TCAS Digital Twin - Final Report

1st Muhammad Muzamil
High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
muhammad.muzamil@stud.fra-uas.de

2nd Muhammad Seerat
High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
muhammad.seerat@stud.fra-uas.de

3rd Ahmed Tahir Shekhani
High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
ahmed.shekhani@stud.fra-uas.de

4th Ali Salman
High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
ali.salman@stud.fra-uas.de

5th Tabassum
High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
tabassum.tabassum@stud.fra-uas.de

6th Sadia Saeed
High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
sadia.saeed@stud.fra-uas.de

***Abstract*—This report presents a simplified digital twin of TCAS II, developed in MATLAB based on FAA guidelines. It simulates mid-air collision avoidance by monitoring two aircraft on the same flight level (altitude) using predefined position and velocity inputs. The system calculates distance and time-to-closest approach to assess threats and issues either a Traffic Advisory (TA) or Resolution Advisory (RA). A radar-like interface visually alerts the pilot in real-time. The development followed a safety-critical workflow, including FMEA and STPA analysis, ensuring the system behaves predictably under various scenarios and aligns with TCAS II principles.**

I. Project Status Report

Our project goal was to design and simulate a simplified version of Traffic and Collision Avoiding System (TCAS) using MATLAB. Our system aimed to detect any mid-air collision threat between two aircraft at the same altitude and issue appropriate advisory (TA or RA) according to the guidelines issued by FAA (Federal Aviation Agency).

A. Initial Scope

Initially we planned to make a system that can handle both vertical and horizontal scenarios but due to complexity and MATLAB learning curve we revised the scope of our project to focus only on horizontal scenario at the same flight level. Our system is designed for two scenarios:

- Head-on-head conflict (opposite direction)
- Overtaking conflict (same direction but different speeds)

B. Key Milestones Achieved

- Designed and implemented core TCAS algorithms (Identify/Track and Threat Detection & Elimination).
- Built 2D simulation in MATLAB with two visual outputs: Simple 2D Chart showing aircraft movement and a pilot's radar.
- Performed Failure Mode and Effect Analysis on our design.
- Performed STAMP Hazard Analysis using STPA to derive Safety Constraints.
- Performed STPA-SafeSec analysis to derive Security Constraints.
- Updated Safety and Security plan.
- Partial implementation of Safety and Security Constraints.
- Manual implementation of SIL level.
- Basic manual testing.

C. Pending/Not Included

- Real-time data feed through transponder or other means.
- Implementation of some Safety and Security Constraints.
- Automatic Sensitivity Level [1] implementation.
- Audio alert.

II. Problem Description

TCAS prevents collisions by defining a protection boundary around each aircraft. The main goal is to keep intruder aircraft from crossing this boundary by predicting future positions and initiating vertical maneuver if necessary.

Terms: Below are some terms which one should know before understanding this system:

- **Range:** The horizontal distance between your aircraft and another aircraft, measured in a straight line.[1]
- **TAU:** A measure of time (in seconds) until two aircraft reach their closest point of approach. It's calculated by dividing the distance between the aircraft by their closing speed.[1]
- **DMOD (Distance Modification):** A minimum distance threshold used by TCAS to ensure alerts are triggered even if two aircraft are closing slowly.
- **Range Test:** A check performed by TCAS to determine if an aircraft is close enough (based on TAU or DMOD) to issue a Traffic Advisory (TA) or Resolution Advisory (RA).[1][2]
- **Protection Volume:** A 3D "safety bubble" based on DMOD and TAU, around an aircraft that TCAS monitors for potential threats.

- **Traffic Advisory (TA):** A warning from TCAS indicating that another aircraft is nearby and may pose a potential threat. It alerts the pilot to look for the traffic but does not require evasive action.[1]
- **Resolution Advisory (RA):** A direct instruction from TCAS recommending a climb or descent maneuver to avoid the collision.[1]

The solution to this problem is as follows:

Our digital twin monitors predefined aircraft parameters:

- Horizontal Position (x)
- Altitude (y)
- Velocity (vx,vy)
- Aircraft ID

The system simulates aircraft motion and computes:

- Relative Position (Range)
- Closing Speed
- Time to Closest Approach (TAU)

It compares these outputs (Range and TAU) against predefined protection volume (DMOD and TAU) to assess the threat level (Range Test)[1]. Based on this evaluation, the system generates one of the following advisories [1][2]:

- Traffic Advisory: Indicates a potential threat and alerts the pilot to monitor the intruder.
- Resolution Advisory: Indicates an immediate threat and suggests a climb or descend maneuver.

