency and the Universal Access Transceiver (UAT), operated on the 978 MHz base frequency. The aircraft Mode-S Transponder transmits both signals. The 1090ES is used for commercially and global applications, while the UAT is a regional system that is specially implemented in the United States for aircraft operating under 18,000 feet. Periodically, at least once every second and without the assistance and interaction of the flight crew or operator, the data set of airplane information is transmitted automatically.

There are two main components of ADS-B: ADS-B Out and ADS-B In. ADS-B Out is a system of monitoring for aircraft detection, as ATC requires to manage traffic. The aircraft's position, speed and altitude are reported once per second by ADS-B Out. ATC and nearby aircraft will receive this transmission and this data represents a radar display equivalent. ADS-B In allows an ADS-B ground station and other aircraft to be transmitted from an aircraft. This is how weather and traffic in the cockpit were collected. Strictly optional to add ADS-B In. An approved ADS-B Out system is required for the ADS-B In operation.

During this discussion of ADS-B, traffic and weather also have their own terms: FIS-B and TIS-B. Via ADS-B In, FIS-B and TIS-B can only be received. Flight Information Services-Broadcast (FIS-B) is a name for Datalink weather. This data is not transmitted on frequency 1090 MHz. Traffic Information Broadcast Service (TIS-B) is a monitoring system designed to increase the visibility of pilots in-flight of local traffic. An aircraft must be fitted with a transponder and must be within radar coverage in order to achieve quality as a TIS-B target.

1.2.1 ADS-B Messages

The main purpose of the ADS-B is to allow all the users to access extremely accurate information about aircraft's location and flight path. The ADS-B will broadcast the data to basically anyone that is listening. Normally, ADS-B messages only have 56 bits of space for data so there are multiples types of messages. The messages can be translated depends on which version of the dump1090 and what kind of stream parser is being used.

Automatic Dependent Surveillance-Broadcast (ADS-B) is a satellite-based surveillance system. ADS-B use a Mode-S Extended Squitter (1090 MHz) to transmits the information about aircraft such as aircraft's position, velocity, and aircraft identification. Most aircraft today continuously broadcast ADS-B messages. ADS-B message

has its own structure.

Usually, the ADS-B message is only collected by ATC for the aircraft information. ADS-B messages are also useful for airlines to track their aircraft and for the historical data of the aircraft. There is a possibility for the ADS-B messages are provided not only from the ATC but also from another person who has an ADS-B receiver of their own and sold the data to the airlines depends on what the airlines need. This can be an opportunity for the ADS-B receiver owner to provide the ADS-B message for those who need it.

ADS-B message can also useful for some researchers. OpenSky-Network is one of the companies that provide ADS-B Data around the world for the Mode S messages. They also provide an offer to be able to become a receiver by buying the OpenSky Receiver Kit from Jetvision. The receiver kit is different from what ATC used for commercials. The OpenSky Receiver Kit is the cheapest receiver that also recommended for receiving the ADS-B data.

1.2.2 ADS-B Receiver

There are many websites that offered a cheap ADS-B receiver kit. OpenSky-Network is also offered a cheap ADS-B receiver kit that includes all the hardware equipment and software that need to set up a receiver. OpenSky-Network offers two alternatives to the receiver kit. The OpenSky Receiver Kit from Jetvision and the Radarcape which is more expensive from the Jetvision version. The Radarcape is equipped with an integrated GPS receiver for precise timestamps as required multilateration applications. Many large networks of air tracking use Radarcape as a standard device.

1.3 Problem Statement

As an effective tool to define aircraft safety and efficiency, air-traffic volume is adopted [5]. In comparison, the volume of air traffic relates to the degree of congestion and provides a time-varying perception of traffic in an airspace system. This research focuses on the movement of several aircraft within a given time span as air traffic increases are significant in a region with the low cost ADS-B receiver and determine the air-traffic volume. For ATC purposes, the concept of air-traffic density or volume is the maximum number of aircraft reaching the field over a certain time period [6]. From all of the information obtained, the visual flight path, the air-traffic volume, and the movement of air traffic from the ADS-B message can be determined and observed.

1.4 Research Purpose

In collecting the data, a maximum result is preferred in order to make a good result. A good data collection can also affect the results of data collection. In this research, the main objectives is to collect the ADS-B messages using the low-cost receiver system. The ADS-B data will be used to visualized and determined the air-traffic volume or density. The data that will be used is the data that comes from the ADS-B receiver as it is.

Air-traffic volume analysis uses complete data as possible. In this research, for calculating the Air-Traffic Analysis it requires data from the altitude, the hex identification, time, and date. A comprehensiveness from the required data is very influencing for calculating the Air-Traffic Analysis. Besides that, there are still many factors that can cause the data is not 100% able to be analyzed. Through some filtering process, the data will be visible and shows the desired results so the Air-Traffic Volume or Density can be visualized and determined.

1.5 Research Scope

The limitations in this study is that the antenna is fixed at Cluster Pasadena, Bukit Dago Housing Estate because of the environment is very supportive. The receiver cannot be monitored and controlled remotely. There are several other candidates to implant the antenna, for example the antenna was not permitted to be installed at the campus building for no reason.

The collecting process is focuses on the movement of aircraft in a region of coverage within a give time (1 month) using the Raspberry-Pi ADS-B receiver system. 1 month is enough to make these two components warm enough to collect the data. The ADS-B data will be used from the ADS-B receiver with the data as it is.

1.6 Research Approach

To achieve the goal of this research, the data is gathered using the ADS-B Receiver kit that is able to receive the signals from aircraft in the radius of up to 200 - 300 km [7] (depending on the environment).

Python is a programming language that is able to read data that has been collected. Some filtering and slicing the data will make the data more visible, by removing some defect data, and made it easier to analyze. In this research, the data that have been gathered is going to be filtered with a range from the altitude or flight level and divided into 6 categories. FL1 indicates the Flight Level or altitude from 0 to 1000 feet, FL2 indicates the Flight Level or altitude from 1001 to 2000 feet, FL3 indicates the Flight Level or altitude from 2001 to 3000 feet, FL4 indicates the Flight Level or altitude from 3001 to 4000 feet, FL5 indicates the Flight Level or altitude from 4001 to 5000 feet, and FL6 indicates the Flight Level or altitude from 5001 feet

and above. Not only the altitude or flight level, the latitude and longitude will also be categorized according to the Flight Level group. After the filtering process, the data can be plotted into graphics. Python has a lot of libraries. In this research, all of the analysis process from filtering to the final result is using python as the programming language. A library such matplotlib is used to plot the graph for the Flight Level categorize with the date as x-axis, and the number of aircraft as the y-axis. Basemap is also a python library that used to plot the map with the latitude and longitude information.

Chapter 2

Literature Review

2.1 Air-Traffic Density

A new approach for maintaining aviation safety and efficiency is studying the amount of aircraft sharing the same airspace. This method is called dynamic density range. Dynamic density or traffic density is an approach to study traffic flow and forecast their behavior in the future. Traffic density is an effective tool to assess the safety and efficiency of the operation of all transport systems. The intensity of air traffic in this report is used to research the movement of multiple aircraft in a particular area of airspace within a given time horizon [2]. In comparison, aircraft density corresponds to the congestion degree and provides a time-varying high-level perspective of traffic in an airspace sector.

2.2 Automatic Dependent Surveillance Broadcast

Automatic Dependent Surveillance-Broadcast (ADS-B) is a surveillance system that relies on aircraft or vehicles that broadcasting their identity, GPS position, and other information such as the Hex ID from the onboard systems, that makes the vehicle can be tracked. For aircraft, the information broadcast will be received by Air Traffic Control (ATC) ground stations. The signal also can be received by any other aircraft who installed ADS-B, so other aircraft with ADS-B equipped can get the situational awareness and enhancing self-separation.

ADS-B has the possibility to replace radar as the primary surveillance for tracking and controlling aircraft. It provides more safety by making an aircraft more visible for the ATC or any other aircraft who equipped ADS-B with flight data, like position and velocity transmitted every second. ADS-B is built in 2 different services. ADS-B In and ADS-B Out. ADS-B In and ADS-B Out are the two different services provided by the ADS-B in general, it can provide traffic and graphical weather information through Traffic Information Service-Broadcast (TIS-B) and Flight Information Service-Broadcast (FIS-B) applications.

FIGURE 2.1: ADS-B Ground Support

- **ADS-B In**
  ADS-B In is the recipient part of the system. This allows ADS-B ground stations and other aircraft to receive the transmission. This is how pilots can get subscription-free weather and traffic in the cockpit. Adding ADS-B In is strictly optional. While it offers some great benefits, the FAA is only concerned about equipping ADS-B out-the free weather and traffic is simply the carrot to get to write the check.

- **ADS-B Out**
  ADS-B Out equipment will continuously transmit aircraft data such as airspeed, altitude, and location to ADS-B ground stations. Those data are then transmitted to air traffic control stations. This transmission is received by ATC and nearby aircraft and this data make up the equivalent of a radar display.

Note that there are combinations of ADS-B Out and ADS-B In. Out-only equipment that simply meets the FAA requirement, while in-only receive weather information, and ADS-B In/Out products do it all.

2.2.1 How ADS-B Technology was Introduced

Regions such as America, Australia, and Fiji create an infrastructure for ADS-B. In the U.S. Gulf of Mexico's oil and gas fields, helicopter traffic is high. ADS-B is used to enhance visibility, traffic information is now used where there is no radar service. Another area where ADS-B is expected to bring real benefits as it is implemented in the congestion of airspace around the east coast of America. For the first time in the Australian Outback ADS-B will enable aircraft to maintain and monitoring capability via direct air-to-air contact with 1090ES. The FAA estimates that the cost to the U.S. economy will be $22 billion in lost economic activity by 2022 without changes

to the U.S. air traffic infrastructure (of which ADS-B technology is one part.). The U.S. Air Surveillance Program, which includes ADS-B is known as 'Next Gen'. This program is expected to have a consumption of 1.4 billion gallons, reduced emissions of 14 million tons and cost savings valued at $23 billion. While these estimates are based on commercial flight operations, there is no doubt that ADS-B will also have a positive effect on GA (General Aviation). [8].

2.2.2 Implementation Of ADS-B

In North America, Europe and other countries, including Asia / Pacific, ADS-B is currently being introduced. The final rule states that aircraft operating in airspace specified in 91.225 must have an ADS-B system with a qualified location origin capable of meeting the requirements set out in 91.227 effective January 1, 2020. These regulations establish a minimum performance standard for both the ADS-B transmitter and the position sources integrated with your aircraft's ADS-B equipment. In North America, Europe, and other fields such as the Asia-Pacific region, ADS-B is currently in progress. EUROCAE and RTCA are creating the criteria of ADS-B together [9].

For aircraft equipment, the ADS-B Out capability onboard is enabled by transponders interfaced with the relevant avionics systems (such as GNSS, pressure altimeters, etc.). Many aircraft already have the ADS-B ES capacity packaged in Mode S Enhanced Surveillance systems for central European airspace. The ADS-B In feature includes a detector and a network of processing system (traffic computer) and HMI unit (often called Cockpit Display of Traffic Information - CDTI). In the Forward field of view, or in the shape of the so-called Electronic Flight Bag (EFB), the ADS-B In system could be integrated. For ADS-B functional use, regulatory authorities need to be accredited and authorized. The relevant certification documents are EASA AMC 20-24 for ADS-B In Non-Radar Airspace or CS-ACNS for "ADS-B Out".

For ground equipment, The ADS-B data are collected at ADS-B stations by aircraft or airport vehicles. In most cases the output of the ADS-B Grounds will be sent to the data processing and distribution systems of surveillance, fused to create a traffic situation image for users by inputs from other possible surveillance sensors [9].

Mode S (1090 MHz) ADS-B systems shall meet with TSO-C166b Technical Standard performance requirements. The aircraft must be fitted with ADS-B Mode S-transponder-based transmitter for the aircraft flying at or above FL180 (18,000 feet). The aircraft need to be fitted with either an Extended Squitter Mode S transponder or Universal Access Transceiver (UAT) compatible with the TSO-C154c performance requirements for a plane flying under 18,000 feet and within US airspace. The UAT hardware offers the ability to receive information from the FAA ADS-B network on

traffic and weather.

The ADS-B rules provisions refer only to airspace as specified in 14 CFR 91.225, whether or not they are working on a VFR or IFR basis. It is an airspace policy which does not extend outside of established airspace to any type of operation. ADS-B Out in the same airspace is mostly required where transponders are necessary [10].

Since late 2006, Indonesia and collaborating partners of Air Service Australia (ASA), SITA and Thales have been installed three ADS-B ground-stations for trials. In Bali, Kupang, and Natuna both ground stations are situated. That was the first phase in ADS-B deployment in Indonesia until the DGCA's functional level. Indonesia has currently installed 30 ADS-B ground and 1 ADS-B ground station in 30 different locations with a dual system, using 1090 Mode S Extended Squitter technology developed to 2014, with a single system for the testbed purpose of each station. All stations are operating independently and all stations are being used at this time to promote ATC daily activities and situational awareness. Nevertheless, in 2017, following the publication of AIRAC AIP Supp Nr. 08/15, DGCA said that ADS-B will be applied to ATS surveillance separation (Air Traffic Service) 245 up to FL 600 from flight level (FL) [11].

