Embedded Systems Project Proposal – CSEE4840

Project Name: Magic Beans

Group Members: Gurleen Kaur Kalra (gkk2122), Linus Lei (nl2997), Daolin Li (dl3832)

[1] Project Abstract

This project aims to use the DE1-SoC FPGA to build a real-time cyber-physical system for plant detection and health monitoring. A camera detects plants visually, while virtual sensors simulate environmental conditions such as soil moisture, light intensity, temperature, and humidity. The FPGA processes this data in real time and simulates actuator responses (water pump, lights, fan, heater) to maintain or improve plant health.

As the plant reaches a healthy state based on simulated sensor readings, the system displays graphics of a magical vine growing across the screen, culminating in a treasure animation. This provides a visual representation of successful plant health improvement. The project demonstrates real-time FPGA processing, image recognition, closed-loop feedback, and graphical visualization, combining embedded systems, cyber-physical systems, and real-time graphics.

This project demonstrates several important embedded-system concepts including:

- Real-time image processing on FPGA
- Hardware-based graphics generation
- Closed-loop control simulation
- Hardware/software system integration

The system will run on the DE1-SoC FPGA board and display results on a VGA monitor.

Fig1. Graphical illustration of Vine Growth from Jack and the Beanstalk

[2] Algorithms

a) Plant Detection (Camera Input):

1) Capture RGB frames from the camera.

2) Apply green color thresholds directly in the RGB space to segment plant regions (e.g., G > R + threshold, G > B + threshold).

3) Perform morphological operations (opening, dilation) to remove noise. (optional)

4) Detect the largest connected green region → calculate bounding box and centroid for vine animation start point.

b) Plant Health Monitoring (Simulated Sensors):

1) Generate virtual sensor readings for soil moisture, light, temperature, humidity.

2) Compare each value with predefined optimal thresholds.

3) Flag conditions as healthy or unhealthy.

c) Actuator Simulation (Virtual Actuators):

1) If a simulated sensor is below threshold → activate corresponding virtual actuator signal.

2) Update simulated sensor values over time to reflect “actuator effect.”

d) Vine Growth and Treasure Display:

1) Start vine animation from plant centroid once sensors indicate plant is healthy.

2) Animate vine growing upwards in real-time.

3) Once the vine reaches the top of the screen, display treasure animation.

e) Integration:

FPGA continuously loops through:

