introduction

For a school project we ware tased with building a system that can automatically register aircraft and more importantly helicopters that use Lelystad airport. One of the requirements of this project was that we can't physically change the aircraft. Sins building an additional radar system was way too expensive we decided to triangulate the location of the aircrafts using mlat and feed the aircraft locations in a series of microservices that can translate coordinates in to the events registered by the automated billing process.

In our first design of the mlat service we used 5 radarcapes (we got the package deal that included an antenna) and the mlat server code written by Jetvision, but cost of using the jetvison server ended up being greater than the amount our product owner wanted to spend on the project.

After this setback our attention shifted to this opensource project but we noticed quite early the setup of the mlat-server was not documented well. We collectively spend over 100 hours between the 4 of us, even dough we are second or forth year ict students. The aim of this document is to provide a comprehensible guide on how to set up your own mlat-system with 5 radarcapes as mode s recivers. At the time of writing this document our main scope is tracking helicopters but this product is mostly tested with aircraft, whit the word aircraft I mean both regular aircraft and helicopters.

I would like to thank wiedehopf who answered a lot of our questions, if only we would have contacted him earlier we would not have had de need to spend so much time. I realize there are probably a lot of thing I could have done better but at the end this worked for me. If you haven any idee how this guide could be better I would like to hear it.

Radarcape location

The first step of building this mlat-system is setting up the radarcape receivers. The receivers are not weather proof and need to be place inside, the ads-b and gps antenna however need to be place outside for archiving the best signal integrity possible. The ads-b antenna needs to be mounted upright for both performance reasons and making sure the antenna is weather proof. It is also important to note that the antenna mound needs to be mounted to a grounded pole, this is to protect the radarcape against electro static shock. All of our radarcapes are connected to the network via a cat 6 utp cable.

For the best accuracy of the whole system placing one antenna on the airport end placing the rest of the antennas in a 10km ring around the airport is optimal, this is the conclusion of report written by Andy van Helden who happens to be our product owner. I understand that the research he did was the basis of the radarcape development, this was the reason we chose the radarcape for this project.

The server

We tried to set up the mlat-server on both linux (Debian 10) and windows 10 whith linux subsystem, eventually I got it working on a 1 core 1gb ram vps that I got for free with my Microsoft azure student account. We found that the project would only work on python 3.5 with 3.5.10 being the most recent version, this Is not ideal but I don't have the python knowledge required to upgrade the project to a different version.

The first step is building python from source I did that by entering the following comments in the terminal, if you would like to have a better understanding of all the commands at the end of the document all the commands are explained.

```
sudo apt update
sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev
libnss3-dev libssl-dev libsqlite3-dev libreadline-dev libffi-dev curl
libbz2-dev git -y
sudo mkdir /usr/local/py-mlat
cd /usr/local/py-mlat
sudo wget https://www.python.org/ftp/python/3.5.10/Python-3.5.10.tar.xz
sudo tar xf Python-3.5.10.tar.xz
cd Python-3.5.10
./configure --enable-optimizations
sudo make altinstall
```

after installing python I realized that is was still missing the pip installer, I installed it by running:

```
sudo apt-get install python3-pip
```

After installing python it was time to get the git repo, I personally used the adsbxchange/mlat-server repo which I later forked. The following commands ware used the clone the git repo:

```
sudo git clone https://github.com/boaskaken/mlat-server.git
cd mlat-server
```

Before running the mlat server we need to make sure all the dependencies are met, using pip3 I installed all the necessary libraries:

```
Sudo pip3.5 install numpy scipy python-graph-core pykalman objgraph
uvloop
```

After installing all the necessary libraries the mlat-server is ready to be run, you can see the help by running:

```
sudo python3.5 mlat-server --help
```

In case you want to make the mlat-server a service I included a basic service file in the repo. For this it is required that you put the correct start up commant in the start_mlat-server.sh file, after this you can enable the service by doing the following:

```
sudo chmod +x start_mlat-server.sh
sudo chmod 644 mlat-server.service
sudo cp mlat-server.service /etc/systemd/system/
sudo systemctl start mlat-server
sudo systemlctl enable mlat-server
```

The radarcapes

Compared to the server the installation of the radarcapes is relatively simple we start of by changing the config of the radarcape, this is done via the web interface. Go to the “tdp data streaming settings” within the settings tap. Enable the port 10003 udp server, the receiving server should be