2.2.3 How ADS-B Work

ADS-B system in which electronic equipment transmits the exact location of the aircraft automatically through a digital data link onboard an aircraft. ADS-B technology uses GNSS (Global Navigation Satellite System) Constellation to track an aircraft and transmits the aircraft's exact location to the closest ADS-B ground station twice in a second. The aircraft signal is then transmitted by the ground station and sent the data directly to ATC air services through a satellite connection. The data can be used on boards without the need for radar by other aircraft or ATC to indicate the location and height of the aircraft [12].

ADS-B Receiver

In the general receiver system, the 1090 MHz Extended Squitter is the receiver for the ADS-B system. ATC ground stations and aircraft fitted with the Traffic Collision Avoidance System (TCAS) are already equipped with 1090 MHz (Mode S) receivers necessary to be able to transmit the ADS-B signals [13].

In this research, the ADS-B receiver needs to be set up. Have to do some installing and setting because of the receiver is made by some different components. The main component for the receiver system is the Antenna and the RTL-SDR.

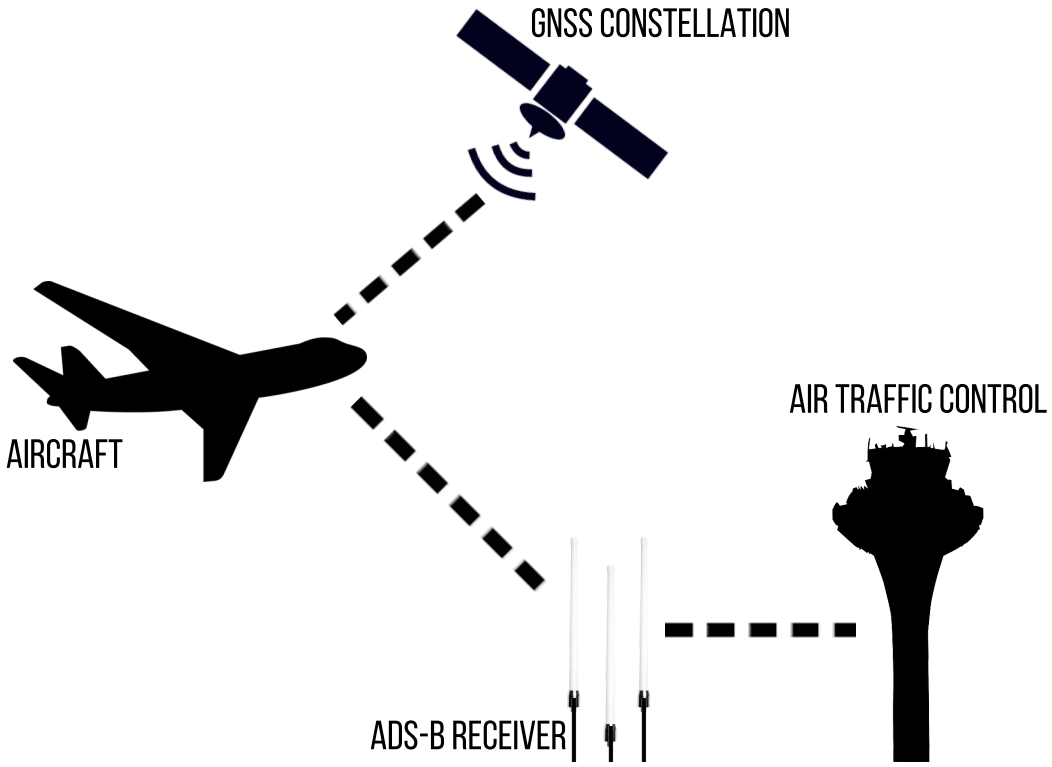

FIGURE 2.2: ADS-B Schematic

ADS-B Transmitter

ADS-B utilizes GPS satellites usually to relay coordinates to ground stations and other aircraft directly. This also known as ADS-B Out and is more accurate than using the standard system of radar monitoring. This eliminates the necessary separation gap between aircraft fitted with ADS-B by air traffic controllers. The International ICAO ADS-B standard is classified as 1090 MHz or more commonly called 1090ES (Extended Squitter). The ADS-B used that frequency to transmit the information. Now aircraft fitted with a Mode S transponder capable of ADS-B will use something called "Extended Squitter" to relay the ADS-B Out data, in common with airlines. The Extended Squitter is simply the extended communication range of the transponder. This includes an ADS-B information data packet. Each data package includes unique information on an aircraft, such as its location, speed, and identity.

2.3 ADS-B Receiver Kit

The kit on Figure 2.3, consists of the following parts:

1. The Raspberry-Pi
    - 1x Raspberry-Pi 3 Model B (A)
    - 1x Raspberry-Pi Universal Power Supply (B)
    - 1x Big Chip cooler (C)

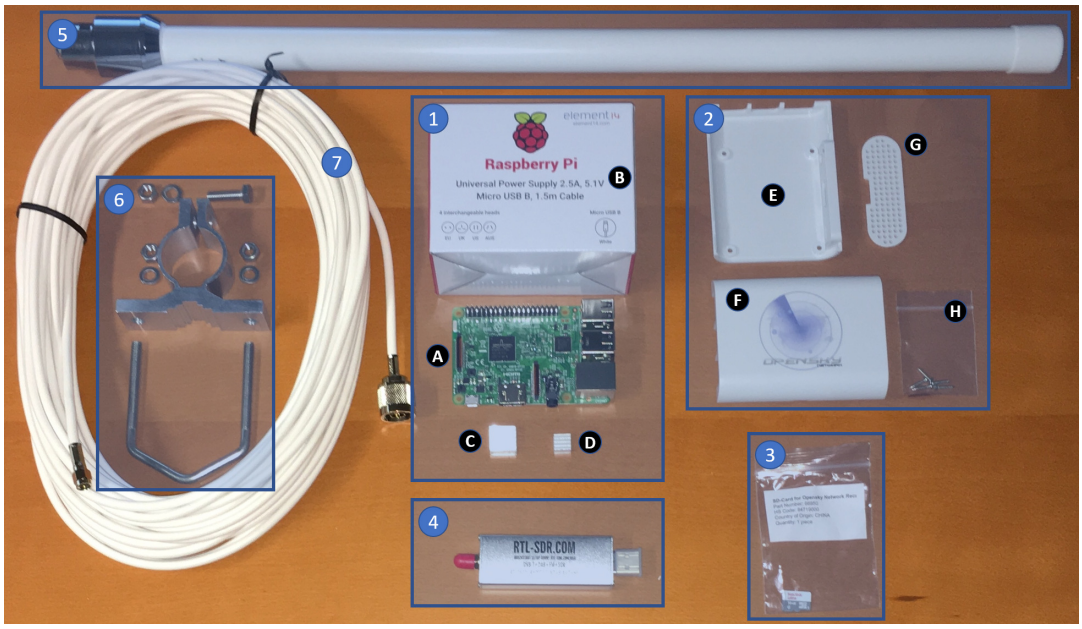

FIGURE 2.3: The OpenSky Network Kit, *source* : *www.github.com*

- 1x Small Chip cooler (D)

2. The Case
   - 1x OpenSky Raspberry-Pi Case (E), (F), (G)
   - 4x Screw + 1x spare screw (H)
3. 1x SD card containing the OpenSky receiver image
4. 1x RTL-SDR USB dongle
5. 1x Jetvision A3 ADS-B antenna 1090 MHz
6. 1x Antenna mounting bracket
7. 1x 15m low-loss coax cable

2.3.1 ADS-B Raspberry-Pi Receiver

The ADS-B receiver system is divided into 2 components; the Raspberry-Pi set, and the receiver. The Raspberry-Pi set components are Raspberry-Pi 3 Model B+, Memory Card 32GB MicroSDHC Class 10, 2 Heatsinks, and Raspberry-Pi Case. This Raspberry-Pi model 3 came with 4 USB 2 ports, Full-size HDMI, 1 GB RAM, Quad-Core 1.2GHz Broadcom 64bit CPU, Wireless LAN and Bluetooth Low Energy (BLE) on board, Micro SD port for loading the operating system and storing the data, and Micro USB power source up to 2.5A. Raspberry-Pi also has extra ports for CSI camera for connecting a Raspberry-Pi camera and a DSI display port for connecting a Raspberry-Pi touchscreen display. This port also can be installed a mini fan, but if 2 heat-sinks are already installed, the mini fan can be another option. Since the

Raspberry-Pi is only a computer, a mouse, a keyboard, and a display output is also important for doing the installation.

Electrical Data	**ADS-B 1090 MHz Antenna**
Type	Ground Plane Colinear 3x 1/2 λ
Design Frequency	1090 MHz (ADS-B system)
Impedance	50 Ω
Radiation (H-plane)	360°omnidirectional
Radiation (E-plane)	Beamwidth @ -3 dB: 23.8°
Polarization	Linear Vertical
Gain	4.8 dBi
Maximum Transmit Power	10 W cw, 500 W peak
Grounding Protection	All metal parts are DC-grounded, the inner conductor shows an open circuit
Connector	N-female, gold plated central pin

TABLE 2.1: Specification ADS-B Antennas 1090 MHz

The Receiver components are ADS-B Antenna with N Female Connector, Cable N Male – SMA Male, and USB RTL-SDR. The ADS-B Antenna component is using the Jetvision A3 ADS-B Antenna 1090 MHz with the specification on Table 2.1 which also suitable for the RTL-SDR R820T2 RTL2832U. This dongle is a good choice to be used as a stable ADS-B receiver (1090 MHz). The N Male - SMA Male Cable is the cable to connect between the antenna to the RTL-SDR. Since the RTL-SDR dongle is also powered using 4.5 V USB, it does not need another connector to the Raspberry-Pi.

2.3.2 ADS-B Message

There are several methods that the antenna can receive the ADS-B message. Usually, the ADS-B message comes with 112 bits long, but there are many ways to be able to read the ADS-B message. For example, the dump1090 stream parser method provides a method to collect the ADS-B message. Dump1090 stream parser is a software that takes a dump1090 stream of ADS-B messages and plops them into an SQLite database with a timestamp.

Dump1090 is a Mode S decoder designed for the RTL-SDR USB dongle devices. This needs to be installed and checked out first before use the dump1090.

Some other methods to collect the ADS-B message is by using the dump1090 and start the dump1090 in the interactive mode from the command prompt/terminal. This will create another collected message in the directory of the computer.

The database message is basically an SQLite database message. It needs `sqlite3` library on python, in order to read the actual message and to be able to understand the message. There are 23 fields in the SQLite database message [14].

Each message comes with different outputs. For some message is heavy and for some is light to read, depends on how long the data have been gathered and the position of the antenna. It will take more time to be able to read or process the data if the data is heavy.

2.4 Dump1090 Stream Parser

Before start using the stream parser, the data was gathered using the interactive mode method. As seen on Figure 2.4a, the interactive mode comes with a `(.txt)` file format which can only be display, not read the data. The streamed data are cluttered and are not readily accessible to be processed. Because the data cannot be read, then the data cannot also trough some filtering or slicing.

```
[H[2JHex     Mode  Sqwk  Flight   Alt    Spd  Hdg    Lat      Long   Sig  Msgs   Ti-
------------------------------------------------------------------------------
8A0650  S                         170  338                     47     1    0
8A02D5  S                                                      23     1    0
[H[2JHex     Mode  Sqwk  Flight   Alt    Spd  Hdg    Lat      Long   Sig  Msgs   Ti-
------------------------------------------------------------------------------
8A0604  S                    4375                              29     2    0
8A0521  S                         216  241                     56     1    0
8A0650  S                         170  338                     47     1    0
8A02D5  S     4640                                             24     2    0
[H[2JHex     Mode  Sqwk  Flight   Alt    Spd  Hdg    Lat      Long   Sig  Msgs   Ti-
------------------------------------------------------------------------------
8A0289  S                    7975                              21     1    0
8A0604  S                    4375                              28     3    0
8A0521  S                    6100  216  241                     49     7    0
8A0650  S                         170  338                     47     1    0
8A02D5  S     4640                247  248                     24     3    0
[H[2JHex     Mode  Sqwk  Flight   Alt    Spd  Hdg    Lat      Long   Sig  Msgs   Ti-
------------------------------------------------------------------------------
8A0291  S                   13600                              21     1    0
8A0243  S                   20175                              18     1    0
8A0582  S                   17500                              48     2    0
8A02D3  S                         322  187                     25     1    0
8A0289  S                    7975  286  111                     21     2    0
8A0604  S                    4375  221  155                     26     7    0
8A0521  S                    6100  216  241                     46     8    0
8A0650  S                         170  338                     47     1    0
8A02D5  S     4640                247  248                     24     3    0
[H[2JHex     Mode  Sqwk  Flight   Alt    Spd  Hdg    Lat      Long   Sig  Msgs   Ti\
------------------------------------------------------------------------------
```

(A) Interactive mode with `(.txt)` format file

```
        message_type  transmission_type session_id aircraft_id hex_ident  \
0                MSG                  7        111       11111    000000
1                MSG                  7        111       11111    000000
2                MSG                  7        111       11111    000000
3                MSG                  4        111       11111    8A05CD
4               \rMSG                 8        111       11111    8A05CD
5                MSG                  4        111       11111    8A05CD
        flight_id generated_date generated_time logged_date   logged_time  \
0          111111     2019/09/02   13:03:57.234  2019/09/02  13:03:57.244
1          111111     2019/09/02   13:04:56.286  2019/09/02  13:04:56.296
2          111111     2019/09/02   13:05:55.338  2019/09/02  13:05:55.347
3          111111     2019/09/02   13:06:30.712  2019/09/02  13:06:30.671
4          111111     2019/09/02   13:06:31.632  2019/09/02  13:06:31.589
5          111111     2019/09/02   13:06:31.751  2019/09/02  13:06:31.719
        callsign altitude ground_speed track      lat      lon vertical_rate  \
0
1
2
3                              340   105                           3712
4
5                              340   105                           3712
        squawk alert emergency spi is_on_ground                parsed_time
0                                          \r  2019-09-02T06:03:15.492876
1                                          \r  2019-09-02T06:03:57.254830
2                                          \r  2019-09-02T06:04:56.308216
3                                           0  2019-09-02T06:05:55.357018
4                                           0  2019-09-02T06:06:30.683801
5                                           0  2019-09-02T06:06:31.597683
```

(B) After Parsed `(.db)` format file

FIGURE 2.4: Before and After Parser

In this study, the parser takes the signals and builds a new data structure in the form of a database. Parser it self is a compiler that breaks data into smaller elements for easy translation into another language. On Figure 2.4b is example from using the stream parser.