In case of RA, the system uses a deterministic rule set (based on aircraft ID parity, speed, and direction) to assign vertical maneuvers that avoids conflicting actions.

Our system also includes a radar-like Human-Machine Interface (HMI) that displays both aircraft, the advisory status, and visualizes proximity and threats in real-time.

III. System Design Model

A. System Architecture

The TCAS II system is divided into 4 components, each responsible for a specific task. Below are the components:

- Identify & Track
- Threat Evaluation
- Advisory Selection
- Radar

These components interact sequentially: aircraft data is first processed by Identify & Track, followed by threat detection, which passes advisory instructions to the Advisory Selection module. The final result is displayed to the pilot via a radar-style HMI.

B. Component Description

1) Identify & Track: The Identify and Track component is responsible for detecting the presence of other aircraft and continuously track its position.

Input:

- Position (a1.x, a1.y, a2.x, a2.y)
- Velocity (a1.vx, a1.vy, a2.vx, a2.vy)
- Aircraft IDs (a1.ID, a2.ID)

Responsibilities:

- Compute Range (distance) between two aircraft.
- Compute Closing Speed
- Compute TAU
- Forward computed Range and TAU to threat detection component.
- Continuously update Radar with position of intruder aircraft.

2) Threat detection: [1] This component determines whether the aircraft are on a collision course using predefined protection volume and the computed values (Range, TAU) from Identify & Track.

Input

- Range
- Current TAU
- Protection Threshold values: TA_DMOD, TA_TAU, RA_DMOD, RA_TAU.

Responsibilities

- Perform range test i.e Compare current TAU and Range with protection threshold values.
- Determine Threat: **No threat (safe separation), Traffic Advisory (TA), Resolution Advisory (RA)**
- Forwards threat level to advisory selection component.

3) Advisory Selection: This component issues appropriate advisories based on the threat level. In case of RA, it assigns non-conflicting vertical maneuvers to each aircraft.

Input

- Threat Level (TA/RA)
- Aircraft IDs
- Velocity Vectors of both aircraft

Responsibilities

- For RA: Display "Traffic Traffic" alert on Radar
- For TA:
  - Decide which aircraft should climb and which should descend.
  - Decide and assign which aircraft will climb and which will descend using the following logic:
    - If one ID is even and the other is odd: `Even → Climb`, `Odd → Descend`
    - if both IDs are even or both are odd: The aircraft with **higher speed** climbs.
    - If both speeds are equal: The aircraft with **positive horizontal velocity** climbs, the one with negative descends.
  - Send maneuver advisory to the Radar.

4) Radar: The radar is the visual output module for the pilot. It provides situational awareness to the pilot.

Responsibilities:

- Display real time position of both aircraft.
- Show current advisory status with clear color-coded alerts.
- Provide audible alerts (e.g., "Traffic Traffic" or "Climb / Descend")

C. Algorithms

1) Step 1: Calculate Range: Horizontal distance (Range) calculated between both aircraft using Euclidean distance formula:

$$\text{Range} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

2) Step 2: Calculate Closing Speed:

$$\text{Closing Speed} = \frac{(x_2 - x_1)(v_{x2} - v_{x1}) + (y_2 - y_1)(v_{y2} - v_{y1})}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{2}$$

3) Step 3: Calculate TAU: The TAU is the estimated time it will take for the two aircraft to reach their point of closest approach, assuming constant velocity:

$$\text{TAU} = \frac{\text{Range}}{\text{Closing Speed}} \tag{3}$$

If the aircraft are flying in same direction and at same speed (i.e., closing speed ≥ 0), TAU is set to infinity.

4) Step 4: Threat Detection: After computing TAU and Range, the system performs range test (compares these values to threshold protection values) [1].

