FAMU-FSU College of Engineering
Department of Mechanical, Electrical, and Computer Engineering

Team 315
Team Members: Colby Hackett, Jimmy Lu, Cameron Sayers, Morgan Skinner, Jackie Ou, Jonathan Tooby

Date: 11/8/24

Preliminary Detailed Design

Table of contents

Introduction (Problem Statement, Motivation, and Requirements)

In recent years, the potential of drones in education and industry has grown immensely, but access to training tools remains limited due to their high cost. The high price of drone hardware and professional-grade simulators often exceeds $500, presenting a significant barrier for schools, hobbyists, and smaller organizations, especially with their limited budgets. For many underfunded educational institutions and students, access to these technologies is out of reach. This issue contributes to a growing gap in STEM education, leaving underprivileged communities with fewer opportunities to explore the field of drone technology.

Our project addresses this need by developing a flight simulator that combines affordability with functional ability, providing a realistic drone flight experience without the high prices of premium systems. By offering an accessible, low-cost alternative, our simulator aims to bridge the gap for less-funded environments and foster exploration and growth in both drone technology and STEM.

To achieve this, our simulation must meet several key requirements:

1. **Drone Simulation:** It should realistically mimic the controls and flight dynamics of an actual drone, giving users an authentic, hands-on experience.
2. **FPV Capability:** Equipped with a front-mounted camera, the system will offer first-person view (FPV) capabilities to enhance user immersion.
3. **AI Integration:** The simulation should be able to use at least one AI model, allowing for experiments with AI-based features.
4. **VR Compatibility:** The simulator will support virtual reality (VR) headsets to enable a fully immersive FPV experience.
5. **Custom Controller:** A custom-designed controller with haptic feedback will provide a tangible response, enhancing the user experience.
6. **Cost-Effectiveness:** Above all, the simulator must remain affordable to ensure accessibility for educational institutions and organizations.

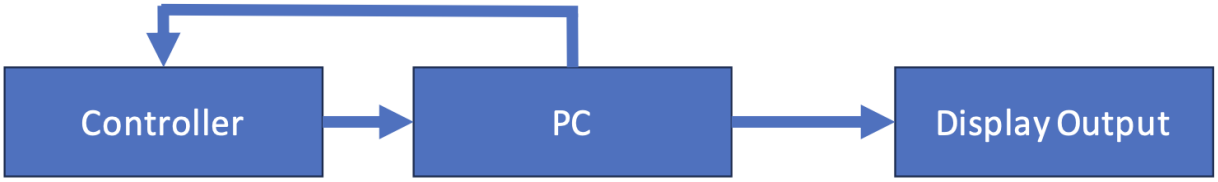
Our approach goes a step further than existing simulators by incorporating AI functionality, paving the way for additional features and capabilities that traditional systems lack. This project aims to provide schools and individuals with a comprehensive, interactive learning tool that encourages hands-on exploration of drone technology and STEM concepts, all while keeping costs within reach.