2.4.1 SQLite Database

The stream parser software uses a dump1090 stream of ADS-B message and plops them into an SQLite database. The database will create and start populating an SQLite database named `adsb_messages.db` in the current directory of the computer.

Without the stream parser, the message can only be view on display as seen on Figure 3.1. The message is also not saved in the memory. As soon the aircraft is out of range from the antenna, the aircraft information can not be seen.

It can start the SQLite by running on the current machine, using the terminal/-command prompt by typing `pythondump1090-stream-parser.py` [15]. The `dump1090-stream-parser.py` contains a python coding that can start parsing the data automatically.

The data is stored in a table called squitters. The message contains 23 fields with different information. The fields are the Message Type, Transmission Type, Session ID, Aircraft ID, Hex Ident, Flight ID, Generated Time, Generated Date, Logged Time, Logged Date, Callsign, Altitude, GroundSpeed, Track, Latitude, Longitude, Vertical Rate, Squawk, Alert, Emergency, SPI (Ident), Is On-Ground, and Parsed Time.

There are six types of the Message Type - MSG, SEL, ID, AIR, STA, and CLK. The MSG (Transmission Message) means that the message is generated by the aircraft and also there are eight different types of MSG. The SEL (Selection Change Message) means that the message is generated when the user changes the selected aircraft in Base Station. The ID is New ID Message type, which is generated when an aircraft being tracked sets or changes its callsign. The AIR (New Aircraft Message) is generated when the SBS (Surveillance and Broadcast Services) picks up a signal for an aircraft that it isn't currently tracking. The STA (Status Change Message) is generated when an aircraft's status changes according to the time-out values in the Data Settings menu. And the last type is CLK (click message). CLK is generated when the user double-clicks on an aircraft.

The Transmission Type is not used by other messages types. This Transmission Type is like MSG subtypes 1 to 8. The session ID is the database session record number, the Aircraft ID is the database aircraft record number, and the Flight ID is database flight record number. The Hex Ident is the Aircraft Mode S hexadecimal code, similar to the ICAO24.

The Callsign is an eight-digit flight ID. It can be a flight number or flight registration. The Altitude is a Mode C altitude. Height is relative to 1013.mb (Flight Level). The Ground Speed field is the speed over ground, which is not indicated airspeed. The Track field is the track of the aircraft (not heading). The Track field is derived from the velocity East/West and velocity North/South. The next field is Latitude which is North and East for the positive value, and South and West for the negative value. The value is also applied for the Longitude field. The Vertical Rate field is the 64ft resolution. The Squawk field is assigned Mode A squawk code. The next field is Alert, which is indicated for changing squawk. The next field is Emergency. The Emergency field is a sign to indicate emergency code has been set. The SPI is a sign to indicate transponder Ident has been activated. The Is On Ground field is also a sign to indicate the ground squat switch. The last field is Parsed Time. This field gives information about when the data is being parsed.

Chapter 3

Research Methodology

The steps taken in this research are based on the steps that contained in the manual inside the ADS-B Raspberry-Pi kit. The steps shows how to assemble, the installation, and other guide that demonstrate how to collect the signals or messages. The guide also available on the official website [7].

3.1 Assembling The ADS-B Raspberry-Pi System

As seen on Figure 2.3, the components are mostly already in pieces, except for the Raspberry-Pi. The Raspberry-Pi needs to be assemble. There are 2 cooler chip for the Raspberry-Pi that needs to be place on the Raspberry-Pi. After that, Raspberry-Pi can be placed inside the case.

The SD Card is installed with the preinstalled software. Because of the interface, the SD Card will be filled with NOOBS (New Out Of the Box Software). NOOBS is an OS installer which contains Raspbian. With Raspbian, the interface will more like a normal computer, which make it easier to operate.

3.1.1 Setting up the Raspberry-Pi

First off, the SD card needs to be formated by using any SD memory card formatter software. Insert the SD card into the PC using the SD adapter. After that, format the SD card. Make sure that the SD card is selected for the correct drive letter. The SD card can be formatted with a full erase option and format it with the size adjustment off. After done formatting, the SD card, copy the downloaded-unzipped NOOBS files to the formatted SD card. Safely eject the SD card is important to prevent the error happened on the SD card later on.

After safely eject the SD card from the PC, the SD card can be inserted into the Raspberry-Pi. Connecting the power supply must be connected after inserting the SD card. Same with other CPU, Raspberry-Pi also needs to connect with the HDMI cable to the monitor (since the Raspberry-Pi only had an HDMI output) and a keyboard and mouse are also important. After connecting with all the needs, continue

with performing the Raspbian installation. It will offer a choice when the Raspberry-Pi is on. There should be a box for Raspbian. Check the box, and then click install. There will be some warning dialog, and just need to click Yes and then continue the installation by itself. It will take a while. When the Raspbian has been installed, proceed with click Ok and the Raspberry-Pi will restart and then boot up by itself. The Raspberry-Pi with Raspbian software is ready to go.

To make sure that the Raspbian is up to date, connect the Raspberry-Pi with an internet connection. After that, open the command line and type:

```
sudo apt-get update
sudo apt-get upgrade
```

Install the git-core and git by typing:

```
sudo apt-get install git-core git
```

After some update and upgrades are installed, reboot the Raspberry-Pi, and the Raspberry-Pi is ready to be used for setting up the RTL-SDR drivers and the dump1090 [7].

3.1.2 Setting up the RTL-SDR and Dump1090

There are many Raspberry-Pi ADS-B installers that provided many websites. In this project, the RTL-SDR drivers and dump1090 are checked out from the Raspberry-Pi-project that provided on Github.

The first thing to do is to download the package from the GitHub. Download the package by typing:

```
cd ~
git clone https://github.com/openskynetwork/raspberry-pi-adsb.git
```

The package can be download because of the git-core and git installation. The `raspberry-pi-adsb.git` is the package to set up the RTL-SDR and dump1090. First, install the RTL-SDR. To install the RTL-SDR receiver, it needs to build and install by typing:

```
cd ~/raspberry-pi-adsb
./setup-rtl-sdr.sh
```

The installation will be prompted to reboot after the set up is done. After the system is rebooted, the RTL-SDR was compiled and installed successfully.

After the RTL-SDR is successfully installed, continue to install the dump1090. The dump1090 is the signal decoder that coming from the RTL-SDR receiver. To install and build the dump1090, type:

```
cd ~/raspberry-pi-adsb
./setup-dump1090.sh
```

The `setup-dump1090.sh` will install the dump1090 as a service so that the dump1090 starts automatically after the system restart. The Raspberry-Pi will be once again prompted to reboot after the installation. After the system is rebooted, check the dump1090 if it's running correctly. In the browser on Raspberry-Pi, type `http://127.0.0.1:8080` and there should be a map with planes, as seen on Figure 3.1. Just to make sure that the receiver works properly, this map is not very useful for further analysis. Even if the receiver can get the ADS-B signal, that does not mean the ADS-B data is saved in the Raspberry-Pi SD memory card. There is some stream parser package that needs to be installed in the Raspberry-Pi in order to be able to save the ADS-B data.

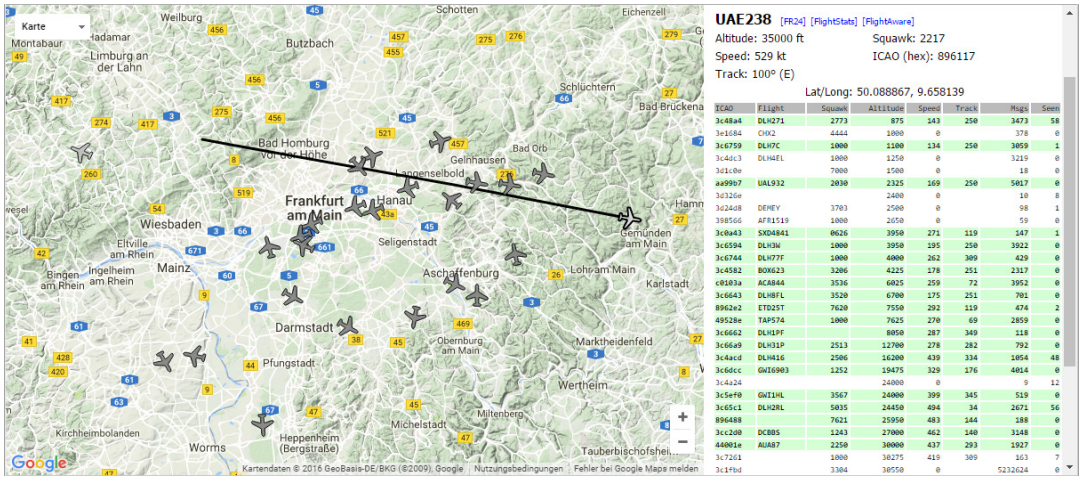

FIGURE 3.1: dump1090 via `http://127.0.0.1:8080`

3.2 Placing The Antenna

The antenna is implanted at Cluster Pasadena, Bukit Dago Housing Estate (6°21'35.8"S - 10°642'19.1"E).

The data collection starts at 09/26/2019 10:10 am. On Figure 3.3, the advantage of placing the antenna in this place have a clear view, which is there are no tall buildings, trees, and any other higher obstacles near around the antenna position. It is very recommended since the antenna has 360°omnidirectional.

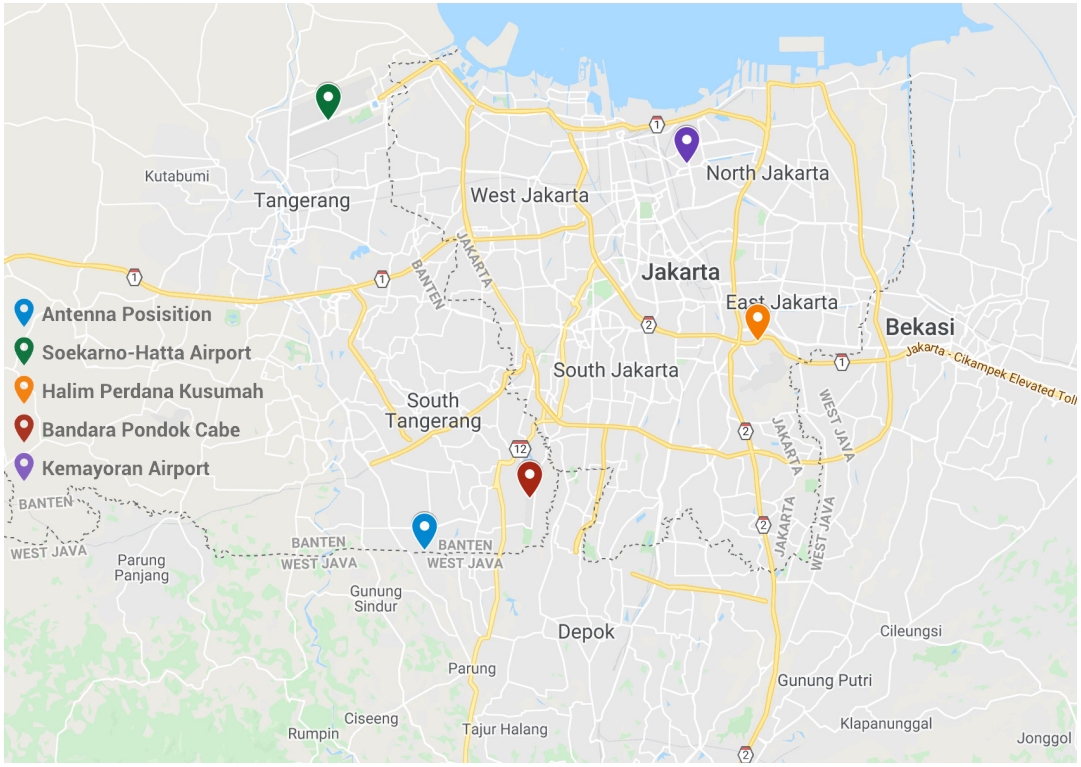

FIGURE 3.2: Position of the Antenna.

