Carnegie Mellon University
Master of
Software Engineering

LG Software Architectures Training Program - Project Description

Intelligent Flight Tracking Assistant

Project Overview

The goal of this project is to improve a system that meets the objectives and requirements of the Federal Aviation Administration (FAA) issued request for proposal (RFP). The RFP scope includes using embedded machines (or computing on the edge) which manages a software defined radio to view local aircraft in the air transmitting ADS-B signals, publishes the aircraft data to a community hub, and connects to a Windows based graphical user interface (GUI). The goal of the system is to introduce air traffic control students to a basic tool that introduces different scenarios and situations that a flight controller will encounter.

The project sponsor, SolveIt Inc., wants to respond to the RFP with its new product line of these flight tracking kits and GUI application. Solvet Inc built a prototype or minimum viable product (MVP) and is seeking the assistance from LG to make the prototype / MVP better. LG has committed several teams to support this effort. The LG software architecture class will improve the software architecture, using a class project with facilitation by Solvit Inc personnel.

SolveIt Inc. will select the best architecture/solution based on the results of this competition among the teams in this course. Key motivators for this project are:

- Ensure resilient and reliable operation of the Raspberry Pi in sending ADS-B data
- Scale ADS-B data by subscribing to community ADS-B information
- Provide the ability to easily extend the features/behaviors of the ADS-B system by users or third parties
- Provide the ability to easily add or replace algorithms that can predict flight information, such as closest point of approach (CPA)
- Provide a remote interactive / graphical user interface for interacting with local and community ADS-B live feeds or recorded feeds.
- Minimize time during the operation (e.g., capture ADS-B data, plot the tracks, and conduct calculations to add features)

In this project, each team will create a software architecture and build / improve the software for a Raspberry Pi to manage an ADS-B software defined radio (SDR), a Windows-based user interface application to interface with aircraft data and maps, and an external data source to enrich the data available for aviation related information.

Once the SDR and Raspberry Pi is turned on and connected to a network, a user may remotely access the Raspberry Pi to initiate the 1090 dump program to receive raw aircraft data stream with position, identity, velocity from the SDR and antennae connected to the Raspberry Pi.

The user interface may be initiated before or after the Raspberry Pi has begun its receiving of aircraft data. The user interface application is an interactive window with map data and several buttons and text boxes. From the user interface, the user may connect to the Raspberry Pi for live aircraft data, or the user may playback back prerecorded aircraft feeds. Further, the user may choose to look at local tracks only (for example, only tracks within a 50 mile radius based upon the limitations of the SDR and antennae), or the user may view a community hub's tracks which may display upwards of 5000 active aircraft globally.

While the user interface is receiving tracks, unless in a recording mode, the data is overwritten multiple times a second as position updates. If in a recording mode using the community hub data, recording SBS data (community ADS-B information) requires about ~15 mb of storage per minute. The length of recording will easily exhaust computing and storage capability locally on both the Raspberry Pi and student computers. Therefore, teams are provided an external database (or data warehouse) to support additional features and analytics for the user interface.

All teams will interact with and communicate their client related questions or concerns with a SolveIt point of contact. There are three primary Solvit Inc staff, and they will be introduced during the project kickoff meeting.