- **Traffic Advisory (TA)** is issued if:

$$\text{TAU} < \text{TA\_TAU} \quad \text{or} \quad \text{Range} < \text{DMOD\_TA}$$

- **Resolution Advisory (RA)** is issued if:

$$\text{TAU} < \text{RA\_TAU} \quad \text{or} \quad \text{Range} < \text{DMOD\_RA}$$

5) Step 5: Advisory Selection: When a threat is detected, the advisory logic decides how to respond:

- **If TA:**
  - Display the alert "Traffic, Traffic" on the Radar.
- **If RA:**
  - **If aircraft IDs differ in parity:**
    - Even ID → `Climb`
    - Odd ID → `Descend`
  - **If both IDs have same parity:**
    - Aircraft with higher speed → `Climb`
    - Aircraft with lower speed → `Descend`
  - **If speeds are equal:**
    - Aircraft with positive vertical velocity → `Climb`
    - Aircraft with negative vertical velocity → `Descend`
- The advisory is then sent to the Radar system for visualization.

D. UML Diagrams

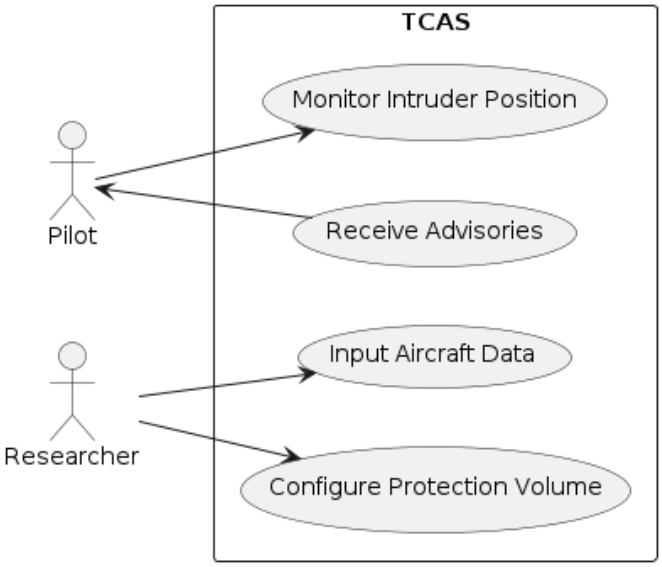

Fig. 1. Use Case Diagram - TCAS

Use Text Case 1

- **Use Case Name:** Monitor Intruder Position
- **Actor:** Pilot
- **Preconditions:** Simulation is running and radar UI is active.
- **Steps:**
  - The system continuously tracks both aircraft.
  - Intruder aircraft positions are sent to the Radar UI.
  - The pilot views real-time location of the intruder.
- **Postcondition:** The pilot maintains situational awareness.

Use Text Case 2

- **Use Case Name:** Receive Advisories
- **Actor:** Pilot
- **Preconditions:** The simulation has started and threat evaluation is active.
- **Trigger:** An intruder aircraft violates the protection volume.
- **Steps:**
  - TCAS detects threat level via thresholds.
  - A TA or RA is generated.
  - The advisory is sent to Radar UI.
  - The pilot sees "TRAFFIC TRAFFIC" or "CLIMB CLIMB NOW / DESCEND DESCEND NOW" command.
- **Postcondition:** The pilot receives timely alert and can respond accordingly.

Use Text Case 3

- **Use Case Name:** Input Aircraft Data
- **Actor:** Researcher
- **Preconditions:** Researcher has the simulation system available.

- **Trigger:** The researcher wants to simulate an encounter scenario.
- **Steps:**
  - The researcher edits the aircraft data in MATLAB script.
  - Researcher saves the MATLAB Script.
  - The system stores the data for simulation.
- **Postcondition:** Initial aircraft data is saved and available for processing.

Use Text Case 4

- **Use Case Name:** Configure Protection Volume
- **Actor:** Researcher
- **Preconditions:** Researcher has the simulation system available.
- **Trigger:** The researcher wants to define thresholds according to the sensitivity level.
- **Steps:**
  - The researcher selects a sensitivity level.
  - Researcher edits the Protection Threshold variables in MATLAB script according to Sensitivity level.
  - Researcher saves the MATLAB script.
- **Postcondition:** Protection thresholds are ready for use in simulation.