In Figure 3.2 the distance between the position of the antenna to Soekarno-Hatta International Airport is 26.4 km, from the antenna to Halim PerdanaKusuma Airport is 22.3 km, from the antenna to Kemayoran Airport is 27.5 km, and from the antenna to Bandara Udara Pondok Cabe is 6.86 km.

3.3 Preparing Raspberry-Pi ADS-B Box

The receiver system is not comes with a cooling system. 2 heat-sinks is not enough for cooling the Raspberry-Pi. It comes with an idea to made a box with cooling system, at least an air blow for both Raspberry-Pi and the RTL-SDR. As seen on Figure 3.4, this Raspberry-Pi ADS-B Box is made out of a shoes box, with a little fan attach on top of it. The box contains the Raspberry-Pi and the RTL-SDR. Both components are easily become hot when both of the components work for more than 24 hours.

3.4 Setting up the Stream Parser

This dump1090 Stream Parser software need a dump1090 stream of ADS-B messages and put them into an SQLite database with the human-timestamp. It will need a dump1090 running on the current machine.

First thing to do is to clone or download the py file on the website. Then, the cloned or downloaded file needs to be on the directory folder on the Raspberry-Pi.

FIGURE 3.3: The antenna was planted on position A.

After that, open the command line and type:

```
python dump1090-stream-parser.py
```

The py file contains a stream parser that enable the ADS-B signal saved into the Raspberry-Pi. The py file will create and start collecting and put it into the SQLite database called adsb_messages.db in the Raspberry-Pi current directory. In order to read the file, the data is stored in a table named squitters.

3.5 Processing the Data

The ADS-B message will be read by Chunk that is divided by its date. The original filtered by date will also stored in another CSV file, just in case, for further use. After the specific date is gathered, then removing the defect data and also divided by its Flight Level group. The defect data means the message that had no information about the altitude or the latitude and longitude during the plotting.

Each date is also filtered per hour in order to made it easier to be input to the new CSV file. The filtering processes are done for the purpose to reduce cluttering.

3.6 Air-Traffic Volume Analysis

In this study, each of the aircraft that entering the area was monitored and its path or trajectories were recorded using the collected ADS-B data. From the Latitude and

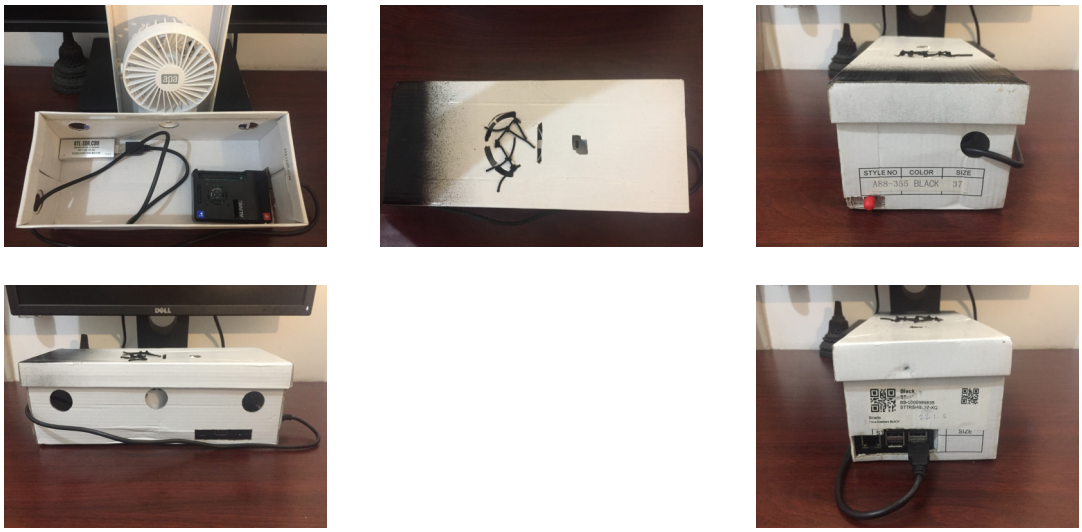

FIGURE 3.4: Raspberry-Pi ADS-B Home-made Box

the Longitude information, the aircraft can be plotted on any point or any place. This research will also investigate the traffics by the Altitudes of aircraft Flight Level (FL) to identify the traffic on each day.

3.6.1 Air-Traffic Volume Analysis based on Flight Level

The altitude or aircraft Flight Level (FL) was grouped into 6 flight level;

- Flight level 1 from 0 to 1000 feet
- Flight level 2 from 1001 to 2000 feet
- Flight level 3 from 2001 to 3000 feet
- Flight level 4 from 3001 to 4000 feet
- Flight level 5 from 4001 to 5000 feet
- Flight level 6 from 5001 feet and above

The obtained data will be categorize based on the vertical separation that a minimum altitude of 1,000 feet should be considered safe and no aircraft should approach more than 1,000 feet vertically to be monitored [16].

The collected data covered flight movement only if the receiver received. This study only focused on the available data. The defect data for the altitude or Flight Level is removed during the filtering process.

The defect data is removed because in this research, the position message takes two messages to get the latitude and longitude. So if the massage haven't got the second message yet, the message will get altitude but not latitude or longitude.

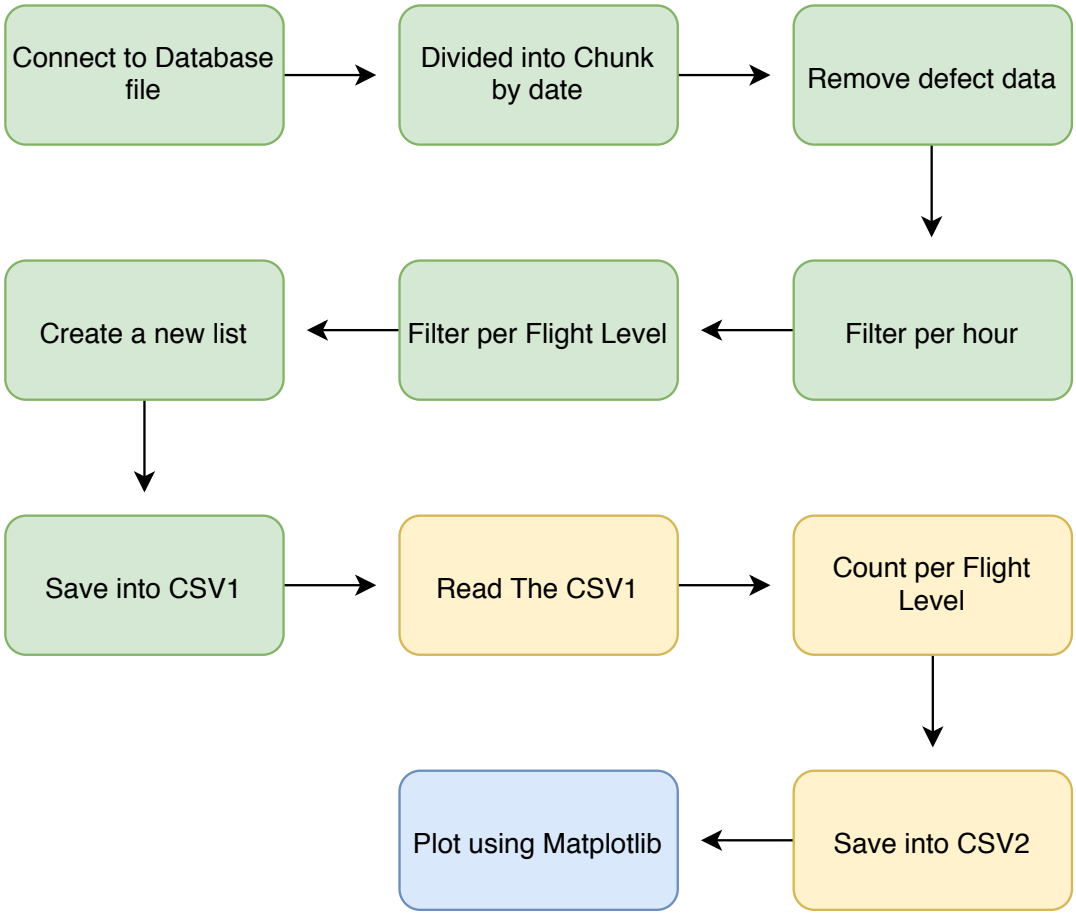

FIGURE 3.5: Flight Level Filtering Flow Chart

The Figure 3.5 shows the filtering process from the flight level filter until plotting the graph. Since the data is an SQLite Database, it needs to be read using sqlite3 library in python. The chunk process is divided by date, not using the chunksize method, because every chunk not contains with same date. Removing the defect data will make the data more smaller, make it more visible and lighter to be read. After removing the defect data, the next step is to divided the data with date into hours. The new list was create for the flight level that have been filtered and append for the next chunk, and then save into CSV1. Count per Flight Level is perform because the CSV1 totaled per hour, there is no total flight per day with specific flight level. After the count succeed, save into a new CSV2 to perform the plotting.

3.6.2 Air-Traffic Volume Analysis based on Latitude and Longitude

The Latitude and Longitude were also grouped based on the Flight Level. Through some filtering, the defect data also will be removed. To prove that the data is filtered correctly while filtering the data, the screening process also includes the date.

Figure 3.6 shows the filtering process for filtering the Latitude and Longitude that also group based on the Flight Level. The filtering process also start with read

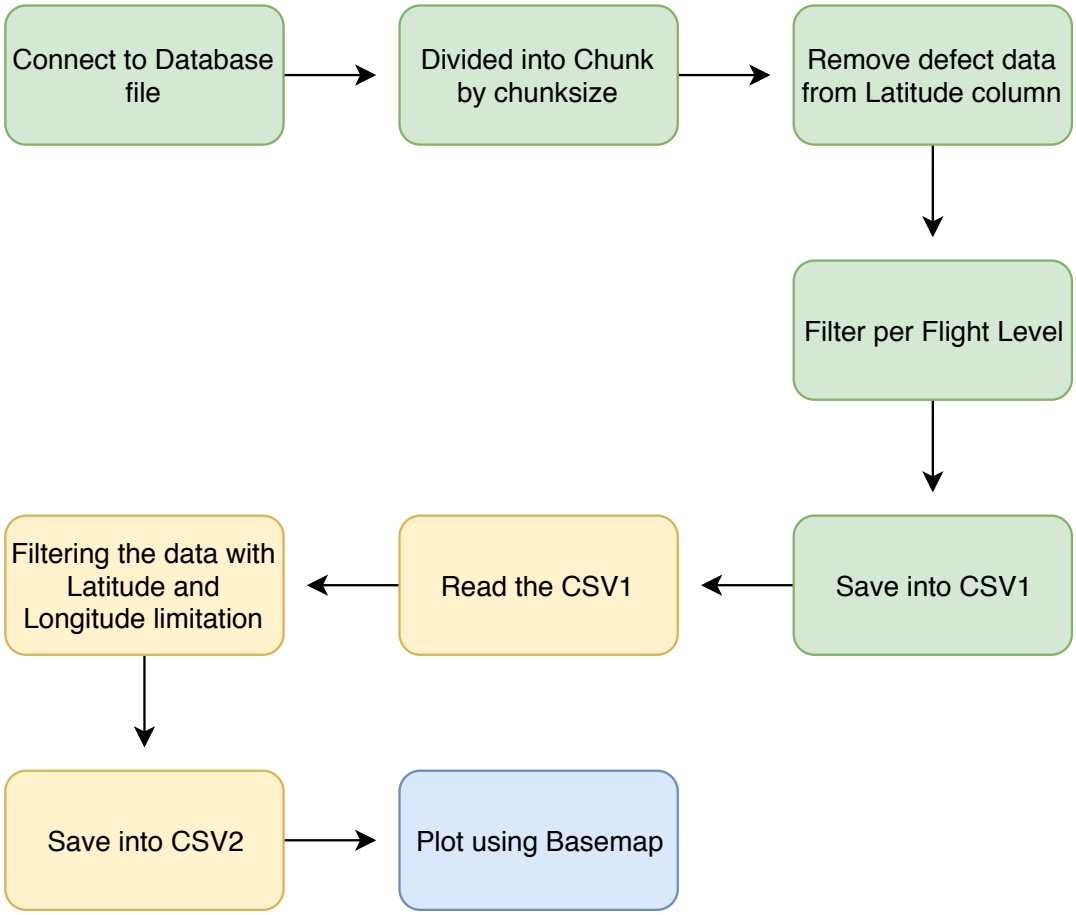

FIGURE 3.6: Latitude and Longitude Filtering Flow Chart

the Database file with sqlite3 library in python. The chunk process is divided by chunksize, which in this research the chunksize is 1 million per chunk. Remove the defect data from the Latitude will also remove the defect data for the Longitude. Filtering per flight level is also performed in this plot. After the filtering per flight level is complete, save into a new CSV1 file for limiting the Latitude and Longitude value. There are some Latitude and Longitude with a value that outside the Indonesia Latitude and Longitude, that makes the position that on the Indonesia territory looks small and less visible to be analyze. Save into another new CSV2, to be able to plot using the Basemap library.