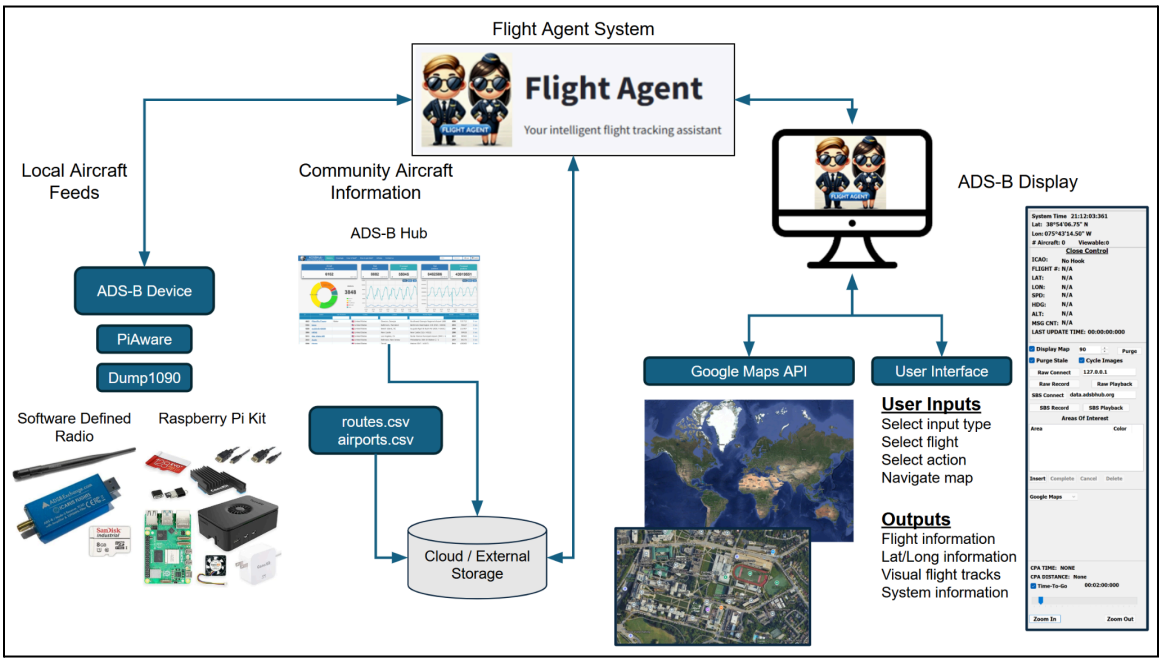


Figure 1: System Level Overview

System Description:

The goal of this project is to design and implement a software system for an embedded system and software defined radio that provides the following functionality.

Remote User Interface (RUI)

An example of the provided base line RUI is shown in Figure 2 below.

This remote user interface shall include communications software required to communicate with a Raspberry Pi Flight Tracker (see next section) via a wireless network. Specifically, the RUI contains a “Raw Connect” button and a field to input an IP address of the Raspberry Pi on the wifi network for the RUI to receive ADS-B data from the Raspberry Pi Flight Tracker.

The RUI loads map tiles that enable the system to plot aircraft positions based on raw ADS-B data (or data from the Raspberry Flight Tracker), recorded raw data, community ADS-B hub data provided as SBS data, or recorded SBS data. SBS data refers to the data message format used by ADS-B Hub to describe aircraft position and status information. If using SBS data, the user interface is simply a client to an external server, in our use case the server or database is managed by ADSBHub.org. SBS is thought of to stand for “Software Base Station”. Student teams will be required to set up an account and publish the data their ADS-B receiver is collecting in order to subscribe to the ADS-B Hub.

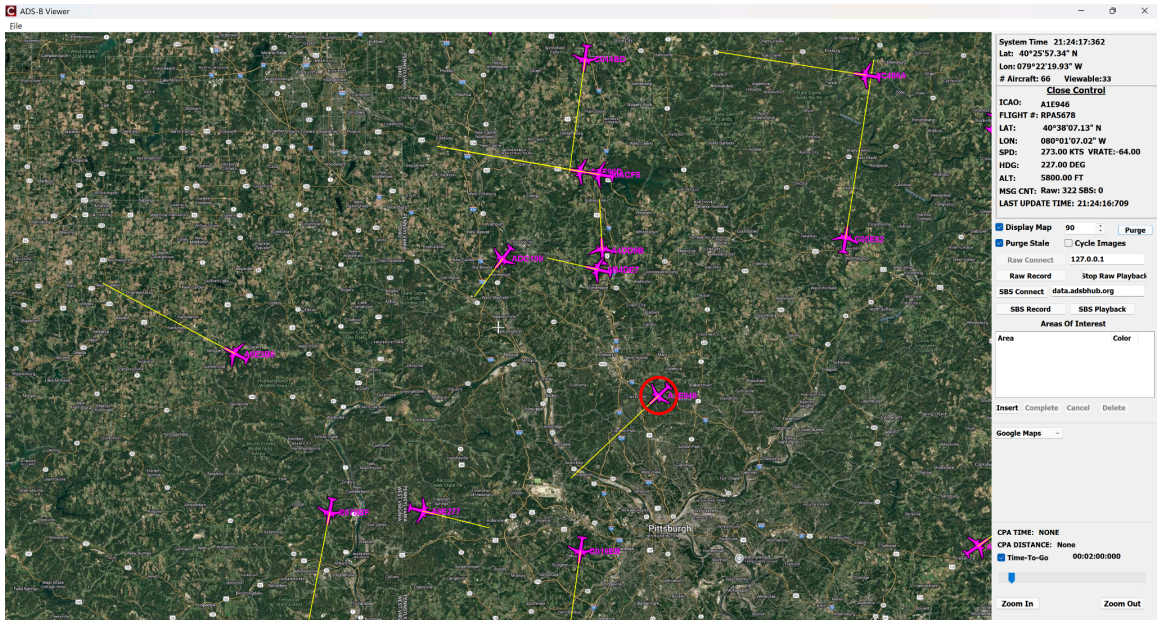
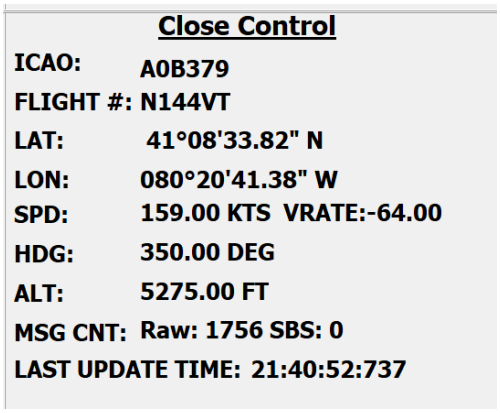


