Digitized management of flight data

Joakim Ingnäs
Mikael Söderberg
Nicole Tutsch
Conrad Åslund
Abstract

Digitized management of flight data

Joakim Ingnäs
Mikael Söderberg
Nicole Tutsch
Conrad Åslund

Digital flight logs were created through the use of an already existing application called Open Glider Network. Open Glider Network’s system allows collection of data from airplanes that have FLARM installed. FLARM, which is an acronym for Flight Alarm, is the current safety precaution used to prevent collision between smaller aircraft. In addition, it also registers measurements such as the aircraft’s speed, altitude, and position. This project focuses on the use of collected data from the FLARM system to create digitized flight logs for Stockholms Segelflygklubb, made available through a website. This will allow them to compare the system with the analog one to ensure the safety aspects are the same. By doing this, they can minimize the human error concerning the logs when using the system.

Extern handledare: Jeremy Hamill-Keays, Ordförande för Stockholms Segelflygklubb
Handledare: Virginia Grande Castro och Björn Victor
Examinator: Björn Victor
Sammanfattning

Contents

1 Introduction ........................................... 1

2 Background ........................................... 1
   2.1 Gliding ............................................ 1
   2.2 Flight logs ....................................... 2
   2.3 The FLARM technology ........................... 2
   2.4 Open Glider Network ............................... 2
   2.5 Django ............................................. 3
   2.6 PhpMyAdmin ........................................ 3
   2.7 Stockholm Segelflygklubb ......................... 3

3 Purpose, aims, and motivation ...................... 3

4 Related work .......................................... 4
   4.1 Open Glider Network ............................... 4
   4.2 Flight Log .......................................... 5

5 Method ................................................. 5
   5.1 Database Design ................................... 5
   5.2 Choice of programming language ................. 5
   5.3 Graphical User Interface construction ............ 6

6 System structure ...................................... 6
   6.1 Open Glider Network ............................... 8
   6.2 Server back end ................................... 8
   6.3 Database .......................................... 8
6.4 User ......................................................... 8

7 Requirements and evaluation methods ................................. 9
   7.1 Database ............................................... 10
   7.2 Packet processing ...................................... 10
   7.3 Web interface ........................................... 10

8 Implementation .................................................. 11
   8.1 Connecting to OGN’s servers ............................. 11
   8.2 The OGN Automatic Packet Report System protocol ........ 11
   8.3 Database interface ....................................... 13

9 System description ............................................... 13
   9.1 Viewing and downloading the stored logs ................... 13
   9.2 Editing stored logs ....................................... 16
   9.3 Adding and editing objects ................................ 18

10 Error handling .................................................. 18
   10.1 Database crash .......................................... 18
   10.2 Error with OGN ......................................... 19

11 Evaluation results ............................................... 19

12 Results and discussion .......................................... 19

13 Conclusions .................................................... 21

14 Future work .................................................... 21
   14.1 Detecting and handling system crashes .................... 22
14.2 Detecting connected planes ........................................... 22
14.3 Automatic pilot detection ............................................. 22
14.4 Updated user login .................................................. 23
14.5 Handling false starts ................................................. 23
14.6 Improved landing detection ......................................... 23
1 Introduction

The concept of flying is one that has intrigued humanity for centuries, with the Wright brothers – according to academic consensus – to be one of the first who actually managed to take it to the sky, and today more and more people take to the skies every day; be it in small aircraft or commercial flights. However, there are those who perceive flying to be as dangerous as it is alluring. To ensure the safety of the pilot and any passengers, flight logs are kept [1] [2] when flying smaller planes or gliders. They serve as proof that services and checkups are performed within the recommended intervals. This is done in a way that these logs contain information such as the total time of the flight, which aircraft was used, and who piloted the aircraft, as well as the dates for performed checkups. This makes it easier to get an overview of how many hours the aircraft has been flying since the last service or checkup. In addition, it also allows the pilot to log any anomalies. The flight logs of such aircraft are kept the same way as the very first ones created: they are written manually by the pilot on paper.