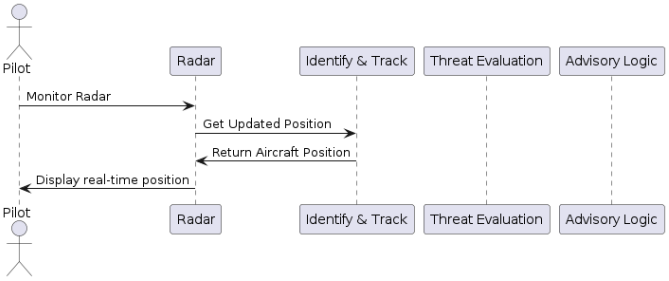

Fig. 2. Sequence Diagram (Pilot - Monitoring Aircraft Position)

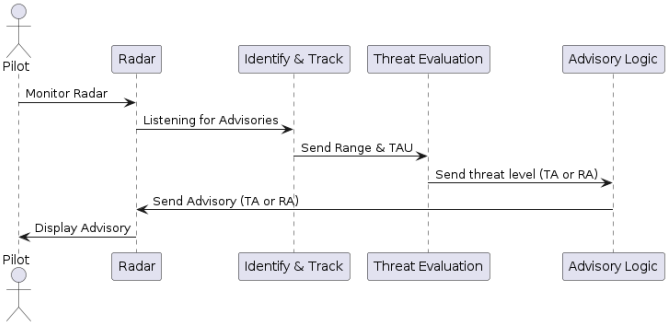

Fig. 3. Sequence Diagram (Pilot - Receiving Advisories)

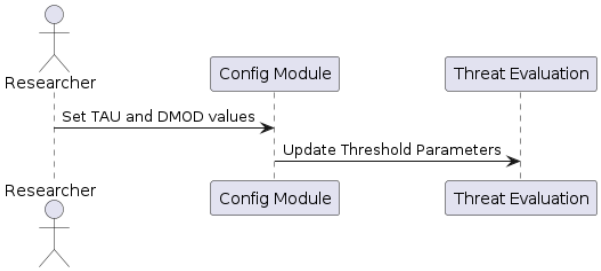

Fig. 4. Sequence Diagram (Researcher - Input Aircraft Data)

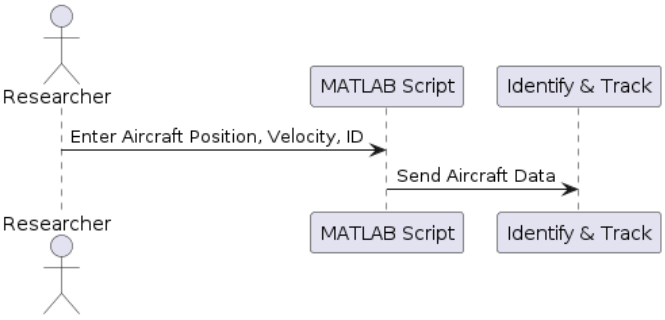

Fig. 5. Sequence Diagram (Researcher - Configure Protection Thresholds)

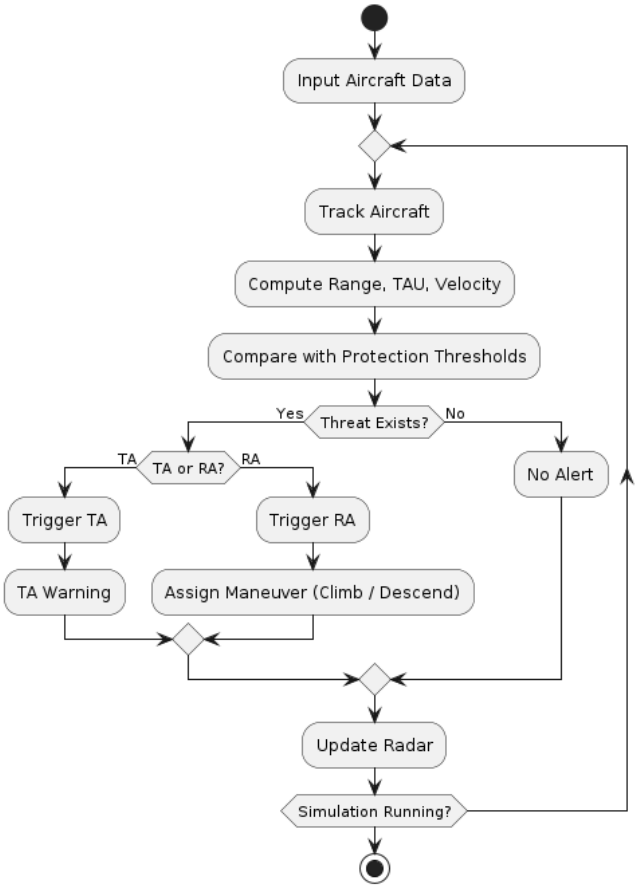

Fig. 6. Activity Diagam - TCAS

IV. Human Machine Interaction

The radar (HMI) in our system provides the pilot with real-time position of the intruder aircraft, potential collision threats and necessary maneuvers (RA). We designed the pilot's radar/UI similar to the real TCAS radar (according to FAA) but obviously it can be improved a lot. Our radar is minimalistic, intuitive and easy to interpret under stress.