Figure 2: Example of Remote User Interface

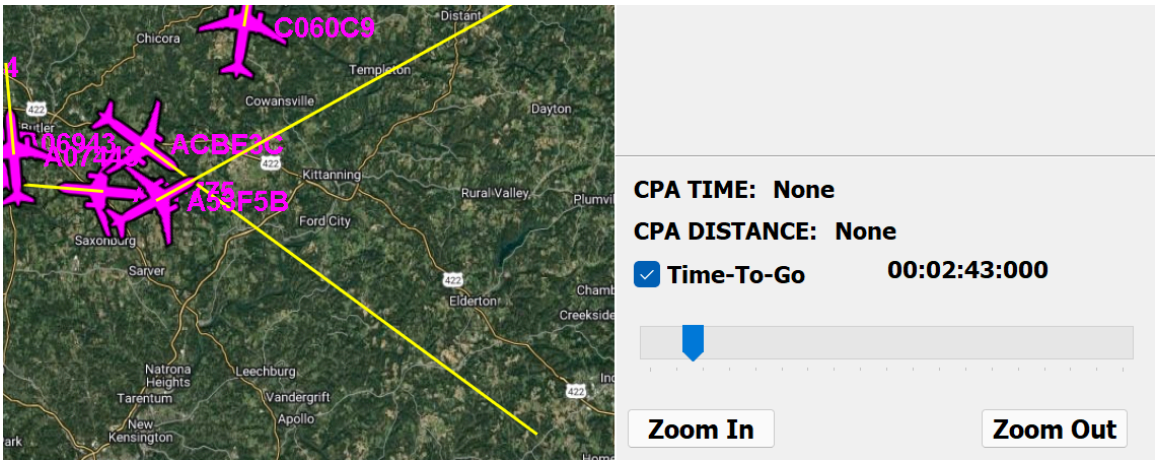
Of note, live data is overwritten in the current application when viewing in "Raw" mode. For example, new positions overwrite the previous position.
The user interface also provides an option to record the data where the data is recorded to an external file. Each new position information is appended in the data file. An additional option, with some code written by the students, is to stream the data to BigQuery - this assumes the students have a table set up in BigQuery with a proper table schema (see BigQuery section).

When the RUI displays an aircraft, right clicking on the aircraft image is called hooking. Hooking is represented by a red circle around the aircraft item. When an aircraft track is hooked, the information in the "Close Control" window displays the International Civil Aviation Organization (ICAO) designation, Flight number (#), latitude, longitude, speed in knots, vertical rate (VRATE), heading in degrees, and altitude (ALT). For example, in Figure to the right, A0B379 is hooked and shows its data in the window.

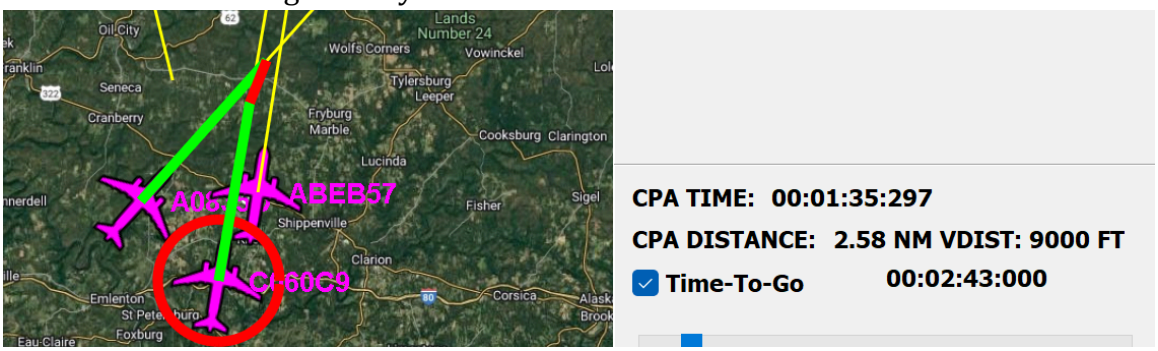


The Close Control window also displays the number of message counts which correlates to how many aircraft are being tracked. Note that the message count delineates between Raw data (or data from the Raspberry Pi Flight Tracker) or SBS data (or community hub data).