Regulations make it difficult to fully digitize the logs, as it is required that the start times are logged on paper [3]. This could be because the security of a digitized system has never been properly tested. However, by keeping only hand written logs, several issues present themselves, both for the pilots and the flight clubs. Human error is impossible to avoid in situations like these, and therefore the flight logs can contain numbers that have been merely approximated. By using a digitized system, the logs could contain more exact data. Even if one of the physical logs would disappear, the data on the computer would remain and allow the log to be remade without major difficulties.

2 Background

The system consists of several parts, from the inner workings of the system itself to its external stakeholder. By understanding the individual parts the system will be easier to understand and less confusing for readers not familiar with the subject.

2.1 Gliding

Gliding is a form of aviation, where one uses an aircraft that is not powered by its own engine, or has a very light engine. When practicing the sport, the goal is to glide through the air for as long as possible. As most gliders do not have engines, they require to be launched. The most common way of launching a glider is by using a towing plane: a
2 Background

A small motor driven plane that pulls the glider to a height at which the pilot wants to start the flight. Another less common way is by winching the plane into the air [4]. The latter is not used at the stakeholder flight club, and was therefore not taken into account when creating the system.

2.2 Flight logs

The Swedish Transport Agency insists that when flying any type of aircraft, logs always have to be kept in a type of logbook. The format of how the logbook is written can differ. However, these logs have to contain the following information: pilot, aircraft identity, starting position, landing position, total flight time, the type of the flight, and notes of any deviations encountered [2]. As this applies to all types of aircraft, these logs have to be kept when gliding.

2.3 The FLARM technology

FLARM is an acronym for flight alarm. The FLARM technology is a traffic awareness and collision avoidance technology used in smaller airplanes, gliders, and helicopters. The FLARM technology is currently the only technology of this type used in smaller aircraft. The aircraft in question are equipped with what is referred to as a FLARM sensor, which transmits information about the aircraft, such as speed, direction, coordinates, altitude, and more. This will be further explained in section 8.2. The data is received through the FLARM receivers, which are positioned on the ground [5]. Each of the FLARM receivers and transmitters have a unique FLARM ID which identifies them, which allows to identify which aircraft or airfield they are on.

2.4 Open Glider Network

Open Glider Network, OGN, is an open source network for users and developers constructing software for the FLARM technology. Open Glider Network also connects FLARM receivers all over the world to make sure that all FLARM sensors’ data is accessible to the general public through their servers. OGN was founded to improve the reception range of the FLARM sensors in the gliders in an attempt to improve the safety in areas where glider traffic interacts with larger aircraft [6].
2.5 Django

Django is a high-level Python web framework that uses a MCV (Model-View-Controller), but it behaves differently compared to regular frameworks. In Django, views control what data should be displayed and templates replace controllers, since templates render HTML dynamically. This means that Django’s MCV becomes MTV (Model-Templates-View) instead [7]. Django also have built-in ORM (Object Relational Mapper) that makes it possible to write database queries in Python instead of SQL statements. In addition, the web frameworks also uses database migration that helps the developer to propagate changes in models to the database schema, and a built-in admin interface that helps the developer to view and change database tables [8].

2.6 PhpMyAdmin

PhpMyAdmin is a software tool written in PHP used to handle the administration of MySQL over the web [9]. It is used to simplify management of a database by using an intuitive web interface and having support for most MySQL features. Frequently used operations can simply be performed through the user interface. It is still possible to execute direct SQL queries.

2.7 Stockholm Segelflygklubb

The external stakeholder for this project is a glider club called Stockholm Segelflygklubb (Stockholm Glider Club), SSFK. It is an organization which during gliding seasons (middle of April to the middle of October), has daily operations of gliding. The organization was founded in 1935 and has been operational ever since. They currently operate at Långtora airfield, which is located between Uppsala and Enköping [10]. While that the glider club only works with gliders, the project is not limited to only this group, and can be applied to other aircraft containing FLARM sensors, such as helicopters.