127.0.0.1:10003. After changing the settings don't forget to save by pressing change settings. After this we can start with installing some dependency's and cloning the git repo:

```
sudo apt-get install python3 wget jq python3-pip git
git clone https://github.com/boaskaken/mlat-client.git
```

installing the client itself is done by running;

```
./setup.py install
```

And again you can see the help message by running.

```
sudo python3 mlat-client --help
```

Making this process run as a service is done by using the same set of commands as with the server.

```
sudo chmod +x start_mlat-client.sh
sudo chmod 644 mlat-client.service
sudo cp mlat-client.service /etc/systemd/system/
sudo systemctl start mlat-client
sudo systemlctl enable mlat-client
```

Conclusion

These are all the steps we took to come to the result we achieved. imported to note here is the fact we run a mlat calculation on all the aircraft we receive, because most large aircraft send out their gps location this is not needed for most people. Running the calculation on all aircraft can cause a huge load increase in the server especially if you have many more recivers. You can revert back to using the original code by using the adsbxchange/mlat-server reapository.

explanation of the commands

below are the commands used in this installation manual, commands are only explained once.

`sudo apt update`

The sudo part of the command means that the rest of the command is executed with root privileges. Apt is the package manager used by Debian and is used to install and manage software from the command line. By giving apt the update parameter apt pulls all the available software and versions from the servers specified in the /etc/apt/sources.list. It is a good idea to run sudo apt upgrade after this command, this installs the newest version of all the installed software.

`sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libsqlite3-dev libreadline-dev libffi-dev curl libbz2-dev`

this command installs the software needed to complete the rest of the process, this is again done with the apt package manager. the names of the different software are listed with spaces to separate the different packages. Normally the package manager will ask if you agree with the amount of diskspace it will use to install all the packages, by adding -y to the command you automatically agree with the amount of diskspace that is used.

```
sudo mkdir /usr/local/py-mlat
```

This creates a new directory (basically a folder), the parent directory will be /usr/local. The name of the directory itself is py-mlat. The user I am using does not have write privileges for /usr/local so again sudo is used for the root privileges.

```
cd /usr/local/py-mlat
```

With cd you can change the directory you are working in, in this case to the newly created directory.

```
wget https://www.python.org/ftp/python/3.5.10/Python-3.5.10.tar.xz
```

Wget is used to download a file from the internet, this file is placed in the current directory.

```
sudo tar -xf Python-3.5.10.tar.xz
```

The downloaded file is a .tar.xz archive, this archive can be extracted with the tar command. The – is used to pass extra options to the software, in this case the xf means that tare is a file that needs to be extracted. Python-3.5.10.tar.xz is the name of the archive that needs to be extracted.

```
cd Python-3.8.2
```

Used to change in to the Python-3.8.2 directory that was created by tar.

```
./configure --enable-optimizations
```

The ./ command is used to run a piece of software or a script, in this case the configure script. This particular script is used to configure the way python is compiled in the next step, by passing the --enable-optimizations option python will be compiled with all the necessary addons needed for the mlat-server.

```
sudo make altinstall
```

This command compiles and installs python, the altinstall parameter is used to signal that it should not be installed as the systems main python3 configuration. This is an older version of python and it is better to use the most recent version for your main configuration.

```
sudo apt install python3-pip
```

```
git clone https://github.com/boaskaken/mlat-server.git
```

This clones my git reposotory, all the files to run the mlat-server are in here.

```
sudo chmod +x start_mlat-server.sh
```

This changes the permissions of the start_mlat-server.sh file so it can be executed.

```
sudo chmod 644 mlat-server.service
```

This changes the permissions of the mlat-server.service file so it can only be executed with root privileges.

```
sudo cp mlat-server.service /etc/systemd/system/
```

This copys the mlat-server.services file to the /etc/systemd/system/ location. The target location can only be written to with root privileges.

```
sudo systemctl start mlat-server
```

Systemctl is used to keep track of all the software that is actively running on the server, I made the server one of those actively running software. With this command the software starts up. If you would like to know if it is still working you can check by typing: sudo systemctl status mlat-server.

```
sudo systemlctl enable mlat-server
```

This makes sure that the malt-server is run on start-up of the system.