Notice that the aircraft images show a yellow line extending from the nose of the aircraft, these lines are called heading lines or track lines. The slider bar on the lower right, above the zoom in and zoom out buttons, allows the user to make the heading line longer or shorter. The heading lines may be turned off by unchecking the "Time-To-Go" box.



The RUI also provides the option of comparing the "hooked" aircraft (discussed above) with a nearby aircraft's closest point of approach (CPA) by holding the control button and right clicking another aircraft. The CPA information is shown in the lower right corner, specifically CPA Time and CPA Distance given both in lateral distance of nautical miles and vertical distance given in feet. On the map itself, the two aircraft's heading lines are changed to the color green and connected by a red line to show the CPA geometry.

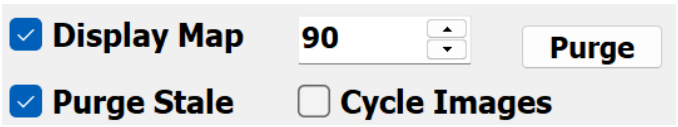


Above the CPA display, a drop down menu for map data is provided. The default is Google Maps data. The user may toggle between map data displayed with the other map types as Visual Flight Rules (VFR), Instrument Flight Rules (IFR) Low altitude, and IFR High altitude.

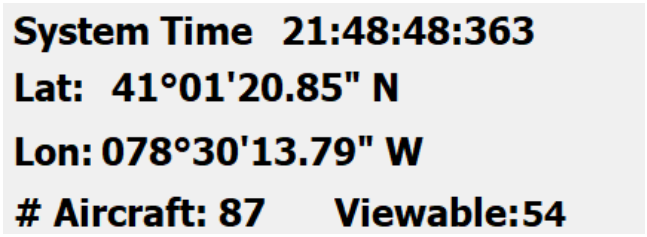
In the "Areas of Interest" above the log data, the user may draw polygons on the map. By clicking on "Insert", the user can add points on the map by right clicking at desired points. When finished adding points, the user left clicks on "Complete" and follows the window prompts to name the polygon and choose a color for the

polygon. Of note, while drawing the polygon, clicking on cancel will stop the process of drawing polygons. After drawing the polygon, the polygon information is displayed in the "Area of Interest" box. Users can delete the polygon by selecting the polygon name in this box and then clicking "Delete".

Above the Area of Interest box, the user has several mode options to select from. These are Raw Connect, Raw Record, Stop Raw Playback, SBS Connect, SBS Record, and SBS Playback. See mode section below.



Above the modes, a couple of viewing options are available. The "Display Map" check box allows the user to turn the maps on and off in the main display. The text box to the left of the "Purge" button sets the stale timer to remove tracks if no updates are received within the specified time. The "Purge Stale" check box must be checked for this feature to be active, or stale tracks will remain displayed on the map indefinitely. Clicking on the Purge button itself removes all currently displayed tracks. Last, the "Cycle Images" randomly assigns different aircraft images to newly received aircraft tracks. If unchecked, all aircraft use the default aircraft image.



Finally, in the upper right corner, the system information is displayed. System time is the user's laptop clock information (what is displayed on a Windows machine lower right corner). The latitude and longitude information represents where the mouse cursor is located on the displayed map. The "# Aircraft" counts how many ADS-B messages are being actively received, but these could be ground stations rebroadcasting or aircraft transmitting on the ground. The "Viewable" counts how many aircraft tracks/images are currently displayed on the map (the global map not the user view).

Raspberry Pi Flight Tracker

The Flight Tracker consists of a Raspberry Pi 5, software defined radio (SDR), and an antennae. Collectively, these components along with dump1090 software and SDR software enable the Raspberry Pi Flight Tracker to become an Automatic Dependent Surveillance – Broadcast (ADS-B) receiver. "dump1090" is a popular open-source software tool used to decode ADS-B messages transmitted by aircraft on the 1090 MHz frequency.

The ADS-B receiver collects local (e.g., 100 mile radius) 'raw' data over the 1090 MHz frequency band. The Raspberry Pi runs the software to manage the software

defined radio (SDR). The Raspberry Pi establishes a server where the server transmits the ADS-B data over TCP, and the user interface is a client on the same wifi network.

Student teams register their Raspberry Pi Flight Tracker with ADSBHub.org. Account registration requires providing information such as station name, location, host IP address, TCP port number, and few other parameters. Once registration is complete, the Raspberry Pi Flight Tracker is configured to establish a TCP connection to data.adsbhub.org to start streaming ADS-B data to the community. When streaming data to ADSBHub, the user may begin to access the community aggregate feed.