3 Purpose, aims, and motivation

The focus of the project laid on collecting and storing data from the airplanes by using the FLARM sensors: a specialized electronic system installed in the planes. By using the FLARM sensors, the goal is that the system should be able to generate, log, and save finished flight logs. This minimizes the risk of the human error interfering with the data,
as well as ensuring that correct values are logged. With a digitized logging system, the safety of digital flight logs can be reviewed to see if there is a possibility to integrate them into regular flight clubs.

Because the logs are always written by hand, there is a constant risk of a manually written flight log to disappear, without any backup, which makes it impossible to recover the values recorded in it. If a digitized version of the flight logs were to be implemented, the risk of losing valuable data would be greatly reduced. If the primary digital log disappears, the backup of the values would still be accessible.

Since the logs get the data from the built-in electronics inside the airplanes, the recorded values are more accurate than what the pilots approximate when they write their logs. This would be of great help for the clubs, as billing would be more correct. Not only that, but the logs are generated during the flight itself, creating a fully working flight log without needing any external input.

This system is deemed a better solution than other systems on the market due to the fact that the solutions are developed too slowly and irregularly as well as are not focusing on an user perspective. By keeping the user interface simple and easy to use as well as making logs easy to access and downloadable for the regular user, the system will be better suited for the public than similar systems on the market.

4 Related work

There are few works that are similar to this project. One is an open source network that is used in this project, but unfortunately it only gathers the data from the FLARM technology and does not create any actual flight logs. A fully automatic logging system does not exist, but rather a system that merely temporarily saves the information for the pilots to write down.

4.1 Open Glider Network

The OGN, explained in section 2.4, is closely related to this project as it is an open source network which already has some minor operations that can be useful for the project. For example, extracting the text packets from receivers is made easy when using the already existing code from OGN’s website. What OGN lacks is an automatized method to store and update the information that is collected.
5 Method

4.2 Flight Log

A simple flight logging system called Flight Log exists that uses the OGN servers to create a simple log for a specific airfield. The problem with this system is that it is not designed to be used as a dedicated log system, and it is not possible to save the data. The system that exists gives a simple overview of the flights from and to the specific airfield and can not log the flight if the airplane would land on a different airfield.

Other problems with this system are that it is not possible to alter the data in any way, such as adding the pilot’s name. It also has a limited user interface, which restricts the possibilities of editing when creating logs. The existing system is not developed to be used as a substitute to the existing flight logs [11].

5 Method

When creating a large project, several methods can be used to solve the problem. To understand how the problem was solved, the methods are vital pieces of information. This section will describe how the chosen methods were used. However, how they were implemented is explained in section 8.

5.1 Database Design

When the database was designed, an original flight log on paper was analyzed. The different types of information that were wanted were then summarized, and first designed into a database in first normal form (1NF), where all the cells of the database only contain one value. By applying normalization to the database in 1NF, the database reached the goal of being in Boyce-Codd normal form. This was done to avoid any unnecessary information retrieval when accessing an entity of a database. By doing so, time will be saved when performing multiple calls using the database.

5.2 Choice of programming language

We needed a programming language that worked on different platforms, to ensure that the system can be run on the platform the user has available. There are multiple programming languages which fits this requirement, and therefore pros and cons for different languages and libraries had to be compared. Multiple languages were good can-
didates, but since everyone in the group were supposed to know the programming language, Python was chosen. This language was chosen over other languages such as Haskell because it is a language that everyone developing the system have previous experience with, and it also fulfills the requirement. Python also enforces a stricter indentation in the code and therefore forcing a code standard which makes code easier to get a better understanding of, and might have resulted in a quicker development process. There was no pressing need to use another programming language, since Python did have all the functionalities needed, and other languages were less desirable compared to Python. This was mostly due to the fact that web frameworks such as Django are compatible with Python, and the project would therefore benefit from the use of Python.

5.3 Graphical User Interface construction

The design was implemented by using a web framework called Django that is written in Python. Django is helpful when there is a wish to create complex database-driven websites, and was a natural choice since the rest of the system was mainly written in Python. Django compared to other frameworks is said to be easy to learn and use. In addition it also has a thorough documentation. It is also used by larger companies, such as Instagram [12], for different back end features.