Selected Concept

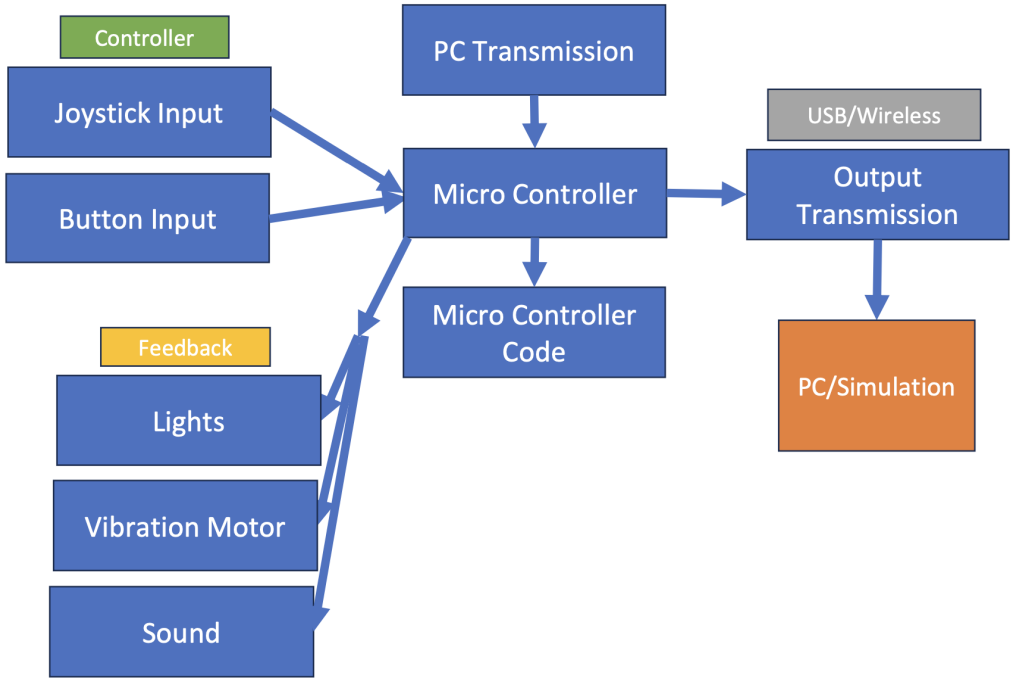
Our selected concept regarding our drone simulator involves using a gaming controller paired with a VR headset to enhance immersion. We will have the users navigate an obstacle course within a 3D urban environment developed in Unity. The course is intended to replicate real-life challenges a drone pilot might encounter, requiring precision and control to navigate through obstacles like rings, narrow passageways, and varying elevations. The VR setup allows users to experience first-person perspective (FPV) from the drone, giving them a realistic sense of depth, speed, and spatial awareness. This setup aims to provide an engaging, hands-on experience for users as they learn and practice piloting skills in a safe, simulated environment. Unity is a strong choice for this project, especially with its VR capabilities, versatile development tools, and robust 3D environment support. This setup in Unity should allow for a well-integrated, immersive experience that meets the project's needs for both realism and functionality in the simulated urban course.

