Sentinel 360: Smarter Vision, Sharper Defense

Group #8

David Ursu

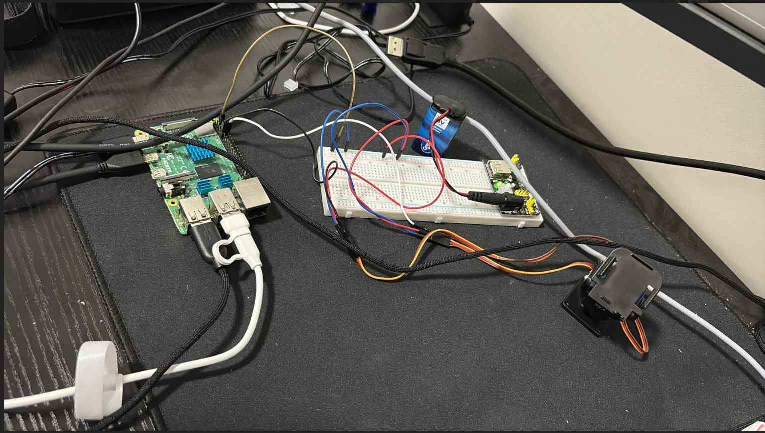
Umberto De Luca

Osman Ali Deol

Joel deHoog

Project Overview

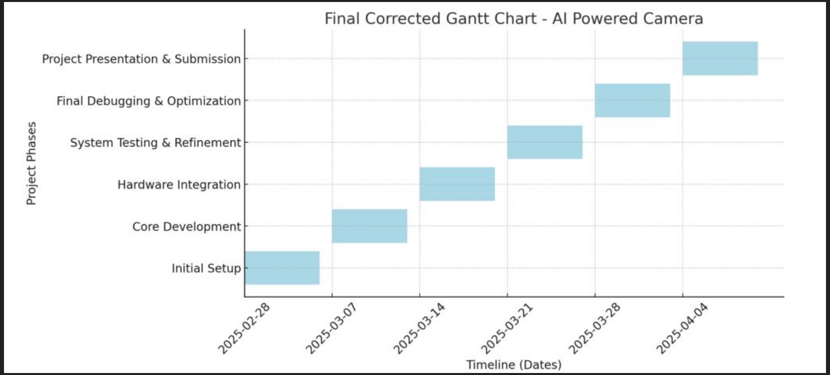
- **Goal**
  - To develop an AI-powered camera system capable of detecting, tracking, and responding to visual targets in real time using computer vision and servo-controlled motion.
- **Scope**
  - Integrating object detection, servo motor control, and a web dashboard interface to create a modular, real-time 360° surveillance and tracking system operated through both autonomous and manual modes.
- **Project Title**
  - Sentinel 360: Smarter Vision, Sharper Defense



Team Roles

- **David**
  - Hardware design, testing / debugging, and integration.
- **Umberto**
  - Hardware design, testing / debugging, and integration.
- **Osman**
  - Software development, testing / debugging, and AI & Object Detection
- **Joel**
  - Software development, and testing / debugging

Milestone Timeline



Week 1 - Hardware Setup & Design

- **Raspberry Pi Configuration**
  - Connected and tested communication between systems
- **Servo-Motor Mechanism Assembly**
  - Assembled servo-motors and tested with manual movement
- **AI Model Selection**
  - YOLOv8 for object detection and high performance

Milestone Timeline

Week 2 - AI & System Development

- **Object Detection Implementation**
  - implemented Python script to capture frames from a camera
- **Servo-Motor Control**
  - Translated object detection coordinates into movement commands
- **Communication Between PC & Raspberry Pi**
  - Established WebSocket communication for real-time data transfer

Week 3 - Bug Fixes & Initial Testing

- **Static Condition Testing**
  - Verified AI detection accuracy in different lighting conditions and simple scenarios
- **Controlled Movement Testing**
  - Ran tests with slow-moving objects to fine-tune tracking responsiveness

Milestone Timeline

Week 4 - Real-World Testing & Optimization

- Performance Evaluation
  - Analyzed servo response time and adjusted control logic accordingly
- Stress Testing & Final Bug Fixes
  - Conducted prolonged operation tests to check for overheating and power consumption
  - Addressed software bugs causing false detections or incorrect motor movements

Real World Applications

Aviation – Monitors runway traffic and ground crew operations to ensure safety and efficiency at airports.