To ensure that the website would adapt to different screen sizes, Cascading Style Sheets (CSS) and JavaScript (JS) were used. Django needs to be structured in different applications, or parts, which meant three different applications had to be developed: personal, news, and log. Personal contains the header for the website, homepage and some basic HTML code, News is an application made for news posts, and Log is the main logging application that shows the data that is logged during flights.

6 System structure

The system uses OGN’s already existing network as seen in red in Figure 1. OGN’s network is used to extract live data from the aircraft. The user can view stored information via a website. The server communicates with the FLARM system, stores the data in the database and helps the website to show the user the correct data.
Figure 1: Block diagram of the general system structure and its connections. The arrows indicate one-way data transmissions, red is external parts of the system, green our own parts of the system.
6 System structure

6.1 Open Glider Network

Collecting data is done through two of OGN’s servers. After obtaining the data, it is sorted by the FLARM ID of the airplanes owned by the flight club. The data sent by the FLARM sensors contains more variables than what is required for flight logs. To find what data is needed, already existing paper logs have been examined. The data is extracted, presenting the user with a complete and correct digitized flight log.

6.2 Server back end

The back end of the server handles the communication between the other three parts of the system: the OGN network, the client side of the system and the database. The server’s primary directive is to collect data from the OGN network, process it and store it in the database continuously during the flight.

The other directive of the server is to make sure any clients requesting logs from the website should be able to retrieve the logs. The only clients who would be able to access the logs of their own aircraft are those who already have already registered their club and aircraft, as the system does not save all logs of every aircraft but only registered aircraft of the flight clubs.

6.3 Database

A database is required to save the data, from which it later can be viewed, when it is needed. The database is used to implement the actual logs and displaying them on the website.

In this particular case, a simple Excel document would suffice. However, to make it more dynamic and sturdy, a database combined with a graphical user interface on a website is used. This also makes the database easier to access from different machines.

6.4 User

As the stakeholder in the project prefers to display the collected logs as an Excel like structure, it will be similar when displayed in the graphical user interface. The user will be directly interacting with a website, in which it can check the flight logs of its
registered flight clubs. This will provide the user with an easily accessible way to view the logs it wants to check out from different machines.

7 Requirements and evaluation methods

The most important requirement from Stockholms Segelflygklubb is that the system should log the data with the same accuracy as the pilots normally do, preferably with a higher accuracy rate than a person. It needs to be as reliable as the analog system, and also secure. The system should be crash-proof and unable to be hacked by outside sources.

In addition, regular users should be able to edit their own flights, while an administrator should be able to edit every entry.

If the system creates a larger workload for the employees of the flight club than the regular analog system or the data the digitized logging system logs varies too much from what the pilots have written, then the digital system will be deemed a failure.

The system should be able to process manually inserted information, for example the pilot ID. The system output should be clear enough for the user to understand, even if they do not have a technological background.

Evaluation of the system is easy as it can be compared to the analog system on a regular day at the flight club, and the person who compares it can investigate if it logs the same data, or at least within sensible marginals, by manually comparing the analog log to the digital log. Marginals will only apply to the take-off- and landing time, as well as the altitude of the towing plane. Landing time- and take-off time are allowed to differ with \( \pm 5 \) minutes, and the altitude of the towing plane might differ \( \pm 10 \) meters. If the FLARM ID is different than what the pilots wrote down, the system will be immediately deemed a failure. These marginals previously discussed were created in corporation with the flight club were created through a dialogue with the flight club, to ensure they would get data they deemed useful.

To summarise, the system is marked as successful if it can collect the information, transfer the flight logs to the storage database as well as contain information that is as correct as possible.
7 Requirements and evaluation methods

7.1 Database

The database must save the information taken from the FLARM system. When the information is saved as logs, they can be accessed in case the original log disappears. Currently, the logs should contain information of the entire flying season, which is around seven months. The database should allow users to edit different information cells, depending on the users’ clearance level.

There are two types of clearance levels. The first one is an user clearance level which gives any flight club the permission to read and write in the database, and this level of clearance is their standard one. This level allows the clubs to view and edit data such as flight logs, aircrafts, and pilots. This data is important for the logs so they can be created in the first place. The other clearance level is the one for the admins. An admin can access all kinds of data. They have access to and can edit things like flight logs, aircrafts, pilots, and regular users.