A. Radar Display (Pilot's View)

The radar is centered around our own aircraft and shows the relative position of intruder aircraft[1][2]. Key features are listed below:

- Concentric rings representing 5 km, 10 km, and 15 km ranges.
- The intruder's position is updated in real-time.
- Threat status is color-coded:
  - **Red:** Resolution Advisory (RA)
  - **Orange:** Traffic Advisory (TA)
  - **Blue:** No Alert
- Text fields display:
  - Altitude, Range
  - Threat Level (TA/RA)
  - Resolution Command (Climb or Descend).

B. 2D Aircraft Simulation

This view provides horizontal simulation of both aircraft trajectories:

- Flight paths of both aircraft are plotted.
- Current positions are marked with colored symbols.
- A connecting line shows their current separation.
- A side panel shows real-time calculations:
  - Range, TAU, and Closing Speed
  - Thresholds for TA and RA
  - Current advisory for each aircraft

C. Alerting Mechanism

- Visual alerts appear on the radar screen.
- The radar displays advisories through text commands such as: ``TRAFFIC TRAFFIC'' or ``CLIMB CLIMB NOW'' or ``DESCEND DESCEND NOW'', according to the guidelines of FAA[1][2].
- The goal is to minimize ambiguity and guide the pilot clearly.

V. Updated Project Plan

A. Project Scope

This project aims to design and simulate simplified version of TCAS II system using MATLAB. The simulation will only focus on horizontal encounters scenarios at same flight level, where aircraft fly at constant velocities and aircraft positions are predefined The objective is to simulate that the system can:

- Track nearby aircraft using predefined input data
- Evaluate threat levels using simplified TCAS logic
- Issue appropriate advisories (TA/RA)
- And visualize system outputs for pilot awareness through a radar interface.

The simulation is limited to the following 2D Horizontal encounter scenarios:

- Head on head conflict (aircraft approaching from opposite directions)
- Same direction conflict (one aircraft overtaking another or both flying at similar speeds)

B. Formal Specifications

1) Functional Requirements:

- **F1:** The TCAS system shall accept predefined position (x, y) and velocity (vx, vy) inputs for two aircraft.
- **F2:** The system shall compute the separation distance and closing rate b/w both aircraft.
- **F3:** The system shall use separation distance and closing rate to determine if any threat exists.
- **F4:** If a threat exists, the system shall determine the severity of the threat.
- **F5:** The system shall trigger advisory based on the severity of the threat.
- **F6:** The system shall display the advisory output onto the radar/screen/UI, to the pilot.

2) Non-Functional Requirements:

- **NF1:** The simulation shall run within MATLAB/Simulink.
- **NF2:** The system shall respond within 1 simulated second.
- **NF3:** The same input data shall always produce the same output (deterministic).
- **NF4:** The system shall simulate each scenario independently.

3) Process Model: For the development of TCAS, V Model XT will be used. V Model XT strongly supports traceability, validation and alignment with safety standards like IEC 61508. The V Model XT allows team to plan weekly builds and review while maintaining the structure. This approach minimizes risk and eases validation, ensuring deliverables and final documentation are complete and on schedule.

C. Project Estimation

We have used COCOMO II Model (Post-Architecture Stage 3) because we need to get the best estimation of our project and balance the workload with other courses as well. It helped us in getting the time duration required and calculated the effort based on certain questions like, reliability required, do we know the architecture or is the team well connected. Based on certain assumptions we got the estimation and based on this estimation we squeezed the tasks to just core requirements to meet the completeness.

D. Project Risks

- **MATLAB/Simulink Learning Curve:** Team members have limited or no prior experience with MATLAB/Simulink which may delay development or lead to incorrect modeling.