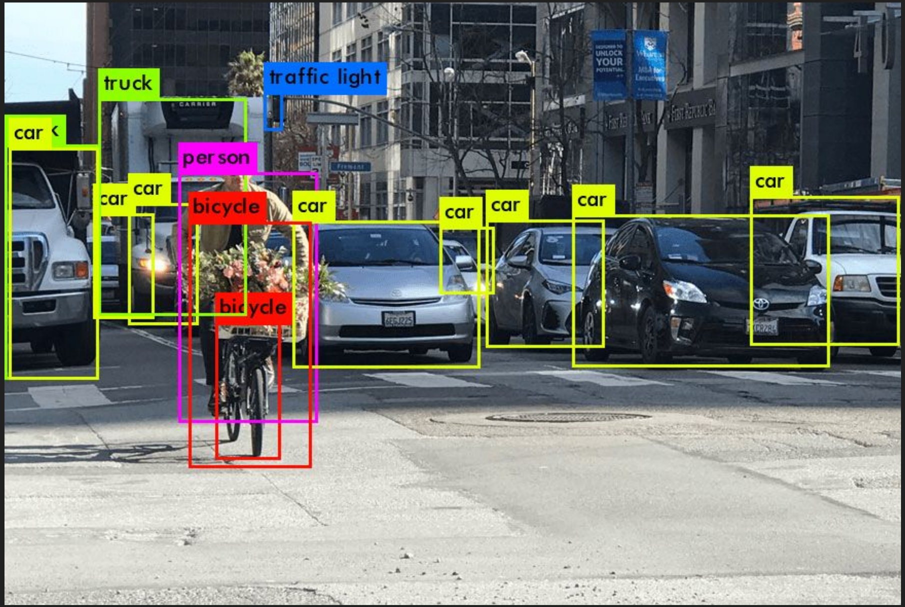
Cinematography & Broadcasting – Enables smooth, automated camera panning for live events and film productions.

Sports Analytics – Tracks players or equipment to capture performance metrics and improve viewer engagement.

Real World Applications

Driver Assistance & Autonomous Vehicles: Real-time detection of pedestrians, road signs, and obstacles to enhance safety and aid in self-driving systems.

Security & Surveillance – Automatically tracks suspicious activities or intruders in real-time for enhanced facility security.



References and Resources

Hardware

- Raspberry Pi, 2 SG90 micro-servos, camera module, pan tilt mechanism, MB102 Breadboard Power Supply Module

Software & Libraries

- Python (OpenCV, YOLO v8, and Flask).

Online Sources & Documentation

- Raspberry Pi documentation for servo pins, YOLOv8 docs for model training.

Design and Development Process

Workflow Design

- **Step 1**: Pi continuously captures images from the camera module.
- **Step 2**: AI software (e.g., YOLOv8) on a remote machine
- **Step 3**: When an object is detected, the Pi calculates the required servo angles to keep that object centered in the frame.
- **Step 4**: The Pi sends PWM signals to the SG90 servos via GPIO pins to adjust the pan/tilt angle accordingly. (Digital I/O Manipulation learned from lecture/ labs)
- **Step 5**: The MB102 module ensures a stable 5V supply so voltage fluctuations from servo movement don’t crash the Pi. (Hardware verification learned from lecture/labs)

Design and Development Process

Power Management & Stability

- Ensuring the Pi and servos had stable voltage from the MB102 module meant carefully checking current draw, voltage drops, and potential noise that could disrupt signals. Solving these issues required systematic testing and iterative refinement.

Servo Control & Camera Alignment

- Translating object coordinates from the AI model into precise servo angles demanded a blend of math (e.g., angles and offsets) and programming logic. Tuning servo movements highlighted problem-solving (e.g., dealing with servo jitter, finding correct pulse widths).

Time & Resource Constraints

- Balancing the scope (e.g., whether to add a web dashboard) with available time and computing resources required analytical thinking. The team had to set priorities, scrap or defer certain ideas, and adapt to real-world limitations.

Public Dissemination

- GitHub Repository
  - Full codebase: AI, Flask server, React dashboard
  - README with setup, architecture, and instructions
  - Commit history for progress tracking
  - Open-source for community learning and reuse

- YouTube Demo Video
  - Live demo of all system modes (Manual, Auto, Surveillance)
  - Narrated explanation of setup and architecture (Results and Findings)
  - Demonstrates real-time tracking and dashboard usage
  - Used for documentation and sharing with peers

Thank You!