7.2 Packet processing

The packet processing has to determine what is seen as a relevant packet and gather information, which is sent in a plain text format as a string. The packets themselves will be described in 8.2. The processing also has to be able to determine which stored information is relevant to update whenever a new packet is received, for example maximum altitude. Aside of that, it should determine which row in the database that should be updated with the new information. For the system to be considered stable, the code that performs the processing also needs to be able to handle packets containing corrupt data, should any information diverge from the specification of the protocol.

7.3 Web interface

The website is designed so the user can easily choose the preferred date and view the log on the website. It is also possible for the user to download the logs sorted by date in XLS format (2003 Excel format). Beyond that, the website also has news that the can be added, depending on the users’ clearance level. As mentioned before, the clearance level only has two levels: administrator and user. To test the websites usability, random selected individuals, both those that worked at the flight club, and those that were there to fly, were asked to try the website and see if they could navigate it. They were asked to find the flight logs, and different parts of the website, such as news and about us. Some problems arose during the testing with finding the log part of the website, but this
was resolved afterwards, since no other user seemed to encounter the problem after the patch.

8 Implementation

In section 5, the different methods used in this project were explained. This section will describe how the methods were implemented in the project.

8.1 Connecting to OGN’s servers

To start an active connection with the OGN servers, a string message consisting of user, password, program name and version, and server filter is sent from the client. By sending a (-1) as password the server allows a listening connection where the client can only get messages, but is unable to send anything to the server. The server side filter field defines an origin spot described in longitude and latitude coordinates and a radius around that spot. This causes the server to send out packets to the client which originate from the aircraft that has been detected inside the defined area.

8.2 The OGN Automatic Packet Report System protocol

The Automatic Packet Reporting System (APRS), used for transmitting and receiving data within OGN’s network, is text based and uses compact representation to be effective.

Data is usually sent in multiple packets at a time, which the first packet is information about which receiver the data is coming from. The following packets, excluding the last packet, are the data transmitted from the aircraft. The final packet is a keep alive line for the APRS server to keep transmitting and receiving data in intervals. The keep alive packet is not transmitted as often as the other lines as it is only needed every 30 minutes [13].

A generalized example of how a packet line looks like is:

[Origin][Source][Time stamp][Position][Heading & Speed][Altitude][Position precision enhancement][Sender address][Climb rate & Turn rate][Signal to noise ratio, Error rate, Frequency offset, GPS Signal quality, Horizontal accuracy & Vertical accuracy][FLARM software & hardware information]
There is a lot of information in each packet, where the most important parts of the packets of making a flight logging system will be explained below.

- **Origin**
  The origin is the senders’ or receivers’ call sign. This is a simplified way of keeping track who sent the packet.

- **Timestamp**
  The timestamp tells the system what time the message was sent, in the Coordinated Universal Time (UTC). It is in 24 hour notation on the form HHMMSSh where HH is the hours, MM is the minutes and SS is the seconds. For example, 22 hours, 1 minute, 32 seconds would be written as 220132h.

- **Position**
  The position is given in latitude and longitude, dividing them both with a slash symbol. The latitude is given in either east or west and the longitude as north or south. An example of how a position would look like is 4658.70N/00707.72E, which would be read as 46 degrees, 58.70 minutes north and 007 degrees, 7.72 minutes east.

- **Heading & Speed**
  This field of information may be left blank, for example if it is a receiver sending the packet, as a receiver would have no heading (direction) or speed. If so happens, "000/000" in this field indicates no data, which is the case if an aircraft is stationary, or the unit which sent the packet is a receiver. If the information is given, it would be given on the form "abc/xyz" where abc is the heading in degrees, where a "090" would correspond to heading east. The speed xyz is given as the ground speed in knots.

- **Altitude**
  The altitude is given as A=XXXXXXXX where the number XXXXXX is the altitude in feet.