Preliminary Design

High-level block diagram



Controller block diagram

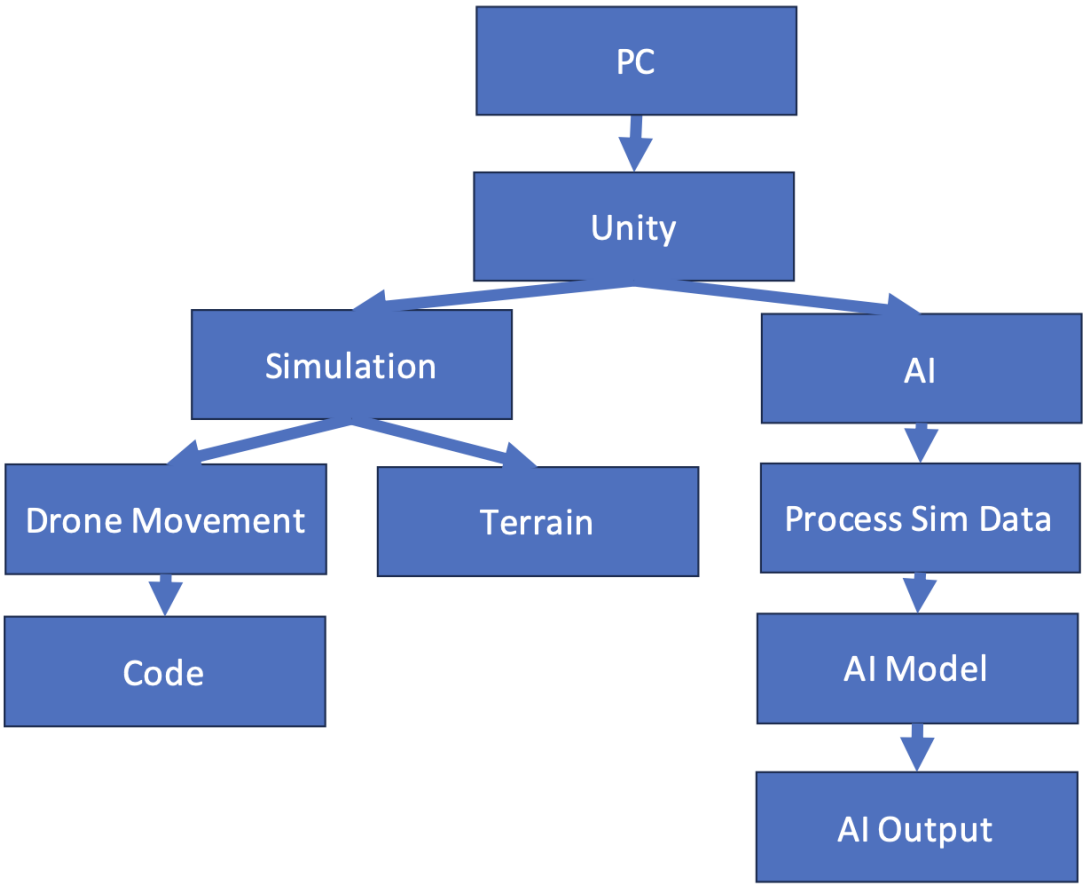


Components and Functions:

1. Joystick Input:
    - Component: Analog joystick(s).
    - Function: Captures directional control inputs (pitch, roll, throttle, yaw) for the drone.
2. Button Input:
    - Component: Digital buttons.
    - Function: Captures discrete actions (e.g., start, stop, take-off, landing) via button presses.
3. Microcontroller:
    - Component: Embedded system (e.g., Arduino, STM32).
    - Function: Processes inputs, manages communication with PC, and controls feedback (lights, vibration, sound).
4. Microcontroller Code:
    - Component: Firmware.
    - Function: Interprets inputs, translates them into commands, and manages communication with the PC.
5. PC Transmission:
    - Component: Communication protocol (e.g., UART, USB, wireless).

    - Function: Transmits data from microcontroller to PC/simulation.
6. Output Transmission:
    - Component: Output module (USB or wireless).
    - Function: Sends control signals to the PC or simulation system.
7. Feedback Components:
    - Lights: LEDs for visual feedback (e.g., power, mode).
    - Vibration Motor: Provides haptic feedback for actions/events (e.g., button presses, warnings).
    - Sound: Buzzer/speaker for auditory feedback (e.g., startup, alerts).
8. PC/Simulation:
    - Component: PC running simulation software.
    - Function: Receives inputs, controls the drone in the simulation, and sends feedback to the controller.

PC block diagram



Components and Functions:

- PC:
  - Function: Acts as the main hardware platform where the simulation is hosted and processed.
  - Components: Desktop or laptop computer, operating system compatible with Unity and VR software.
- Unity:
  - Function: The primary development environment where the simulation, physics, and user interface are built.
  - Components: Unity Engine, asset libraries for 3D models, scripts for simulation controls, and plugins for VR and AI integration.
- Simulation:
  - Function: Manages the real-time flight simulation, including drone physics, controls, and environmental interactions.
  - Components:

- Drone Movement: Handles the control and behavior of the virtual drone, simulating aspects like lift, thrust, pitch, yaw, and roll.
    - Code: Underlying scripts and algorithms that define the drone's physics, control responses, and movement mechanics.
- Terrain: Creates the simulated environment, including obstacles and paths within the obstacle course.
    - Components: 3D models of terrain, obstacles, and environmental elements that the drone will navigate through.

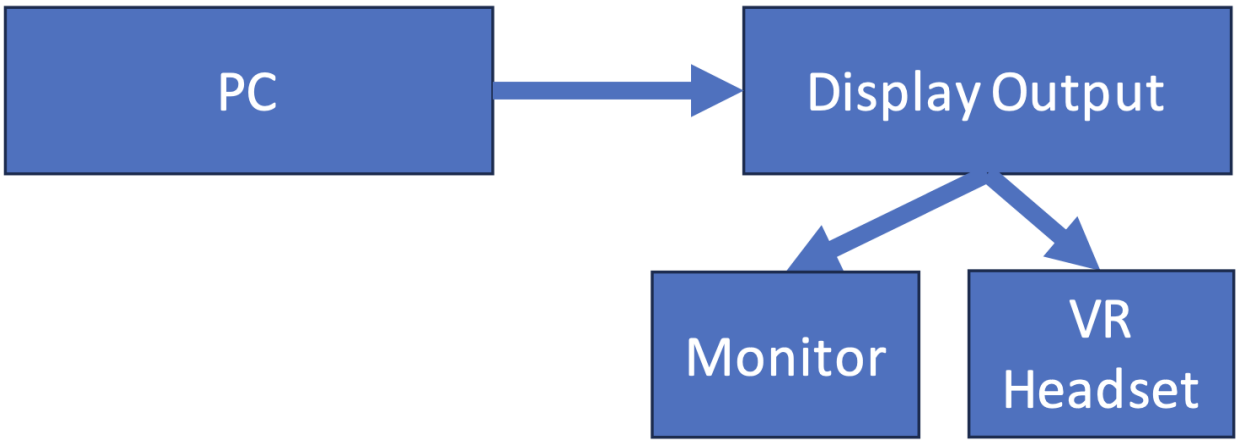
- AI:
    - Function: Provides intelligent features and functionalities, such as autonomous flight, obstacle avoidance, or path planning.
    - Components:
        - Process Sim Data: Collects and interprets data from the simulation (e.g., drone position, velocity, and environmental feedback).
        - AI Model: Uses the processed simulation data to train or execute AI algorithms (e.g., reinforcement learning models for autonomous flight).
        - AI Output: Provides actionable insights or control signals back to the simulation, enabling the AI-driven behavior of the drone in real time.