Chapter 4

Results and Discussion

4.1 Obtained Data

The data is start collecting from the 26th of October with a period of 1 month collecting. It is as expected that the data will be large. At the end of the day, the data is around 12.03 GB. It also large enough to be read on python, so the data cannot be read in an ordinary way. Since the data is large, there are many ways to read the data and one of them is to read by chunk. Chunk method is by breaking a list into pieces of size N in python. The N is defined by a maximum request size per chunk.

After reading the data by chunk, the data has 89.338.383 million rows and 23 columns in total. There are a view days of electricity problem, that made the data need to be re-run in order to continue gathering the data. The data was cut from the 28th of September until the 1st of October. The receiver cannot be controlled from a distance, so the receiver needs to be re-run the stream parser manually. Overall, the data that have the information start from the 26th of September, 27th of September, and continues from the 2nd of October at 4 pm until the 25th of October. Unfortunately, the receiver is once again shut off automatically on the 26th of October at 1 am-midnight. In total, only 26 days of data that have the information for the ADS-B message.

Through the filtering process, there are 51.734.305 data that have the altitude information, and 15.563.095 data that have the altitude, latitude and longitude information. That is around 57.91% that have the altitude information, and 17.42% have the altitude, latitude and longitude information. The maximum altitude or flight level that the data have captured is at 61,600 feet and for the minimum is at 0 feet. During the filtering process, while gathering the altitude information not all the altitude attached with the latitude and longitude information, but while gathering the data for the latitude and longitude the altitude information attached with it.

4.2 Number of Flights Based on ADS-B data

In this research, the data set contains the latitude, longitude, and altitude of all flighted aircraft within more or less one month. Their unique ICAO number is defined for each aircraft. This includes all other flight information kept in a 0 to the

highest altitude and class of 6 Flight Level levels.

4.2.1 Air Traffic Based on Flight Level

The data that was collected was filtered and gathered on a different CSV file. The filtering code can be seen in Appendix A. Each file is named by their group of 6 different Flight Levels and containing date and number of unique aircraft. The graphics are made on scatters and colored with the same tone on each day. The results in Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5, and Figure 4.6 shows how many aircraft flew on the altitude of aircraft per day.

Flight Level from 0 to 1000 feet

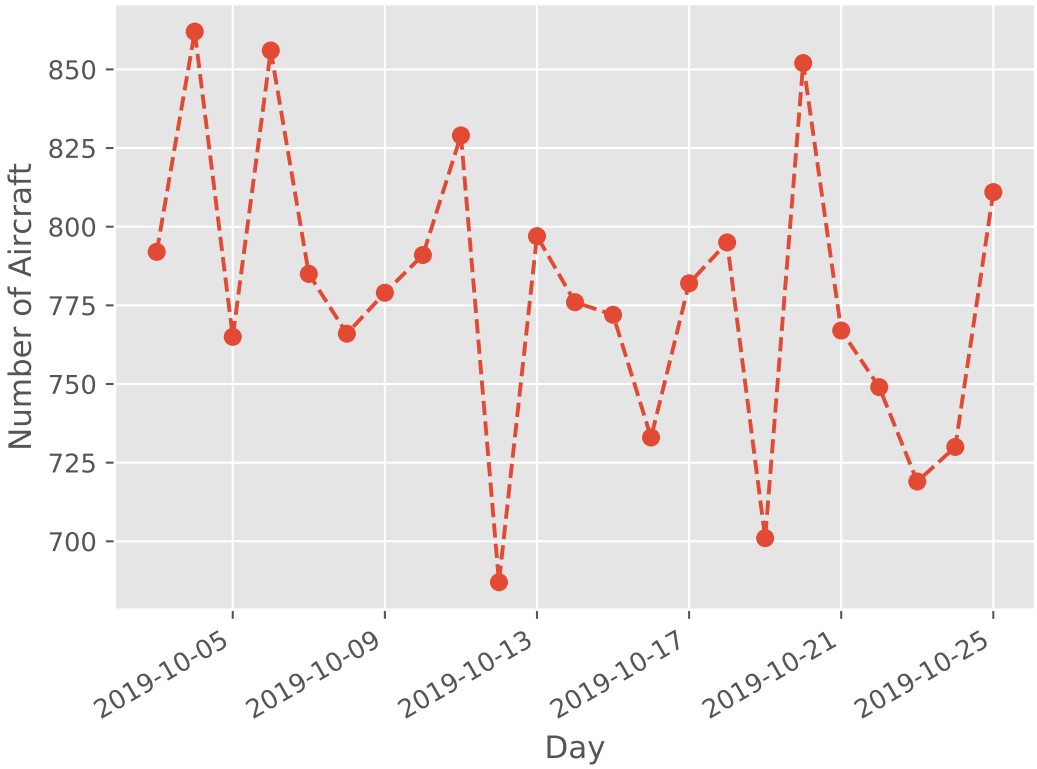

FIGURE 4.1: Number of Aircraft at Flight Level 1

From Figure 4.1, the graph shows the number of aircraft with altitude or flight level between 0 to 1000 feet. The figure shows that 4th October 2019 has the highest value and 12th October 2019 has the lowest number. Then, follow by 6th October 2019 has the second-highest value. The average of the total number of flights for 23 days is 778 per day.

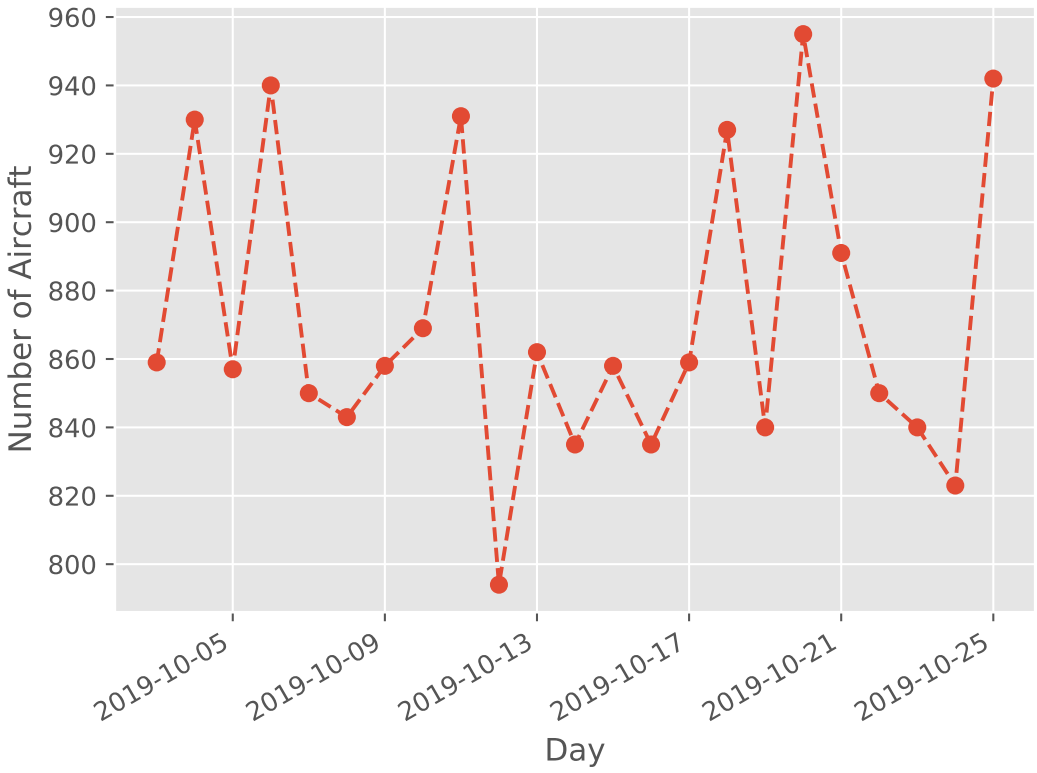

FIGURE 4.2: Number of Aircraft at Flight Level 2

Flight Level from 1001 to 2000 feet

The Figure 4.2 indicates the number of aircraft from altitude or flight level between 1001 to 2000 feet. The figure has an average of the total number of aircraft is 834 aircraft per day. In the figure above has the highest number on 20th October 2019 with a value of 955 and the lowest value is on 12th October 2019 with a value of 794 aircraft. On 9th October 2019 have a similar value with 15th October 2019 which is 858 aircraft on both days. The same value also happens on 19th October 2019 and 23rd October 2019 with 840 aircraft.

Flight Level from 2001 to 3000 feet

From Figure 4.3, the chart shows the number of aircraft at altitude or flight level between 2001 and 3000 feet. On 20th October 2019, the value is 960 aircraft which is the highest value and on 12th October 2019 has the lowest number. This has a similar case with 4.2 graph. On the 3rd of October 2019 has the same value as 16th October 2019 which is 834. It maybe looks similar, but on 5th October 2019 and on 9th October 2019 have a difference only 1 aircraft. The average of the total aircraft on flight level 3 is 842 per day.

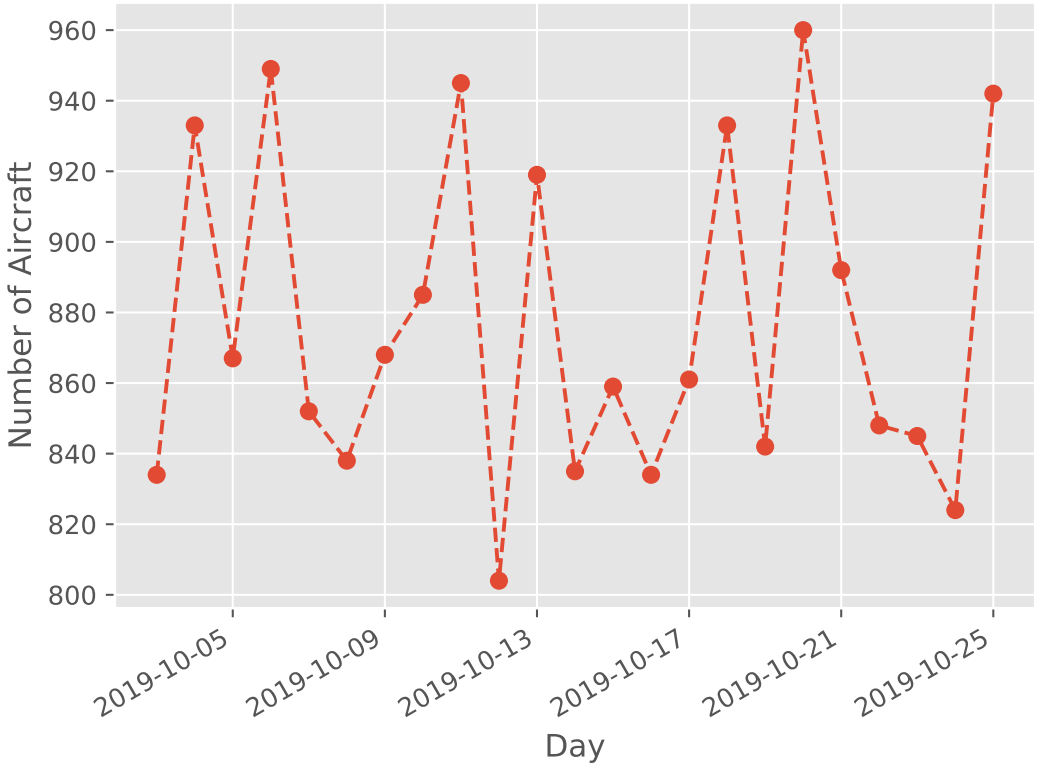

FIGURE 4.3: Number of Aircraft at Flight Level 3

Flight Level from 3001 to 4000 feet

Figure 4.4 shows the graph of altitude for flights between 3001 to 4000 feet. 13th October 2019 has the highest value with 906 aircraft. Then, follow by 11th October 2019 and 6th October 2019. Meanwhile, 12th October 2019 has the lowest value of the total number of aircraft compared to 24th October 2019, 15th October 2019 and 16th October 2019. The average of the total aircraft is 801 per day. 19th October 2019 and 22nd October 2019 has the same value, which is 756 aircraft.

Flight Level from 4001 to 5000 feet

Figure 4.5 indicates the number of aircraft from altitude or flight level between 4001 to 5000 feet. The figure shows an average of the total aircraft on flight level 5 is 767. 13th October 2019 has the highest value and on 19th October 2019 has the lowest value. 17th October 2019 and 23rd October 2019 has the same value which is 739 aircraft.