BigQuery

Google BigQuery is a fully-managed, serverless data warehouse that enables fast SQL queries using the Google Cloud Platform infrastructure. It's designed for analyzing large datasets with scalability, real-time insights, and built-in machine learning.

The project sponsor, SolveIt Inc., recognizes that recording SBS data (community ADS-B information) requires about ~15 mb of storage per minute. Planning worst case scenario, recording SBS data for one hour requires about 1 Gb of data storage. The length of time in recording the data will easily exhaust computing and storage capability locally on both the Raspberry Pi and student computers.

Therefore, Google BigQuery will be provided to student teams as an external data warehouse service to support team architecture, data needs, and additional features. Student teams may add this functionality in a number of options. For example, the Raspberry Pi could stream ADS-B data to BigQuery, the user interface could stream ADS-B to BigQuery, or the user interface could stream SBS data to BigQuery.

Before streaming data into BigQuery, ensure the destination table is created in advance. Streaming requires a predefined schema; BigQuery does not auto-create tables during real-time data ingestion.

System and Software Quality Attributes

The project sponsor, SolveIt Inc., determined that the top quality attributes are performance, resiliency, and extensibility, but conversations may reveal other important quality requirements. The project sponsor promotes creativity by project teams as there will be design trade offs depending on how teams' prioritize these three attributes. In conversations with SolveIt, the teams may elicit other lower priority quality attribute requirements. A short description of the key quality requirements are provided below:

- The system provides quick responses to user interface actions. It includes system response time when changing aircraft focus—retrieving and rendering metadata on screen. This is both measurable (in milliseconds) and observable (perceived delay). Additionally, performance depends on the number of items being monitored, as increased load may impact responsiveness.

- The system is able to handle and recover from faults or disruptions gracefully. It includes how the GUI reacts when the Raspberry Pi connection is lost (e.g., Wi-Fi turned off)—ideally, the application should detect the disconnection, alert the user, and attempt reconnection without crashing.

- The system can easily be enhanced with new features or modified to accommodate evolving requirements. For a user interface, this includes the ability to integrate new functionalities such as identifying unregistered aircraft or detecting deviations from registered flight paths. A highly extensible system allows these capabilities to be added with minimal impact on existing code, supporting modular development and future scalability.

- A developer can easily switch the user interface application to use a different map provider. Switching the map component should be done with minimal code updates, low risk of introducing bugs, and without impacting unrelated parts of the system.

- The software system can be easily understood, corrected, improved, and adapted over time. For a user interface, the goal is not to exhibit defects or errors. An essential element to achieve this goal includes a codebase that is clean, well-documented, has good test coverage, and is structured in a way that simplifies debugging and enhancements.

Overall Supported Modes of Operation

Raw Connect - In this mode, the Raspberry Pi Flight Tracker is collecting ADS-B messages and transmitting the message data to the user interface over TCP for display.

Raw Record - In this mode, Raw Connect is active and the aircraft position data is written to a local file.

Raw Playback - In this mode, the user interface reads the Raw Record local file, and the recorded aircraft position data is displayed in the user interface.

SBS Connect - In this mode, the user interface connects to ADSBHub.org and accesses the community aggregated ADS-B feeds.

SBS Record - In this mode, SBS Connect is active and the aircraft position data is written to a local or external file.

SBS Playback - In this mode, the user interface reads the SBS Records from a local (e.g., Big Query), and the recorded aircraft position data is displayed in the user interface.

Fault Conditions

When the Raspberry Pi locks up or disconnects from the internet (Wifi), two conditions result. First, if the tracks' purge rate is set to a long duration, the user may not notice stale tracks in the user interface and assume the system is functioning properly. Second, since the Raspberry Pi is locked or shutdown, Raw Connect will not work and SBS Connect will not work (since the station is no longer transmitting data).

Another fault condition will occur if the SDR is disconnected from the Raspberry Pi. The Raspberry Pi will detect the missing SDR and disrupt the execution of the dump1090 software. The effect experienced from the user interface is similar to the lock up or wifi disconnection discussed in the preceding paragraph.

Desired and Mandatory Features

The following features remain open to be implemented in the prototype / MVP provided. Two categories are provided. Mandatory features are features that the student teams must implement. Desired features are features that student teams will receive bonus points for accomplishing.