Pseudocode for drone movement block:

```
//Initialize drone and controls
Initialize DronePhysics()
Initialize UserControls()
While simulationIsRunning:
    throttle = GetUserInput("Throttle")   // Vertical movement control
    yaw = GetUserInput("Yaw")             // Rotational movement along the vertical axis
    pitch = GetUserInput("Pitch")         // Forward and backward tilt
    roll = GetUserInput("Roll")           // Left and right tilt

    //Calculate Forces based on Input
    thrust = CalculateThrust(throttle)
    yawRotation = CalculateYaw(yaw)
    pitchTilt = CalculatePitch(pitch)      // Forward/backward tilt based on pitch input
    rollTilt = CalculateRoll(roll)         // Left/right tilt based on roll input

    // Apply Physics Calculations
    // Adjust forces with Unity's physics engine
    ApplyThrust(thrust)
    ApplyYawRotation(yawRotation)
    ApplyPitchTilt(pitchTilt)
    ApplyRollTilt(rollTilt)
    //Update Position and Orientation
```

```
    newPosition = CalculateNewPosition()
    newOrientation = CalculateNewOrientation()

    // Set drone's position and rotation in Unity
    SetDronePosition(newPosition)
    SetDroneRotation(newOrientation)
    RenderDrone()
End While
```

Display block



Components and Functions:

- **PC**:
  - **Function**: Acts as the central processing unit for running the simulation and rendering graphics.
  - **Components**: Hardware capable of running Unity-based simulations and VR applications, along with software drivers for display outputs.
- **Display Output**:
  - **Function**: Manages the output of visual data from the simulation, determining how the user views the simulated environment.
  - **Components**: Graphics card and video output settings that control display resolution, frame rate, and compatibility with different display devices.
- **Monitor**:
  - **Function**: Provides a standard display option for users to view the simulation on a traditional screen.
  - **Components**: Standard computer monitor with appropriate resolution and refresh rate to display the simulation clearly. It offers an accessible option for users who may not have VR equipment.
- **VR Headset**:
  - **Function**: Offers an immersive first-person view (FPV) experience for users by displaying the simulation in a VR environment, enabling a more realistic and interactive experience.
  - **Components**: VR headset with motion tracking, VR-compatible software, and controllers (if applicable) for enhanced interactivity. The VR headset synchronizes with the PC and Unity to provide an immersive simulation experience.

Note on changes from functional decomposition:

The updated block diagrams provide a more detailed breakdown of the drone simulation system, expanding on key functionalities essential for the selected concept. Compared to the initial functional decomposition, these diagrams now delve deeper into specific components like "Control Drone Flight," which shows the interaction between user inputs (such as joystick and button inputs) and how they influence the drone's movement within the simulation. The controller diagram, in particular, illustrates the flow of data from input devices through the microcontroller to feedback mechanisms (like lights and vibrations) and finally to the simulation environment on the PC.

Summary

For our preliminary design, we focused on essential requirements, building our approach largely on guidance from our advisor. To meet the drone simulation requirement, we are using a VR headset, which integrates with Unity as our initial game engine for developing the simulation and custom controller. As the project progresses, we may consider other game engines if they offer improved features for our simulation. Our simulation will also incorporate an AI model that interacts with the user, providing feedback to help them improve their skills. In addition to meeting technical requirements, we are keeping project costs low as this is a key objective. We have developed preliminary designs, which may evolve as we dive deeper into the project.