Flight Level from 5001 feet and above

Figure 4.6 indicates the number of aircraft from altitude or flight level between 5001 feet and above. The Figure 4.6 has the highest number of aircraft on 4th October 2019 which is 1262 aircraft, and the lowest number of aircraft is on 12th October 2019 which is 1049 number or aircraft. The average for this flight level si quite big

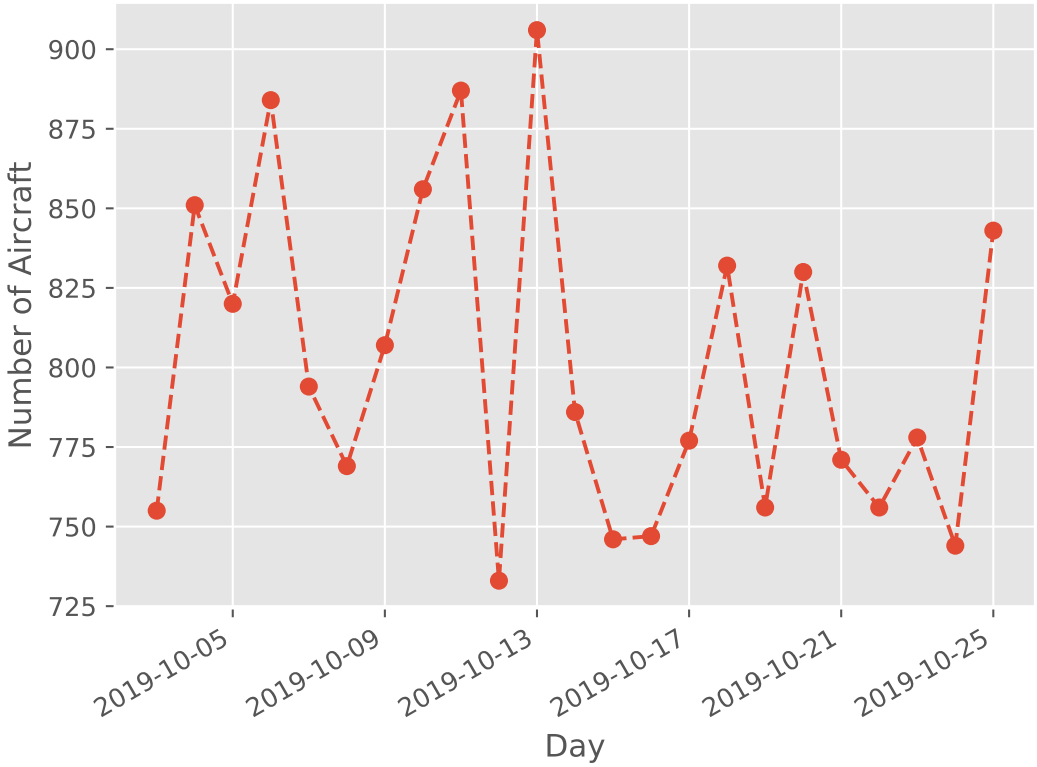

FIGURE 4.4: Number of Aircraft at Flight Level 4

since the data starts from 5001 and reach the highest flight level available on the data, which is 1193 number of aircraft per day.

On Figure 4.7 shows the averages and standard deviations. The exact value was provided on Table 4.1.

4.2.2 Air Traffic Based on Latitude and Longitude

The dataset contains information about all the Latitude and Longitude records that gathered by the ADS-B receiver. After a few filtering process, the data left with 15.563.095 information about the Latitude and Longitude only. The filtering codes refer to Appendix B. That value includes with the uncertainties of the dataset, that leave with doubts after plotting with Basemap. Each aircraft is identified by their unique Hex ID. It also divided by flight level that group into 6 groups of flight level.

Flight Level from 0 to 1000 feet

The 4.8 shows the location of aircraft with flight level from 0 to 1000 feet. There are 1166 of unique aircraft that have the Latitude and Longitude information which have the flight level between 0 to 1000 feet. As shown on the Figure 4.8, there are 2 group for aircraft that scattered around, but still around the island.

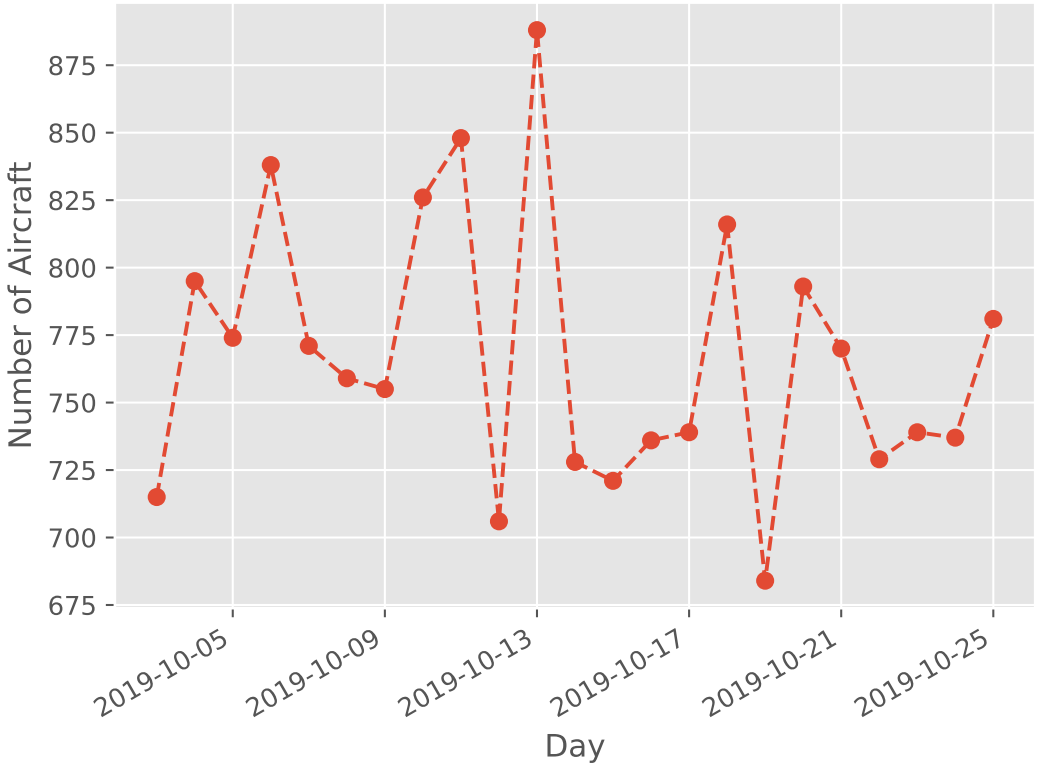

FIGURE 4.5: Number of Aircraft at Flight Level 5

Flight Level from 1001 to 2000 feet

Figure 4.9 indicates the location of the aircraft between flight level 1001 to 2000 feet. 1232 of unique aircraft that is on this plot that have the information about the Latitude and Longitude. The scattered aircraft as seen on Figure 4.8 is missing. There is a possibility that those aircraft are already landed, so the aircraft are not sending their information again after the transition is complete.

Flight Level from 2001 to 3000 feet

Figure 4.10 illustrates the location of the aircraft between flight level 2001 to 3000 feet. There are 1225 aircraft with different identity. From the latitude and longitude on the Figure 4.10, there is no aircraft that have a Longitude over 107.6°E and Latitude over 6.97°S. From Figure 4.9, the number of aircraft have decreased, but the path on Figure 4.10 is wider.

Flight Level from 3001 to 4000 feet

Figure 4.11 display the location of the aircraft between flight level 3001 to 4000 feet. In this Figure, there are 1193 of different aircraft with different identities. From Figure 4.11 the area of the aircraft is getting wider, not because of the total of the aircraft, but because there are more aircraft with flight level 3001 to 4000 feet and that have more information about their Latitude and Longitude.

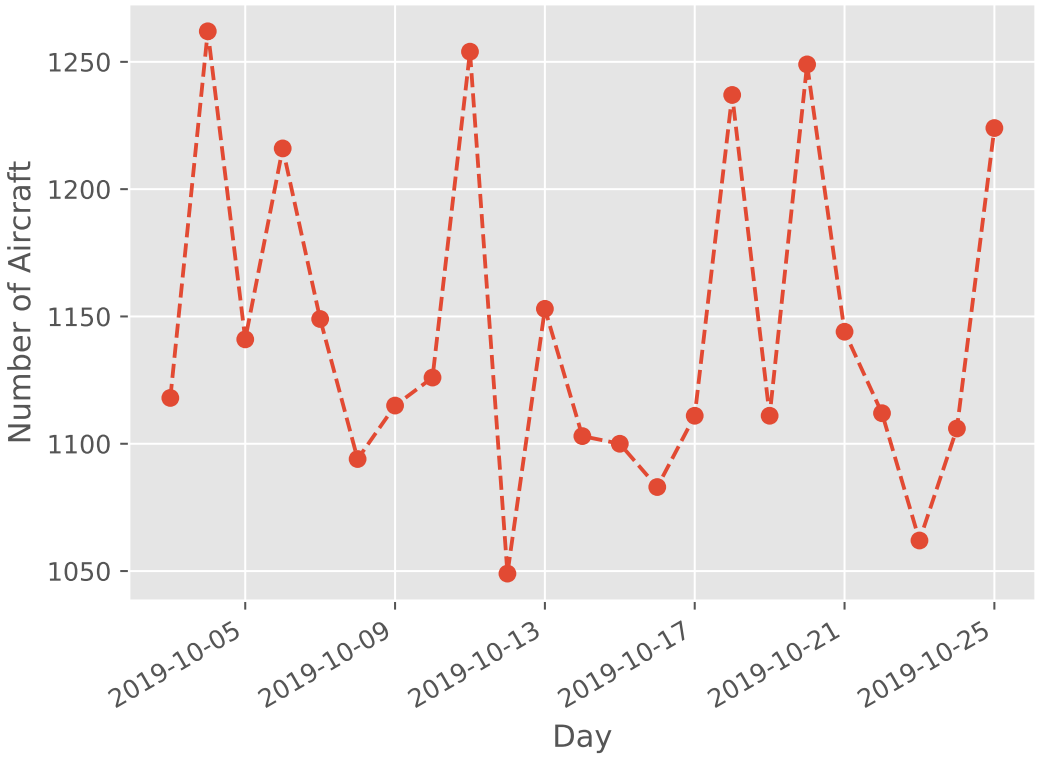

FIGURE 4.6: Number of Aircraft at Flight Level 6

Flight Level from 4001 to 5000 feet

Figure 4.12 depict the location of aircraft with flight level 3001 to 4000 feet. There are 1190 of different aircraft identities in this flight level. In this Figure, the aircraft is beginning to spread further and wider.

Flight Level from 5001 feet and above

The Figure 4.13 illustrates the amount of information or location of the aircraft with flight level 5001 feet and above. In this group, there is no limit maximum for the

Flight Level	Averages	Standard Deviations
FL1	778.086	45.07
FL2	871.652	43.13
FL3	876.913	46.083
FL4	801.217	50.251
FL5	767.304	49.292
FL6	1144.304	62.369

TABLE 4.1: Averages and Standard Deviations

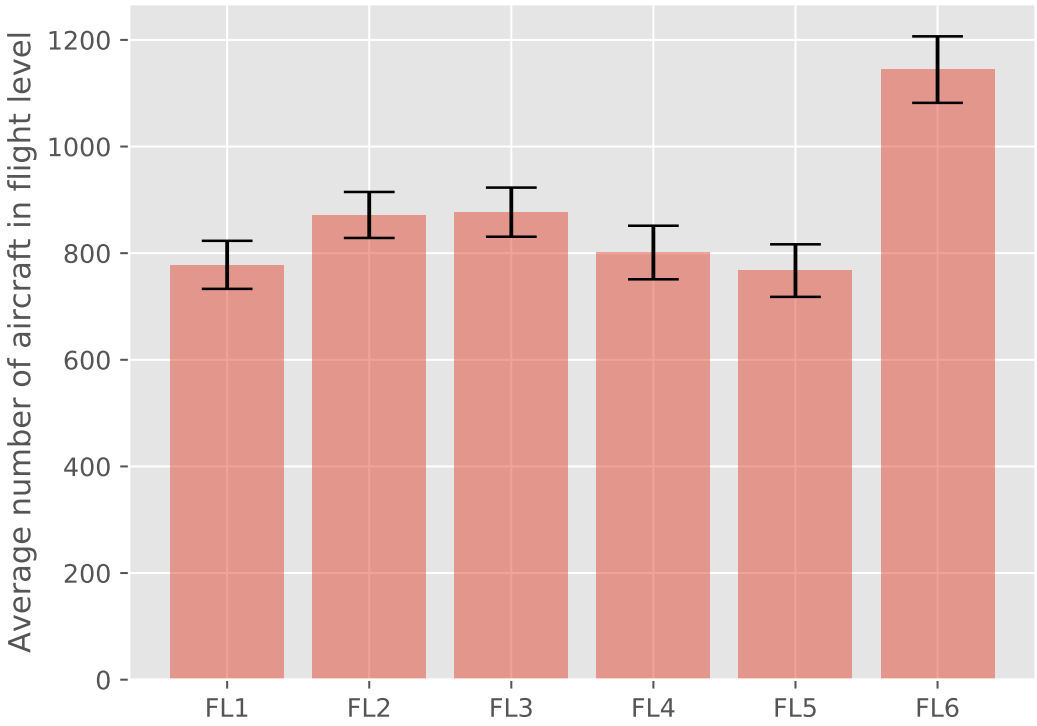

FIGURE 4.7: Air-Traffic Volume During the Collecting Period

altitude or flight level, as long as the information is available on the data. The aircraft with flight level 5001 feet and above which have the information about their Latitude and Longitude is 1502 of a unique Hex ID. As shown on the Figure 4.13, the scale becomes smaller in proportion to the amount of information about the position of the aircraft.