- **Incorrect Implementation of TCAS Logic:** Misinterpretation of TAU, DMOD or RA rules may result in incorrect or unsafe advisory outputs in the simulation.
- **Team Communication:** Delays in responses and lack of regular check-ins may lead to fragmented progress, missed updates and last minute firefighting.
- **Threshold Selection Uncertainty:** Multiple literature sources with varying parameters may lead to confusion in selecting appropriate thresholds for TCAS implementation.
- **Project Timeline Variability:** Increasing workload from other courses and shifting priorities may affect the estimated project timeline.
- **Limited Simulation Scope:** Simplified simulation approach (single direction, horizontal constant approach at y=0) may not fully represent real-world scenarios.

VI. Hazard Analysis

A. Classical Methods

1) FMEA - Failure Mode and Effect Analysis: We chose to do Failure Mode and Effect Analysis (FMEA) on our design because of:

- Modular structure and software driven nature of the system.
- FMEA is more suitable to analyze potential failure points at component level.

Since our simulation is software based so we had full control over input and logic. FMEA allowed us to model software errors in a structured way.

We analyzed our functions for:

- Possible failure modes
- Their causes
- Their effects
- Proposed detection/prevention measures

FMEA Table: *FMEA Table/Working has been included in Hazard Analysis.pdf due to space issues.*

FMEA Result:

- Multiple high risk failures were identified in Identify and track module which were mainly caused by bad or missing inputs.
- Threat Evaluation was found to be vulnerable to misconfigured or missing thresholds or stale data. This potentially led to missed, late or no advisories at all.
- The advisory selection logic posed serious risks if Aircraft IDs were not unique, had same parity which could lead to failure in assigning maneuver.
- Through this analysis we came across multiple edge cases where our logic would fail. Through FMEA we were able to find multiple failure modes in our components which can be easily mitigated through software constraints such as range checks, validation of input data and redundancy.

B. STAMP - STPA

We also performed a detailed and traceable STAMP analysis with STPA which is purely focused on the software driven hazards and control logic of our system. *STPA has been included in Hazard Analysis.pdf due to space issues.*

- **Hazard Identification:** First we identified and listed down all potential hazards that would lead to unsafe system behavior. We identified 15 hazards. These included incorrect advisory issuance, misconfigured thresholds, late reaction times and ambiguous maneuver assignment.
- **Unsafe Control Actions:** After that we created a table of control actions performed by software components and analyzed how these control actions could become unsafe. Each UCA was also mapped to one or more previously identified hazards. Each UCA was classified as:
  - Not provided when needed
  - Provided incorrectly
  - Provided too late
  - Provided out of context
- **Casual Scenarios:** After that we brain-stormed and documented real-world reasons why these Control Actions would be unsafe and can take place.
- **Safety Requirements:** After that we revised our safety requirements that reflect what our system must do to remain safe.
- **Safety Constraints:** Using UCAs and safety requirements we formed software level safety constraints for each component.

Conclusion: STPA helped us in understanding how unsafe software behavior could merge even when components don't physically fail. STPA added more depth and traceability to hazard analysis. Especially it helped us understand how multiple operations can lead to unsafe state if not coordinated properly. It helped us uncover another important use case when a plane can be advised to climb but the plane is already at maximum allowed altitude.

VII. STPA-Sec

For security hazard analysis we followed a similar process. It extends STPA method to include the cybersecurity aspects. Our goal was to analyze how control actions could lead to violations of system confidentiality, integrity and availability. *STPA-Sec has been included in Hazard Analysis.pdf due to space issues.*

- **Security Hazards Identification:** In the first step, we identified potential security hazards which could compromise security of our system. These included:
  - Unauthorized access or modification of the source code
  - Accidental overwriting or deletion of advisory logic
  - Inability to trace who made changes or when
  - External interference during simulation via Wi-Fi, Bluetooth, Remote Access
- **Unsafe Security Control Action:** Next, we identified control actions or missing actions that could lead to security breaches.

- **Security Requirements:** After that we formed security requirements to ensure system integrity and confidentiality.
- **Security Constraints:** Lastly we translated security requirements into security constraints to be applied on the processes.

VIII. Safety Plan

A. Safety Measures

- **Validation Checks:** Perform unit testing and scenario based testing to verify that threat detection and advisory logic behave as intended.
- **Simulation Based Validation:** Run a variety of aircraft encounter cases to test advisory accuracy and timing.
- **Safe State Fallback:** In the event of inconsistent or invalid input data, the system will halt advisory generation.
- **Bound Checks:** Ensure altitude, distance and velocity inputs remain within defined operational ranges.
- **Traceability:** Every system related requirement will be mapped to a test case to support system level validation.