1) Read camera → detect plant

2) Generate simulated sensor readings → check thresholds

3) Update virtual actuators → modify simulated plant health

4) Animate vine + treasure based on health status

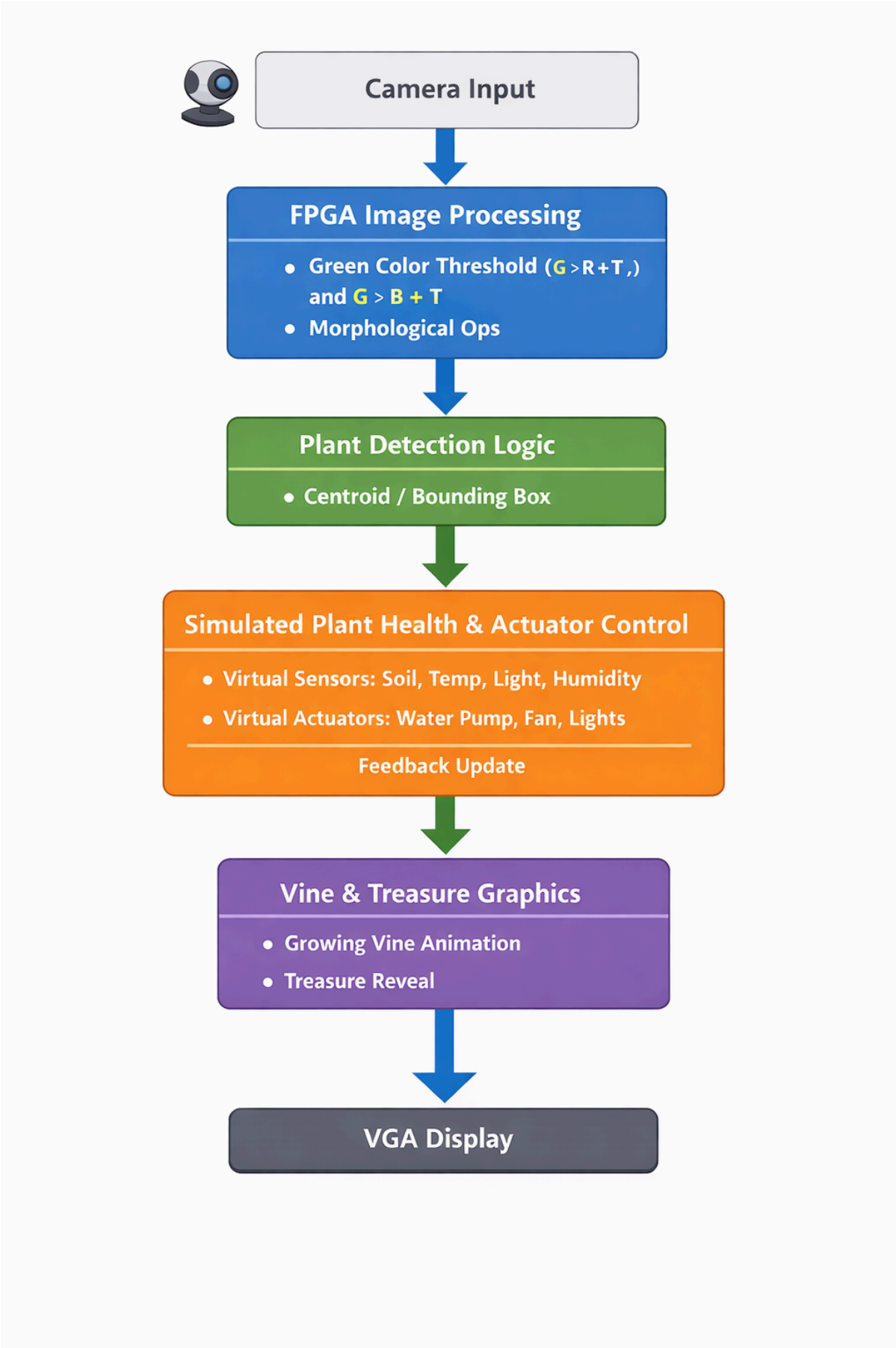

Fig 2. System Design

[3] Hardware and Software Required

Hardware:

1) DE1-SoC FPGA (Cyclone V)

2) Camera (USB webcam or compatible composite camera)

3) VGA Monitor for display

4) Optional: switches or keyboard for adjusting simulated sensor parameters

Software / IPs:

Verilog / FPGA side:

1) Image processing pipeline (RGB color threshold, morphological operations (optional))

2) VGA controller / buffer for graphics

3) Plant health monitoring logic (simulated sensors + actuators)

4) Vine and treasure animation pipeline

C / Software side:

1) Optional GUI for monitoring virtual sensors and actuators

[4] Project Milestones

Week	**Date**	**Milestone**
Week 1	**Mar 13**	Finalize project scope and architecture. Define system pipeline: camera input → RGB processing → plant detection → animation → VGA display. Assign tasks among team members.
Week 2	**Mar 20**	Set up development environment. Verify DE1-SoC board, camera interface, and VGA display. Test basic camera capture and display raw frames on VGA.

Week 3	**Mar 27**	Implement RGB threshold-based plant detection module in Verilog. Test detection using stored image frames or test patterns.
Week 4	**Apr 3**	Implement connected region detection or bounding box logic for plant location. Verify that plant region is correctly identified in the frame.
Week 5	**Apr 10**	Integrate plant detection with VGA display. Highlight detected plant region on the screen. Begin development of vine animation logic.
Week 6	**Apr 17**	Implement vine growth animation starting from plant centroid. Test animation pipeline independently.
Week 7	**Apr 24**	Integrate camera input, detection pipeline, and animation system into a full real-time system. Perform debugging and timing optimization.
Week 8	**May 1**	System testing and performance validation. Capture demonstration results and screenshots for the report. Start writing final project report.
Week 9	**May 8**	Complete final report, finalize system demo, and prepare presentation slides and demonstration video.

[5] References

1) Application of a Real-Time Field-Programmable Gate Array-Based Image-Processing System for Crop Monitoring in Precision Agriculture – this paper discusses FPGA usage for agricultural image processing systems, showing real-time implementation relevance.
🔗 https://www.mdpi.com/2624-7402/6/3/191

2) IoT-based real-time object detection system for crop protection and agriculture field security – a recent research work on embedded image processing for agriculture that combines object detection and real-time embedded systems.
🔗 https://link.springer.com/article/10.1007/s11554-024-01488-8

3) Cyber-Physical Systems for Smart Farming: A Systematic Review – overview of CPS applications in agriculture, supporting the concept of plant health monitoring as CPS.
🔗 https://www.mdpi.com/2071-1050/17/14/6393

4) Smart Cyber-Physical Systems for Controlled-Environment Agriculture – example of sensor-based monitoring and control in agriculture (not FPGA-specific but directly relevant to CPS concept).
🔗 https://link.springer.com/article/10.1007/s43926-025-00205-6

5) Lab 4: FPGA and Color Detection | ECE-3400 – tutorial showing practical color detection logic on FPGA, including RGB comparisons and thresholds, relevant to your plant detection algorithm.
🔗 https://pbc48.github.io/ECE-3400-Fall-2018/lab4.html

6) Image Processing Toolbox in Verilog (GitHub) – example repository showing image processing implementation on FPGA hardware, applicable for RGB thresholding and VGA output.
🔗 https://github.com/Gowtham1729/Image-Processing

7) Different Morphological Operations in Image Processing — A detailed tutorial explaining erosion, dilation, opening, closing, and structuring elements.
🔗 https://www.geeksforgeeks.org/different-morphological-operations-in-image-processing/

8) Image Morphology Examples (JuliaImages) — Covers common morphological operations like erosion, dilation, opening, and closing with visual examples.
🔗 https://juliaimages.org/stable/examples/image_morphology/image_morphology/

9) Binary Morphology in Image Processing (MathWorks) — Academic courseware describing erosion, dilation, and compound operations for binary images.
🔗
https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/images/responsive/supporting/academia/courseware/binary-morphology/binary-morphology.pdf

10) Morphological Image Processing Intro (BioimageBook) — Explains basic concepts of dilation, erosion, opening, and closing with diagrams.
🔗 https://bioimagebook.github.io/chapters/2-processing/5-morph/morph.html

11) Digital Image Processing — Morphology Lecture — University lecture examples covering erosion, dilation, and other morphological operations.
🔗
https://web.stanford.edu/class/ee368/Handouts/Lectures/Examples/7-Morphological-Image-Processing/