- **Sender address**
  The sender address is on the form idXXYYYYYY where XX is a 2-digit hexadecimal number encoding 8 bits of sender details, such as one bit for stealth mode, one bit for no tracking mode, four bits for sender type and two bits as address type. The YYYYYY which follows is the sender address, the ID which is unique for every sender and receiver.

- **Climb rate & Turn rate**
  Firstly, the climb rate is given as ±nnnfpm where ”nnn” is the climb rate in feet
per minute. The climb rate is then followed by the turn rate, given on the form ± x.xrot where x.x is the turn rate in halves per minute [14, 15].

8.3 Database interface

The database interface used to implement the designed database was phpMyAdmin. This was used to ease the manipulation of both database structure and database data during the testing phase. By using phpMyAdmin, editing single elements of data in objects was made simple. Had another interface been used, for example MySQL Workbench, the user would instead have needed to run a query. By using phpMyAdmin it instead becomes a simple task of clicking an object and changing one element.

9 System description

The system consists of multiple parts, which are all necessary for the system to function properly. These parts together create the back- and front end of the system. The key parts for full functionality of the system will be described both front and back-end wise.

9.1 Viewing and downloading the stored logs

Viewing the stored logs is done through the Django built user interface. To do so, the user connects to the website which hosts the user interface. After doing so, the user can press one of two buttons on the website marked in red in Figure 2 to view the logs that the system has created. A second button to get to the flight logs was implemented after testing showed that some users thought that one of the buttons was just a logo. This was a quick solution, and should the system be upgraded, a better solution would be desirable.

Once one of those two buttons has been pressed, the user will be redirected to another page, which will show them the flight logs for the day’s date. The user will also have the choice to show logs for a certain date, by entering the desired date in the blue box and pressing the red button as shown in Figure 3. The user can also download logs for a certain date, which is done by entering the desired date to download for in the green box and pressing the orange button also shown in Figure 3.
Figure 2: Rotated image of the start page for the web based user interface with the two buttons to access the flight logs highlighted in red and the button to login highlighted in blue.
Figure 3: Rotated image of the flight log page of the user interface, with date choosers marked in green and blue and buttons for approving the date marked in red and yellow.
9.2 Editing stored logs

Not every user can edit the logs stored by the system. To do so, the user has to be of at least user level in Django’s admin interface. To access it, the user simply presses the login button on the start page, marked blue in figure 2. This redirects the user to Django’s admin login page, to which the user enters its login information to access the admin user interface. Once logged in, the user can then choose to press the change button for the Flight datas object, marked red in figure 5. This will bring the user to a page to choose which flight log to edit. Once the user has chosen a specific log to edit, it will bring the user to the edit page for that specific log, which displays all information and allows the user to change the information it wants to, as seen in figure 4.

Figure 4: Image of Django’s admin interface with an active instance to edit a flight log.
Figure 5: Rotated image of Django’s admin interface in the object menu, where buttons to add objects are marked in blue, buttons to edit objects are marked in yellow and the button to edit flight logs marked in red.
To add and edit objects, the user still needs to be logged onto Django’s admin interface. To then choose which type of object the user wants to add, one of the buttons highlighted in blue in Figure 5 should be used. To choose which type of object the user wants to edit, one of the buttons highlighted in yellow in Figure 5 should be used. Once one of these has been chosen, the user will either be redirected to create a new instance of the object or choose an existing object to edit. If the user choses to edit an existing object, after choosing which object to edit the user will be redirected to the edit page for that specific object. It will look similar to the page seen in Figure 4, but showing the data for that specific object. If the user instead choses to create a new object. It will be redirected for a custom creation page for that specific object. An example of the creation page for gliders is shown in Figure 6, where the data needed for the object is prompted by the user.

Figure 6: Image of Django’s admin interface when choosing to add a new object of the type glider.

10 Error handling

The system is programmed to withstand most of the different errors and crashes that can occur. Should the system crash, it will most likely be because of hardware-related problems, such as the computer containing the database being damaged.

10.1 Database crash

Currently, the system cannot handle the error of a crashed database. Should the system lose the connection to the database, it will stop running. This could be fixed by making
the system try to reconnect to the database continuously until it manages to establish a connection once again.