B. Safety Requirements

- **SR1:** The system shall fail safely by halting the advisory generation if input data and threshold values are invalid, missing or internal errors are detected.
- **SR2:** The system shall issue a valid Traffic Advisory (TA) or Resolution Advisory (RA) in a timely manner whenever protection volume values (TAU, DMOD) are violated.
- **SR3:** protection volume values should be adjusted according to the altitude.
- **SR4:** Under normal conditions and correct logic, simulated aircraft should not enter each other's protection volume.
- **SR5:** The system shall never issue contradictory advisories such as TA instead of RA or vice versa or both aircraft receiving climb or descend.
- **SR6:** The system shall issue advisory early enough, giving the pilot a manageable time to execute the maneuver.
- **SR7:** The radar shall present intruder's position, threat status and advisory message in a visually intuitive and easily understandable format.

C. Safety Constraints

- **SC 1:**
  - The system shall validate all aircrafts' data (position, velocity, IDs) before computing Range and TAU.
  - If the inputs are missing or are out of bounds, simulation should update the radar with the error message for the pilot and should halt the simulation.
- **SC 2:** The system shall pre check all divisions and mathematical operations such as (TAU = Range / Relative Velocity) and should avoid execution if the denominator is 0 or undefined.
- **SC 3:** The system shall validate protection volume values. If protection volume values are missing, out of bound or misconfigured (TA = RA or RA = TA), the system shall not generate any advisory.
- **SC 4:** The system should automatically set the protection volume values based on the aircrafts altitude, according to the Sensitivity level.
- **SC 5:** The advisory module must validate that the threat level is correctly interpreted before issuing TA or RA. TA shall not be issued when RA conditions are met and vice versa. This must be ensured through distinct comparison logic for both TA and RA thresholds.
- **SC 6:** The system should use redundant conditions such as Speed, Velocity while assigning a RA and should not solely depend on Aircrafts' IDs for assigning RA.
- **SC 7:** The system shall check if the altitude before giving a RA, making sure the maneuver could be executed properly.
- **SC 8:** The UI should update the aircrafts' positions, threats and advisories in real time (1 sec) with visual makers and labels, clearly showing intruder identity and type of advisory.
- **SC 9:** The advisory displayed must be accompanied by audio alert and should persist for at least 3 seconds to ensure pilot's acknowledgment.
- **SC 10:** In case of system failure, data error or missing inputs, the radar must display 'System Error / Advisory Halted' warning so that the pilot is aware of advisory suspension.

IX. Security Plan:

A. Security Measures

- **Simulation Integrity:** Validate that test inputs are not unintentionally overwritten and that all simulated inputs are consistent with the project specification.
- **Threshold Value Protection:** Advisory thresholds and core logic parameters will be protected from unintentional editing using parameter encapsulation or code modularity.
- **Access Control:** Limit access to simulation files, source code and configuration parameters to authorized team members only.
- **Version Controlling:** Use versioning (Git) to track and manage changes, ensuring traceability and rollback options.

B. Security Requirements

- **Sec-R1:** The source code should be protected from unauthorized access and modification.
- **Sec-R2:** All source code changes must be tracked through version control (Git) with proper commit documentation.
- **Sec-R3:** Only Authorized team members should have access to code repository.
- **Sec-R4:** All purposed changes to source code shall require team approval prior to implementation.
- **Sec-R5:** During simulation execution, the system shall run in an offline environment to prevent external interferences.

C. Security Constraints

- **SC-Sec 1:** The source code shall reside in a private GitHub repository with restricted access.
- **SC-Sec 2:** Only authorized group members should be added as collaborators with commit/push permissions.
- **SC-Sec 3:** All changes must be peer reviewed and approved before being committed to the repository.
- **SC-Sec 4:** All commits must be documented with messages describing the change purpose.
- **SC-Sec 5:** During simulation runtime, Wifi, Bluetooth should be turned off to prevent remote interferences.