Mandatory Features:

Aircraft and Route Metadata

- Look up aircraft information (e.g., Airline)
- Look up route information
- Determine aircraft point of origin and destination
- Use APIs and pull information (e.g., https://vrs-standing-data.adsb.lol/)

Time Series Analysis

- Track history - Flight tracking of time and location
- Know age of track (drop after 30 seconds)

Enhancements to User Interface

- Improve the look and user experience with the user interface
- Update map tiles
- Plot airports
- Modify aircraft icons and leaders

Desired Features:

Safety and Deviation Analysis

- Collision avoidance - Assess aircraft within proximity to each other
- Compare contrast a flight's SBS or ADS-B data with planned route
- Analyze aircraft flight profile for anomalies (e.g., flying to the wrong airport)
- Is the flight deviating from the flight plan?

Time Series Analysis

- While receiving and recording data, play back tracks (Playback at 2x at 3x speed) by moving a mouse cursor
- Track simulation
- Dead reckoning - plot tracks now or where they should be based upon constant velocity and altitude (parametric fit)

System Hardware

This section describes the hardware configuration for this project that consists of a Raspberry Pi 5 and a software defined radio (SDR) with antennae. Note that hardware modifications are extremely limited. Please contact Solvit, Inc if your team desires to modify the hardware.

The Raspberry Pi 5

The CanaKit Raspberry Pi 5 Starter Kit PRO - Turbine Black (128GB Edition, 16GB RAM) is a high-performance kit featuring the Raspberry Pi 5 with 16GB RAM, a 128GB microSD card, premium turbine-style cooling case, USB-C power supply, and essential accessories. This kit and Pi 5 is very capable of running the ADS-B software defined radio (discussed in the next section).

Kits will be provided already assembled to the student teams.

Figure 3: Raspberry Pi 5 Single Board Computer

Software Defined Radio

The Software Defined Radio (SDR) comprises a radio frequency (RF) tuner chip (R820T2) made by Rafael Micro, a demodulator chip (RTL2832U) made by Realtek, a TCXO (Temperature-Compensated Crystal Oscillator) that generates the reference clock signal with a 0.5 Parts Per Million frequency stability rating, a blue rugged aluminum case, industrial microSD card, and antenna—ideal for reliable ADS-B signal reception.

Putting it all together, the Blue R820T2 RTL2832U SDR is a combo of chips used in USB radio dongles that allow your computer to receive and analyze radio signals over a wide frequency range—just by using software.

Figure 4: ADS-B SDR w/Amp and 1090 Mhz Filter, Antenna

Router

A Wi-Fi Router similar to one shown in Figure 5 will be provided along with a label that can be found on the bottom of the router which has its access point name and the password. Teams must configure their Raspberry Pis to connect to the Wi-Fi access point name found on the bottom of the Raspberry Pi.

The Wi-Fi Routers provided are for student use away from campus to facilitate SSH into the Raspberry Pi on a local network. The router network does not connect to the world wide web unless configured by student teams in that manner.

Figure 5: Wi-Fi Router

Of note, the principal Wi-Fi used will be CMU-Secure and / or personal mobile hotspot. Students are expected to configure their Raspberry Pi with CMU- Secure and establish a mobile hotspot with their Windows PC.

Team Laptops

Teams will use their individual personal laptops to serve as part of their development environment and to run the remote user interface application. Therefore, team laptops will need to be configured to connect to CMU-Secure, the provided Wi-Fi Router, and establish a mobile hotspot.

System Software

Commercial and Open Source Software

The Raspberry Pi™ 5 Single Board Computer will be configured to run the Raspbian Bookworm operating system (Raspberry Pi OS (64-bit) with desktop) in Wi-Fi headless mode.

Required Student Personal Computer Software

Secure Shell (SSH) network protocol software

Some form of SSH capable telnet client and secure file transfer software will need to be installed on laptops to communicate with the Raspberry Pi™ 5 for remote shell access and file transfer.

For Windows laptops, you may use:

- PuTTY - an SSH and telnet client for the Windows platform
- WinSCP - provides secure file transfer between a local and a remote computer and provides a basic file manager and file synchronization functionality.

For Macintosh laptops, you may use:

- Mac Terminal- the built in SSH and telnet client. Mac Terminal also natively offers 'scp' as a shell command for copying files to a remote machine.
- Cyberduck or Filezilla – which provides secure file transfer between a local and a remote computer and provides a basic file manager and file synchronization functionality.

Integrated Development Environment Software

Embarcadero C++Builder Community Edition
https://www.embarcadero.com/products/cbuilder/starter

Baseline Code Archive

The provided zip files contain folders and files to support the following:

1. Software to run a remote user interface
2. Software to run the SDR and dump1090
3. Several csv files containing aircraft and other flight informationThe issued

Raspberry Pi's will come with ADSB-PI.img loaded on the SD card. This image is also provided in the Studio Project files.