10.2 Error with OGN

Should OGN become unavailable for some unknown reason, it would be impossible for the system to function as it should, since it is from OGN the system acquires the data to create flight logs. Unfortunately, this would not be possible for the system to fix as OGN is a separate system.

11 Evaluation results

The system was properly tested on a regular day at the flight club. The results from the system were manually compared to the analog results the flight club kept, together with notes taken by the creators of the project. This ensured that there would be multiple points to reference when evaluating the results.

From the results gathered, the project should not create a larger workload for the personnel at the flight club, as they do not have to do more than input the pilot ID.

The format of the display of the data resembles an Excel sheet, or rather a database table, which was a requirement from the stakeholder of the project.

12 Results and discussion

The system gathered information from the FLARM system, and logged it with only minor flaws, one which was the way it detects which tow plane tows which glider. The recorded take-off and landing times were the same as what the flight club had written. The altitudes were the same as well, with a more precise value than the analog log. This is because the analog uses the altitude as a way of billing, since the flight clubs have set their prices depending on the altitude the glider gets towed.

The system manages to achieve the goals set in section 3, as it extracts the data and saves it to a database that can act as a backup. The finished log was considered as easy to read by both the personnel at the flight club and pilots, as they did not encounter any problems with the website. The output can be seen in Figure 7, Figure 8 and Figure 19.
12 Results and discussion

Figure 7: First part of the output in the database, showing which flight it is, the date it was logged, the name of the pilot, ID of the glider plane, what kind of flight it was, and the take-off time.

<table>
<thead>
<tr>
<th>Flight No</th>
<th>Logged Date</th>
<th>Glider Pilot</th>
<th>Glider Id</th>
<th>Flight Type</th>
<th>Takeoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>05/14/2017</td>
<td>—</td>
<td>UGI</td>
<td>—</td>
<td>12:16</td>
</tr>
<tr>
<td>63</td>
<td>05/14/2017</td>
<td>—</td>
<td>USG</td>
<td>—</td>
<td>12:31</td>
</tr>
<tr>
<td>64</td>
<td>05/14/2017</td>
<td>JockTori</td>
<td>UPZ</td>
<td>F</td>
<td>13:03</td>
</tr>
<tr>
<td>65</td>
<td>05/14/2017</td>
<td>—</td>
<td>UGI</td>
<td>—</td>
<td>13:45</td>
</tr>
</tbody>
</table>

Figure 8: Second part of the output in the database, showing the ID of the towing plane, the pilot of the towing plane, the time the towing plane landed, the time the glider landed, what height the glider was towed to, and how long it took to tow.

<table>
<thead>
<tr>
<th>Towing Id</th>
<th>Towing Pilot</th>
<th>Towing Landing</th>
<th>Glider Landing</th>
<th>Towing Height</th>
<th>Towing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSB</td>
<td>—</td>
<td>12:24</td>
<td>13:25</td>
<td>618</td>
<td>00:07</td>
</tr>
<tr>
<td>CSB</td>
<td>—</td>
<td>12:42</td>
<td>12:52</td>
<td>904</td>
<td>00:10</td>
</tr>
<tr>
<td>CSB</td>
<td>—</td>
<td>13:17</td>
<td>13:40</td>
<td>1206</td>
<td>00:14</td>
</tr>
<tr>
<td>CSB</td>
<td>—</td>
<td>13:52</td>
<td>—</td>
<td>614</td>
<td>00:07</td>
</tr>
</tbody>
</table>

9. In this report, the output here has been broken up into three parts, to make it more readable in this report.

Some values are shown as “—” because they have yet to log any data. Some columns, such as the notes, pilots and flight type, are “—” since the flight club need to manually enter the data. It is impossible to acquire that data from the FLARM system. Other columns, such as glider landing and flight time, are “—” because the glider has yet to land. There is therefore no data to show, and that is why it shows “—”.

20
14 Future work

Figure 9: Third part of the output on the website, showing how long the flight lasted, if the flight is ongoing or finished, if there are any notes about the flight, and the maximum height the glider reached.