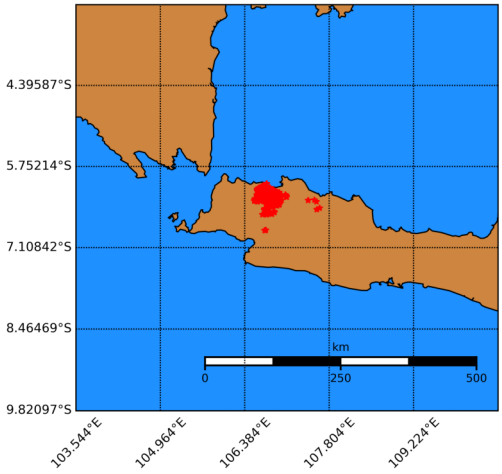

FIGURE 4.8: Flight Level 1 With Latitude and Longitude

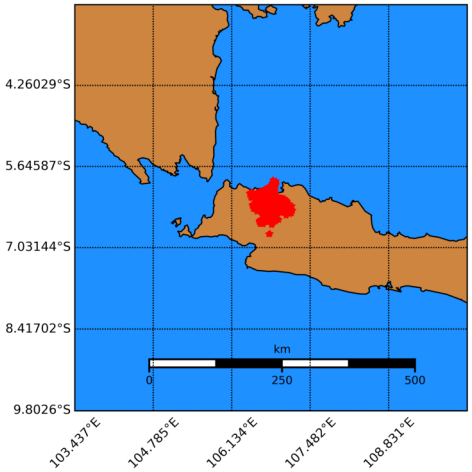

FIGURE 4.9: Flight Level 2 With Latitude and Longitude

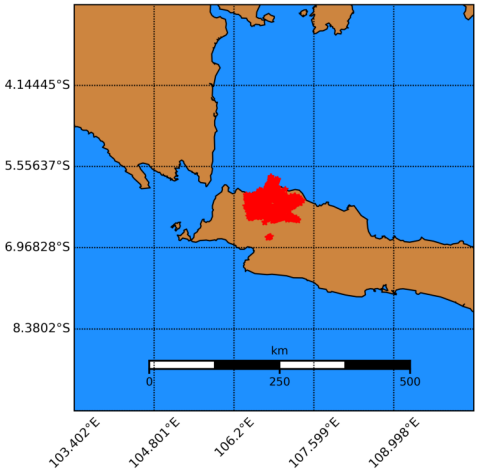

FIGURE 4.10: Flight Level 3 With Latitude and Longitude

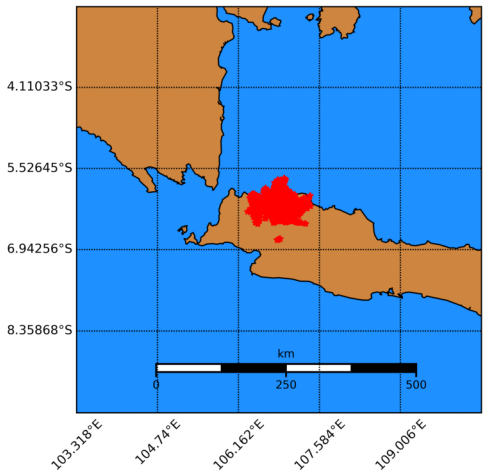

FIGURE 4.11: Flight Level 4 With Latitude and Longitude

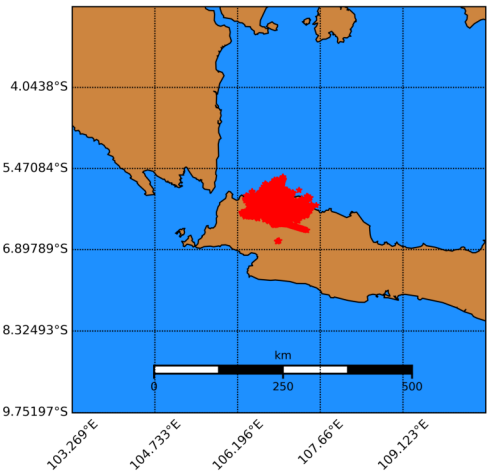

FIGURE 4.12: Flight Level 5 With Latitude and Longitude

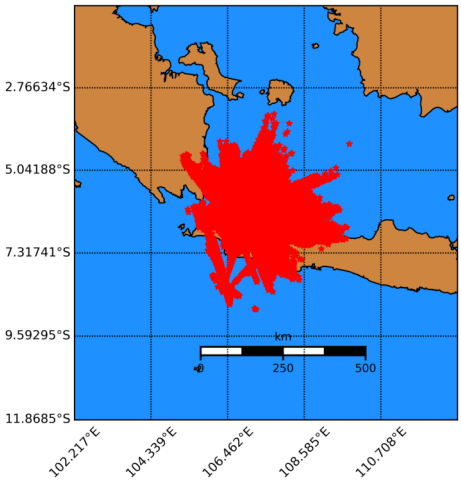

FIGURE 4.13: Flight Level 6 With Latitude and Longitude

Chapter 5

Conclusions

5.1 Conclusions

An analysis has been done according to the objectives of this study. This data is obtained using data from the accuracy level obtained from the ADS-B receiver as it is.

The ADS-B data was able to collect using the low-cost ADS-B receiver system. The stream parser could be the reason why the messages is not 100% filled with complete information; the ADS-B receiver is only receiving the captured data which then processed by the stream parser and be translated into a new structure of data. The stream parser works as a compiler and translate into a new structure of data which is making a stream parser is beyond this study purpose. From the collected ADS-B data, the air-traffic volume can be visualized and determined from the message that have been gathered using the ADS-B receiver system as it is.

5.1.1 Air-Traffic Volume by Analyzed the Number of Aircraft using Flight Level

The altitude or flight level distribution reported that the most common altitude category in which most aircraft fly is the altitude from 5001 feet and above. This can be seen on Figure 4.6, that the lowest Number of aircraft that fly on altitude from 5001 feet and above is 1049. This proves that the aircraft that fly below 5000 feet has a smaller result. The altitude or flight level that below 10,000 feet are either take off or landing position.

5.1.2 Air-Traffic Volume by Analyzed the Direction of Aircraft using Latitude and Longitude

The path-plot of latitude and longitude in Figure 4.13 shows a high volume of air traffic. Based on Figure 4.13, there are also some aircraft that have reached the latitude and longitude cross from Java island to Sumatra island. This also proved that the ADS-B receiver receives a message up to 250 km in radius. At category Flight Level 6, the flight path is spreading out more from the island since the altitude on Flight Level 6 had no limit.

5.2 Recommendations for Further Work

Because of the file size and many parameters of the collected ADS-B data, the receiver hardware that was used in this research has a thermal problem, especially the RTL-SDR. Hence, better receiver hardware will improve the receiving process. Therefore, parser with the better capability to decode all messages types and receive the high data sizes are strongly recommended for further study and for a longer time period. Better tools to process the big data in much more efficient way to parallel the computing of the big data that have a real-time air-traffic volume and tools to generate the flight path based on the ADS-B messages are also recommended.

Other studies such as performance assessment is also possible with some improvement on both hardware and software in order to get a better, bigger, and more complete data. An analyze on optimization of the receiver endurance with a better system is highly recommended.

Appendix A

Function library for Flight Level or Altitude Filtering

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import sqlite3 as lite
import sys
import csv

database = "adsb_messages.db"
query = "SELECT * FROM squitters WHERE logged_date = "
jump = 1000

pd.options.mode.chained_assignment = None

def delete_empty(df2,var):
    df2[var] = pd.to_numeric(df2[var], errors='coerce')
    df2[var].replace('', np.nan, inplace=True)
    df2 = df2.dropna(subset=[var], inplace=True)

def main():
    #VARIABLE
    low = 0
    num = 1
    total_count = 0
    total_count1 = 0
    total_count2 = 0
    total_count3 = 0
    total_count4 = 0
    total_count5 = 0
    total_count6 = 0
    dataTotal = []

    #Database Connect
```

```
38      sd = lite.connect(database)
39
40      #Chunk Divided by date which the date that starts from the first
        day collecting the data
41      date = ["'2019/09/26'","'2019/09/27'","'2019/09/28'","'2019/09/29'"
        ,"'2019/09/30'","'2019/10/01'","'2019/10/02'","'2019/10/03'","
        '2019/10/04'","'2019/10/05'","'2019/10/06'","'2019/10/07'","
        '2019/10/08'","'2019/10/09'","'2019/10/10'","'2019/10/11'","
        '2019/10/12'","'2019/10/13'","'2019/10/14'","'2019/10/15'","
        '2019/10/16'","'2019/10/17'","'2019/10/18'","'2019/10/19'","
        '2019/10/20'","'2019/10/21'","'2019/10/22'","'2019/10/23'","
        '2019/10/24'","'2019/10/25'","'2019/10/26'"]
42      i = 0
43      while i < len(date):
44          query2 = query + date[i]
45          print(query2)
46          print(date[i])
47          df = pd.read_sql_query(query2, sd)
48          print(df)
49          if df.empty == False:
50              #VARIABLE1
51              filename = str(num)
52              num = num + 1
53              df1 = df
54              start = 0
55              dataChunk = []
56
57              #For every hour
58              while start < 24:
59                  #VARIABLE2
60                  hex_count = 0
61                  hex_count1 = 0
62                  hex_count2 = 0
63                  hex_count3 = 0
64                  hex_count4 = 0
65                  hex_count5 = 0
66                  hex_count6 = 0
67                  start_str = str(start) + ':00'
68                  end_str = str(start+1) + ':00'
69
70                  #Filtering process for each hour, that took from the
        logged_time column
71                  df2 = df1[df1['logged_time'].between(start_str, end_str
        )]
72
73                  #Will Only Continue if the data is not empty
74                  if df2.empty == False:
75                      #Remove the empty altitude. The to_numeric process
        is needed, because the altitude is having a dtype=object
76                      df2['altitude'] = pd.to_numeric(df2['altitude'],
        errors='coerce')
77                      print(df2)
```

```
78                      df2['altitude'].replace(['',"NaN", 'NaT'], np.nan,
    inplace=True)
79                      print(df2)
80
81                      #If the data is not empty
82                      if df2.empty == False:
83                          df2.dropna(subset=['altitude'], inplace=True)
84                          print(df2)
85
86
87                          if df2.empty == False:
88                              #VARIABLE3
89                              high = int(df2['altitude'].max())
90                              curr = int(low)
91
92                              #Input to CSV. This CSV is only for each
    chunk.
93                              df2.to_csv('Chunk'+filename+'.csv', index=
    False, header=True)
94
95                              #Filtering with altitude range
96                              df3 = df2[df2['altitude'].between(0, 1000)]
97                              if df3.empty == False:
98                                  print(df3)
99                                  print("1st Iteration (Flight Level 0 to
     1000)")
100                                 print("Num. of Unique Hex: " + str(df3[
    'hex_ident'].nunique()))
101                                 print("Unique Hex: " + str(df3['
    hex_ident'].unique()))
102                                 print("
    ------------------------------------------------")
103
104                                 #For the unique hex count
105                                 hex_count1 = df3['hex_ident'].nunique()
106                                 total_count1 = total_count1 +
    hex_count1
107
108                             df3 = df2[df2['altitude'].between(1001,
    2000)]
109                             if df3.empty == False:
110                                 print(df3)
111                                 print("2n Iteration (Flight Level 1001
    to 2000)")
112                                 print("Num. of Unique Hex: " + str(df3[
    'hex_ident'].nunique()))
113                                 print("Unique Hex: " + str(df3['
    hex_ident'].unique()))
114                                 print("
    ------------------------------------------------")
115                                 hex_count2 = df3['hex_ident'].nunique()
```

```
116                                 total_count2 = total_count2 +
    hex_count2
117
118                             df3 = df2[df2['altitude'].between(2001,
    3000)]
119                             if df3.empty == False:
120                                 print(df3)
121                                 print("3rd Iteration (Flight Level 2001
     to 3000)")
122                                 print("Num. of Unique Hex: " + str(df3[
    'hex_ident'].nunique()))
123                                 print("Unique Hex: " + str(df3['
    hex_ident'].unique()))
124                                 print("
    ------------------------------------------------")
125                                 hex_count3 = df3['hex_ident'].nunique()
126                                 total_count3 = total_count3 +
    hex_count3
127
128                             df3 = df2[df2['altitude'].between(3001,
    4000)]
129                             if df3.empty == False:
130                                 print(df3)
131                                 print("4th Iteration (Flight Level 3001
     to 4000)")
132                                 print("Num. of Unique Hex: " + str(df3[
    'hex_ident'].nunique()))
133                                 print("Unique Hex: " + str(df3['
    hex_ident'].unique()))
134                                 print("
    ------------------------------------------------")
135                                 hex_count4 = df3['hex_ident'].nunique()
136                                 total_count4 = total_count4 +
    hex_count4
137
138                             df3 = df2[df2['altitude'].between(4001,
    5000)]
139                             if df3.empty == False:
140                                 print(df3)
141                                 print("5th Iteration (Flight Level 4001
     to 5000)")
142                                 print("Num. of Unique Hex: " + str(df3[
    'hex_ident'].nunique()))
143                                 print("Unique Hex: " + str(df3['
    hex_ident'].unique()))
144                                 print("
    ------------------------------------------------")
145                                 hex_count5 = df3['hex_ident'].nunique()
146                                 total_count5 = total_count5 +
    hex_count5
147
```