X. Prototype

A. Screenshots of running project

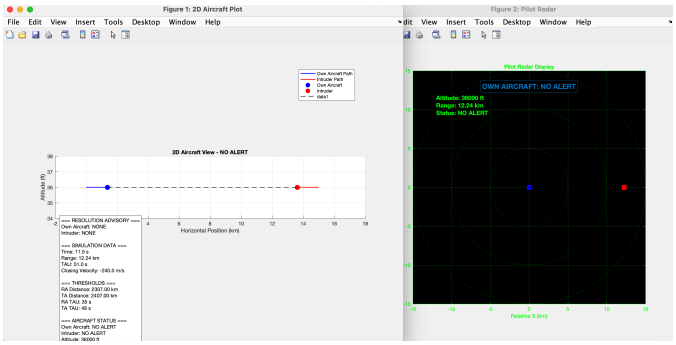

Fig. 7. 2D Simulation & Pilot's Radar (No Alert)

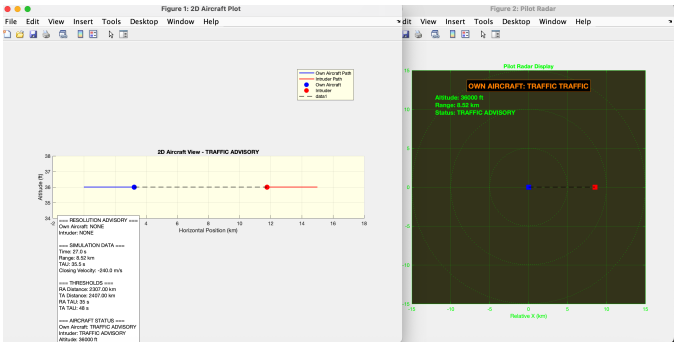

Fig. 8. 2D Simulation & Pilot's Radar (RA Triggered)

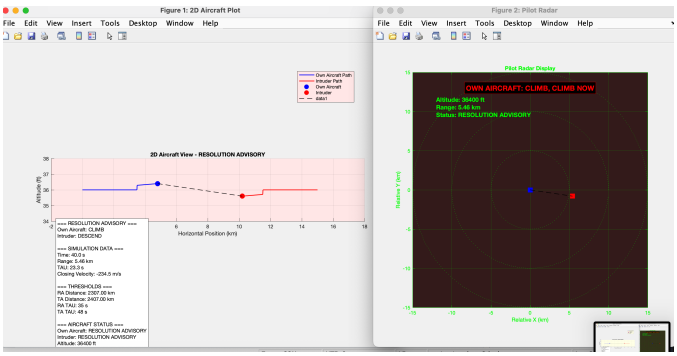

Fig. 9. 2D Simulation & Pilot's Radar (TA Triggered)

XI. Project Contribution:

TABLE I
Team Member Contributions

Work Product	**Contributor(s)**	**Contribution Summary**
Literature Review & Scoping	Muzamil, Seerat, Ahmed	Basic literature review conducted by everyone, filtering matching to project's scope by Muzamil
Software Project Plan	All Team Members	Collaborative effort to define scope, process model, project estimation, schedule, risks, and resources.
Requirement Analysis	Sadia, Muzamil	Initial requirements provided by Sadia, revised to match the scope and finalized by Muzamil.
System Design Model	Muzamil	Designed architecture & algorithms - fully designed the working of TCAS.
UML Diagrams (Use Case, Sequence, Activity)	Muzamil, Ali, Tabbasum	Use Case Diagram: Tabbasum (revised by Muzamil), Activity: Ali (revised by Muzamil), Sequence: Muzamil
Programming & Simulation (MATLAB)	Muzamil	Programmed basic logic, improved via external sources (YouTube, Web articles etc).
FMEA Hazard Analysis	Muzamil, Seerat, Ali, Sadia, Tabassum	Identify & Track: Muzamil, Threat Detection: Seerat, Advisory Selection: Sadia & Tabbasum, Radar: Ali & Ahmed
STPA / STPA-SafeSec Analysis	Muzamil	Performed detailed software-level hazard analysis and derived safety/security constraints.
Final Documentation (LaTeX)	Muzamil, Ahmad	Muzamil compiled and wrote the final report, Final Project Plan was Updated by Ahmad.

References

[1] Federal Aviation Authority - FAA, Introduction to TCAS II Version 7.1. [Online] Available: https://www.faa.gov/documentlibrary/media/advisory_circular/tcas%20ii%20v7.1%20intro%20booklet.pdf [Accessed: Jul. 3, 2025]

[2] International Civil Aviation Organization - ICO, Traffic Alert and Collision Avoidance System II (TCAS II) Tutorial [Online] Available: https://www.icao.int/NACC/Documents/Meetings/2024/GTE24/GTE24-P03.pdf [Accessed: Jul. 3, 2025]