<table>
<thead>
<tr>
<th>Flight Time</th>
<th>Flight Status</th>
<th>Notes</th>
<th>Max Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:08</td>
<td>Finished</td>
<td>—</td>
<td>743</td>
</tr>
<tr>
<td>00:21</td>
<td>Finished</td>
<td>—</td>
<td>908</td>
</tr>
<tr>
<td>00:37</td>
<td>Finished</td>
<td>—</td>
<td>1222</td>
</tr>
<tr>
<td>—</td>
<td>Ongoing</td>
<td>—</td>
<td>619</td>
</tr>
</tbody>
</table>

13 Conclusions

While there were some minor bugs found on the day of the testing, such as receiving the wrong altitude, the results overall were a success, with minor deviations from the analog flight logs.

The project itself will benefit the large amount of flight clubs around the world as a stepping stone towards digitizing flight logs. It is the first system that creates the actual log to be saved, by using the data from the planes themselves. Related systems do not have the capability to save the logs, but rather merely temporarily saves the data so the pilot can write it down. As such, this project is a large step towards a digitized future.

14 Future work

While the system is complete by the defined functionality it was required to have, it is not perfect, and there are still several upgrades that could be implemented to improve the system.
14 Future work

14.1 Detecting and handling system crashes

There is currently no detection and handling of the system crashing. This could cause an issue if the system was to crash. As a countermeasure, the risks of a crash occurring are minimized by building the system to handle close to all software errors that could occur. This still means that a hardware issue could cause a crash of the system, and if so were to happen, the detection and handling of this crash would be insufficient. By implementing further handling of these type of crashes, the system would become even more robust.

14.2 Detecting connected planes

The current way of detecting which tow plane is towing a glider or which glider is being towed by which tow plane is by using time intervals and checking which aircraft take-off together. During the testing on the airfield, some minor issues were found with this: if some FLARM ID:s of aircraft were missing, and take-offs were very close together, it would cause the logs to mismatch the planes with each other. On a normal day of the airfield, during the circumstances that all planes FLARM ID:s are in the system, this would not happen. However, if there are any events, such as a championship, this could turn into an issue when take-offs are very close to each other. To fix this issue, a position based algorithm could be used instead. Utilizing the coordinates which are sent with the APRS packets, one could set up a radius around all aircraft taking-off and decide if two aircraft are overlapping while take-off to match them as the same flight.

14.3 Automatic pilot detection

An improvement which could be made to further ease the workload of the flight clubs would be to implement an automatic pilot detecting system. This would, except for the software changes required, also require a type of new hardware to be used. In theory, any type of contactless object, for example a card, could be used. A reader would be installed in each aircraft, which the pilot would then scan their card with to give the system the information regarding who is about to fly the aircraft. This would further automate the entire system. Because of right now, the flight clubs themselves must manually edit the pilots of the flight using Django’s admin interface.
14.4 Updated user login

The current user interface has the functionality that is needed. One can display the logs, download them and if a user has special permission they can also edit the logs, and add pilots and aircraft by using Django’s already existing admin page. This can be tedious, as when the admin page is entered and the user is not logged in, there is no link to get back to the original page. To solve this issue, custom functionality could be added to integrate this into the regular user interface instead of using Django’s admin page.

14.5 Handling false starts

A plausible issue that could occur due to certain algorithms is that when a glider is towed on the ground at a high enough speed, a take-off will be initiated. If this occurs, the system should be able to identify this as a false start and erase the initiated flight. This is not yet implemented as it is not such a frequently occurring issue and therefore not highly prioritized.

14.6 Improved landing detection

The algorithm for detecting when an aircraft is landing could be improved. Currently, an algorithm based on the aircraft’s speed is used. The concern with this is that aircrafts might not slow down fast enough for the system’s view on a landed aircraft to be the same as the flight leader at the airfield. The algorithm could be included to include different parameters such as height as well. This is left to be implemented into the future as this only offsets the system’s logged time by a maximum of roughly a minute compared to the analog logs. This could also be improved by internationally setting a general way of determining when a plane is to be considered as landed when taking different parameters into consideration, for example altitude and speed.

References