The following additional steps are applied in the base image of the Raspberry Pi:

- sudo apt update
- sudo apt full-upgrade
- sudo apt install librtlsdr-dev
- sudo apt install netcat-traditional

- git clone https://github.com/antirez/dump1090.git
- cd dump1090/
- make

Dump1090-main.zip will be provided also, though the git clone and make commands installed these files.

There will be additional steps to configure the Raspberry Pi, and these steps will be posted in a separate document.

The project teams will be provided with user interface application code in the following zip drives:

- ADS-B-DisplayV31.zip

Network Connection

The Raspberry Pi is set to connect to Wifi with the following settings:
SSID: LGRPI
Password: LG_Project

Set your Windows PC mobile hotspot initially to the SSID and password provided for initial connection with the Raspberry Pi. The mobile hotspot window will display the IP address connected. SSH into the Raspberry Pi with the IP address shown and the login information below.

Default Raspberry PI™ 5 Login Information:
The default username and password for your Raspberry PI™ 5 is below:
Username: `lg`
Password: `lg`

You should login as the pi user, which is a normal user. Root commands can be run as the root user by using the `sudo` command before the program you want to run.

After initial connection, configure wifi settings to connect to CMU-Secure and your team's router.

If you want to install additional software on Raspbian OS using the `sudo apt install` tool, then your Raspberry Pi will need to be connected to the internet either via CMU-Secure or your PC's mobile hotspot.

Project Deliverables

There are three milestones required for this project (each of which will be graded independently):

1. **Requirements, project plan, plan for experimentation, and risks**: For the first milestone the team will turn in the prioritized architectural drivers, any draft design decisions they are considering, the technical risks identified, and the experimentation plans (or any results already achieved).
2. **Experimentation results, design, plan for construction**: The team will turn in the results of experiments conducted, a design description with different architecture views, and the plan for construction.
3. **Demo and lessons learned**: For the final milestone, the team will demo the completed system, and present the lessons learned.

Milestone 1

The submission can be informal documents describing the team's current understanding of the items listed below. Team mentors will be meeting with the teams to ask follow up questions.

- Project Plan
    - The plan should describe the division of roles, the specific tasks planned, and the associated milestones.
    - The tasks should reflect construction based on the overall architecture.
    - The tasks should also reflect planned technical experiments.
- Architectural Drivers
    - Are the quality attribute requirements "actionable"? In other words, are they expressed in such a way that the team will be able to determine if a given design supports these drivers or not?
    - Do the drivers seem to relate to the overall objectives of the project?
    - Are the measures clearly derived from the overall goals of the project?
    - Are the functional requirements understood?
    - Is there a mechanism for prioritizing the requirements?
- Risk Assessment/Planned Experiments
    - What are the technical and non-technical risks? How do you assess each risk with respect to probability and impact in a High-Medium-Low scale?
    - Are the open questions/issues clearly related to things that will affect the outcome of the project?
    - Have there been any actions identified to address the open questions/issues?
    - Are the technical experiments concretely articulated?
    - Is it clear what question/issue is being addressed by the experiments?
    - Will it be clear when the experiments are complete?
- Architectural Approaches
    - What is the overview-level description of the architecture?

    - What are the main architectural approaches (tactics, patterns, design strategies) in your solution?
    - Is the proposed design sound enough to guide construction?
    - Are the architectural approaches clearly related to the drivers (will they likely impact the properties of interest)?
    - Can you explain how well the architectural drivers are supported by the architecture?

Milestone 2

Again, the submission can be informal documents describing the team's current understanding of the items listed below. The mentors will be meeting with the teams to ask follow up questions.

- Project Plan
    - How has the plan changed?
    - Has the team been actively assessing risk and updating the plan accordingly?
    - Does the team have a plan for any remaining significant issues/risks?
    - Does the team have a reasonable construction plan?
- Experiments/Results
    - What experiments have been conducted?
    - Have the results of the experiments addressed the open questions/issues?
    - What experiments remain?
    - Are the experiments focused on issues relevant to the overall goals of the system?
- Architecture
    - What is the architecture in terms of the organization of code units and their dependencies? (The team shall create at least one module view of the architecture.)
    - What is the architecture in terms of components and connectors (runtime perspective)? (The team shall create at least one runtime/C&C view of the architecture.)
    - What is the architecture in terms of the supporting infrastructure (deployment perspective)? (The team shall create a deployment view that shows high-level component allocation to hardware elements and communication channels.)
    - Have the experiments led to a refinement of the architecture?
    - Does the team understand the architectural approaches they selected and respective tradeoffs?
    - Do the architectural approaches align with the goals of the system?
    - Are there significant concerns that have not been addressed?
    - Has the architecture been evaluated?