```
148                             df3 = df2[df2['altitude'].between(5001,
    high)]
149                             if df3.empty == False:
150                                 #df3.to_csv('Chunk'+filename+'-FL4.csv
    ', index=False, header=True)
151                                 print(df3)
152                                 print("6th Iteration (Flight Level 5001
     to " + str(high) + ")")
153                                 print("Num. of Unique Hex: " + str(df3[
    'hex_ident'].nunique()))
154                                 print("Unique Hex: " + str(df3['
    hex_ident'].unique()))
155                                 print("
    ------------------------------------------------")
156                                 hex_count6 = df3['hex_ident'].nunique()
157                                 total_count6 = total_count6 +
    hex_count6
158                 #VARIABLE4
159                 start = start + 1
160                 hex_count = hex_count1 + hex_count2 + hex_count3 +
    hex_count4 + hex_count5 + hex_count6
161                 total_count = total_count1 + total_count2 +
    total_count3 + total_count4 + total_count5 + total_count6
162                 time = start_str + " - " + end_str
163
164                 #Into list process with append.
165                 temp = [time,hex_count1,hex_count2,hex_count3,
    hex_count4,hex_count5,hex_count6,hex_count]
166                 dataChunk.append(temp)
167                 total_temp = [time,total_count1,total_count2,
    total_count3,total_count4,total_count5,total_count6,total_count]
168                 dataTotal.append(total_temp)
169
170                 #Into the CSV which divided by the Flight Level
171                 df4 = pd.DataFrame(dataChunk, columns = ['Time', '
    0-1000', '1001-2000', '2001-3000', '3001-4000', '4001-5000', '5001+
    ', 'Count'])
172                 df4.to_csv(str(i)+'-Count.csv', index=False, header=
    True)
173
174         i = i + 1
175     df5 = pd.DataFrame(dataTotal, columns = ['Time', '0-1000', '
    1001-2000', '2001-3000', '3001-4000', '4001-5000', '5001+',  'Count
    '])
176     df5.to_csv('TotalCount.csv', index=False, header=True)
177
178 if __name__ == "__main__":
179     main()
180
181 # Plotting the graph
182
183 import numpy as np
```

```
184 import matplotlib.pyplot as plt
185 from matplotlib.dates import strpdate2num, num2date, datestr2num
186 import matplotlib.dates as mdates
187 plt.style.use('ggplot')
188 import datetime
189
190 def convert_date(date_bytes):
191     return mdates.strpdate2num('%m/%d/%Y')(date_bytes.decode('ascii'))
192
193
194 data = np.genfromtxt("Data_Number_Airplanes_Gede.csv", skip_header=1,
        converters = {0: convert_date}, delimiter=";")
195
196 avg_fl_1 = np.average(data[:, 1])
197 avg_fl_2 = np.average(data[:, 2])
198 avg_fl_3 = np.average(data[:, 3])
199 avg_fl_4 = np.average(data[:, 4])
200 avg_fl_5 = np.average(data[:, 5])
201 avg_fl_6 = np.average(data[:, 6])
202
203
204 sd_fl_1 = np.std(data[:, 1])
205 sd_fl_2 = np.std(data[:, 2])
206 sd_fl_3 = np.std(data[:, 3])
207 sd_fl_4 = np.std(data[:, 4])
208 sd_fl_5 = np.std(data[:, 5])
209 sd_fl_6 = np.std(data[:, 6])
210
211
212 FLs = ['FL1', 'FL2', 'FL3', 'FL4', 'FL5', 'FL6']
213 x_pos = np.arange(len(FLs))
214 avgs = np.array([avg_fl_1, avg_fl_2, avg_fl_3, avg_fl_4, avg_fl_5,
        avg_fl_6])
215 sds = np.array([sd_fl_1, sd_fl_2, sd_fl_3, sd_fl_4, sd_fl_5, sd_fl_6])
216
217
218 # Average and standard deviation plot
219 fig, ax = plt.subplots()
220 ax.bar(x_pos, avgs, yerr=sds, align='center', alpha=0.5, ecolor='black'
        , capsize=10)
221 ax.set_ylabel('Average number of aircraft in flight level')
222 ax.set_xticks(x_pos)
223 ax.set_xticklabels(FLs)
224 ax.yaxis.grid(True)
225
226
227 # fig, ax = plt.subplots()
228 # ax.plot_date(data[:, 0], data[:, 6], 'o--')
229 # fig.autofmt_xdate()
230
231 # ax.fmt_xdata = mdates.DateFormatter('%Y-%m-%d')
232
```

```
233 # ax.set_xlabel("Day")
234 # ax.set_ylabel("Number of Aircraft")
235
236 # plt.savefig("FL6.pdf", dpi=600, bbox_inches='tight')
237
238 plt.savefig("avg_sds.pdf", dpi=600, bbox_inches='tight')
239
240
241 plt.show()
```

Appendix B

Function library for Latitude and Longitude Filtering

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import sqlite3 as lite
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import csv
from matplotlib_scalebar.scalebar import ScaleBar
import matplotlib.pyplot as plt
import matplotlib.cbook as cbook

#1st connect to the database file. The database file has to be in the
    file directory in the computer.
sd = lite.connect('adsb_messages.db')

#Read the file into chunk, in this case, the chunk didn't neet to be
    divided by date. Chunksize is enough.
for chunk in pd.read_sql_query("SELECT hex_ident, logged_date, altitude
    , lat, lon FROM squitters", sd, chunksize=1000000):
    chunk['lat'].replace('',np.nan,inplace=True)
    chunk.dropna(subset=['lat'],inplace=True)
    dataChunk.to_csv('file_name1.csv', mode='a')
#Save into new csv
#-----------------------------------------------------------------
#Read the new csv
df = pd.read_csv('filtered.csv')

#Turn the altitude data type from object to float46, so the altitude
    can be filter using between method.
df['altitude'] = pd.to_numeric(df['altitude'], errors='coerce')

#filter using between method
```

```
34 FL1 = df[df['altitude'].between(0,1000)] #the number in between bracket
       need to be change every Flight Level.
35 FL1.to_csv('file_name2.csv')
36 #save into new csv2
37 #saving into csv after every filtration is to prevent the missing data
       after filtration.
38 #-----------------------------------------------------------------------
39 #read the new csv2
40 sd = pd.read_csv('file_name2.csv')
41
42 #the 'lon' can be change into 'lat'. Also the number '111' change
       depend on the limitation
43 sd = sd[(sd['lon'] <= 111) | (sd['lon'].isnull())]
44
45 sd.to_csv('file_name3.csv')
46 #save into csv3
47 #-----------------------------------------------------------------------
48 #PLOTTING WITH BASEMAP
49
50 lats, lons,names,altitude = [],[],[],[]
51
52 # the asos_stations file can be found here:
53 # https://engineersportal.com/s/asos_stations.csv
54 with open('FixFL1.csv') as csvfile:
55     reader = csv.DictReader(csvfile,delimiter=',')
56     for data in reader:
57         names.append(data['hex_ident'])
58         lats.append(float(data['lat']))
59         lons.append(float(data['lon']))
60         altitude.append(float(data['altitude']))
61
62 # How much to zoom from coordinates (in degrees)
63 zoom_scale = 3
64
65 # Setup the bounding box for the zoom and bounds of the map
66 bbox = [np.min(lats)-zoom_scale,np.max(lats)+zoom_scale,        np.min(
       lons)-zoom_scale,np.max(lons)+zoom_scale]
67
68 plt.figure(figsize=(12,6))
69 # Define the projection, scale, the corners of the map, and the
       resolution.
70 m = Basemap(projection='merc',llcrnrlat=bbox[0],urcrnrlat=bbox[1],
             llcrnrlon=bbox[2],urcrnrlon=bbox[3],lat_ts=10,resolution='i'
       )
71
72 # Draw coastlines and fill continents and water with color
73 m.drawcoastlines()
74 m.fillcontinents(color='peru',lake_color='dodgerblue')
75
76 #m.drawmapscale(lon,lat,  lon0,  lat0,length,barstyle)
77 m.drawmapscale(108., -9., -3.25, .5, 500, barstyle='fancy')
78 #m.drawmapscale(lon,lat,  lon0,  lat0,length,fontsize)
```

```
79 #m.drawmapscale(108., -9, -3.25, 39.5, 500, fontsize = 14)
80
81 # draw parallels, meridians, and color boundaries
82 m.drawparallels(np.arange(bbox[0],bbox[1],(bbox[1]-bbox[0])/5),labels
       =[1,0,0,0])
83 m.drawmeridians(np.arange(bbox[2],bbox[3],(bbox[3]-bbox[2])/5),labels
       =[0,0,0,1],rotation=45)
84 m.drawmapboundary(fill_color='dodgerblue')
85
86 # build and plot coordinates onto map
87 x,y = m(lons,lats)
88 m.plot(x,y,'r*',markersize=5)
89 #plt.title("")
90 plt.savefig('FixFL1scale.pdf', format='pdf', dpi=600)
91 plt.show()
```

Bibliography

[1] "Air Traffic Analysis". In: (2012). URL: http://geoanalytics.net/and/papers/mobibook13a.pdf//// (visited on 12/20/2019).

[2] "An Approach to Air Traffic Density Estimation and Its Application in Aircraft Trajectory Planning". In: (2012). URL: https://engineering.purdue.edu/~jianghai/Publication/CCDC2012_ATC.pdf//// (visited on 05/24/2012).

[3] "Kalahkan Changi, Penerbangan di Soetta Setiap 66 Detik". In: (2018). URL: https://www.cnbcindonesia.com/news/20180427113940-4-12750/kalahkan-changi-penerbangan-di-soetta-setiap-66-detik////.

[4] "COMPARISON OF SURVEILLANCE TECHNOLOGIES". In: (2010). URL: http://geoanalytics.net/and/papers/mobibook13a.pdf//// (visited on 12/10/2010).

[5] "Preliminary Study on Air Traffic Density of Peninsular Malaysia using Visual Flight Path Trajectories from Automatic Dependent Surveillance-Broadcast (ADS-B) Data". In: (2018). URL: https://www.sciencepubco.com/index.php/ijet/article/view/22238//// (visited on 04/25/2018).

[6] "Air Traffic Volume and Air Traffic Control Human Errors". In: (2011). URL: https://www.scirp.org/pdf/JTTs20110300005_65685953.pdf//// (visited on 05/22/2011).

[7] *Raspberry Pi ADS-B Base Station for OpenSky Network*. 2017. URL: https://github.com/openskynetwork/raspberry-pi-adsb//// (visited on 09/16/2017).

[8] *Introduction to ADS-B*. 2016. URL: https://www.trig-avionics.com/knowledge-bank/ads-b/introduction-to-ads-b// (visited on 09/10/2019).

[9] *Automatic Dependent Surveillance Broadcast (ADS-B)*. 2019. URL: https://www.skybrary.aero/index.php/Automatic_Dependent_Surveillance_Broadcast_(ADS-B)// (visited on 10/09/2019).

[10] *What are the ADS-B rules*. 2019. URL: https://www.faa.gov/nextgen/equipadsb/resources/faq//// (visited on 09/25/2019).

[11] *AUTOMATIC DEPENDENT SURVEILLANCE BROADCAST (ADS-B) IMPLEMENTATION IN INDONESIA*. 2017. URL: https://ops.group/blog/wp-content/uploads/2018/02/indon_1817.pdf//// (visited on 05/25/2017).

[12] *How ADS-B works*. 2015. URL: http://www.airservicesaustralia.com/projects/ads-b/how-ads-b-works/// (visited on 09/18/2015).

[13] *FAA Announces Automatic Dependent Surveillance-Broadcast Architecture*. 2012. URL: https://web.archive.org/web/20121022201516/http://www.faa.gov/news/press_releases/news_story.cfm?newsId=5520&print=go//// (visited on 10/22/2012).

[14] *Socket Data and BST files*. 2019. URL: http://woodair.net/sbs/article/barebones42_socket_data.htm//// (visited on 10/11/2019).

[15] *dump1090 stream parser*. 2018. URL: https://github.com/yanofsky/dump1090-stream-parser//// (visited on 10/23/2017).

[16] "Section 3. Wake Turbulence". In: (2009). URL: https://web.archive.org/web/20090905114743/http://www.faa.gov/air_traffic/publications/ATpubs/AIM/Chap7/aim0703.html//// (visited on 09/05/2009).