Milestone 3

For the final deliverable there will be both a team presentation and a demonstration of the final system.

Team Presentation

The presentation should cover:

- Quality attribute requirements for the system.
    - Present evidence that the performance approach is sound.
    - Present evidence that the structural design supports the addition of new features.
    - Present evidence through verification test data that the system achieves active measures to manage loss of communication.
    - Present evidence how the developer/maintainer can easily replace map data in the user interface.
    - Present evidence of good code quality (e.g., low defect numbers)
- Architecture description showing architecture views and highlighting key architectural approaches adopted and their rationale.
- Description and results of experiments and architecture evaluation activities.
- Lessons learned (what went right, what went wrong, what would you have done differently).

The presentation duration is 20 minutes (subject to change). That is not enough time for an extensive coverage of the topics listed above. Therefore, you should select one or a few key points in each topic to talk about.

Demonstration (also known as 'Demo')

There will be a live demo of the system. Guidelines and evaluation criteria:

- Your Raspberry Pi Flight Tracker will be placed outside of the room near a window.
- The student team inside the classroom connects to the Raspberry Pi Flight Tracker through the remote user interface.
- The student team connects to the ADSBHub.org data.
- The student team demonstrates basic functionality
    - Using each of the modes
    - Displaying map data
    - Displaying flight information
- The student team demonstrates the added features:
    - Safety and Deviation Analysis
    - Aircraft and Route Metadata
    - Time Series Analysis
- The student team demonstrates how system meets the quality requirements

- The student team demonstrates upgrades to enhance the user experience with the remote user interface
- Maximum time limit for the demonstration will be 20 minutes.
- Additional guidance will be provided 1-2 weeks from the scheduled demonstration date.

Demo Scoring

Teams are ranked by total points scored. The maximum possible score is 200 points with an additional 10 bonus points.

Possible Grading Criteria Architecture Class:

Functional capability [40 points]
Additional features or capabilities added. Each feature graded as works completely/works partially/doesn't work or not implemented. Examples include:
- Detect airplanes on collision course
- Detect unregistered airplanes
- Detect deviations from registered flight path
- Lookup aircraft information
- Record history to file (need to determine what specific information should be recorded)

Response time [30 points]
Response time for the remote user interface application, Raspberry Pi, and external database when actions are initiated by the user in the remote user interface. Examples include:
- Timely respond when the data load scales
- Timely respond when executing added analytical features

Fault detection and error handling [30 points]
Examples include:
- How does the GUI application react when it loses connection with the Raspberry Pi? (Test turning off wifi)
- How does the system react when the Raspberry Pi loses connection with the ADS receiver? (Test pulling the USB cable)

Architecture supports new requirements or enhancements [30 points]
Examples include:
- Are key functionalities abstracted behind interfaces?
- Can new behavior be injected without modifying existing code?
- Are dependencies explicitly managed?

Map Support [20 points]
The GUI application should support multiple map data types and different levels of scale. Examples include:

- Support map data from different map providers
- Support zoom in and zoom out features

System and software quality [- 3 points per error observed for up to -24 points]

- Point deductions for any errors

Remote User Interface (Max 50 points + 10 possible bonus points)

- Real time display of maps and aircraft tracks
- Real time display of "hooked" aircraft
- Display aircraft data and other display data
- Implement click buttons (or similar feature) to remotely connect to the Raspberry Pi Flight Tracker, ADS-B Hub, recorded files, or other data files
- Implement slide bars or similar features to promote interaction of flight track data on the map visually
- Display polygons and show only aircraft that are within the defined waypoints on the map (e.g., square or irregular shape following an highway)
- Best user interface (UI): 10 points; 2nd Best UI: 8 points, 3rd Best UI: 6 points, 4th Best UI: 4 points, 5th Best UI: 4 points(points awarded based on client/sponsor judgment)

Assumptions and Hints

- Each team will be given a complete Raspberry Pi Flight Tracker Kit (e.g., with Raspberry Pi, SDR, etc.)
- Each team's Raspberry Pi will come pre-loaded with a base image with dump1090 installed.
- Each team will be given a router to support local network testing between Raspberry Pi and remote user interface.
- Each team will be given sample source code for the remote user interface.
- Each team will be given Google Cloud Platform credit to use Google BigQuery (SW Architect - $100 per team)
- The team may use any third-party software but the choice of technology must be justified.
- Assume that anything can fail… your design should account for failure and recovery to the greatest extent possible.
- Teams will use their own laptops for software development and to execute the Flight Agent user interface.

- The provided software will function as shown in the Kick Off demonstration; however, there may be better ways to design or implement functionality and quality attributes.