Introduction

Efficient resource management in agriculture and landscaping has become critically important due to mounting environmental pressures. Two of the most pressing issues are the unnecessary overuse of water for irrigation—leading to water scarcity and waste—and the excessive application of fertilizers and chemicals, which infiltrate the soil and contaminate groundwater.

This project addresses these challenges by leveraging real-time, sensor-driven monitoring to optimize irrigation precisely when and where it's needed. By integrating an ESP32 microcontroller with a VL53L8CX Time-of-Flight (ToF) sensor, the system can detect the presence and position of plants or objects in a monitored area. Coupled with instant wireless data transmission and automated control of watering systems, the setup enables the following environmental benefits:

- **Water Conservation:** Irrigation is triggered only when the sensor detects plant presence and proximity, reducing unnecessary watering and helping to preserve scarce water resources.
- **Targeted Fertilizer Application:** By knowing exactly where and when plants are present, the system can help guide precise application of fertilizers and reduce runoff—limiting the amount of chemicals infiltrating natural soil and groundwater.
- **Reduced Environmental Footprint:** Intelligent control systems such as this not only save resources but also help reduce the carbon footprint and ecological impacts associated with traditional, less-efficient agricultural practices.

This project demonstrates how low-cost, network-connected sensors and automation hardware can contribute to sustainable practices in agriculture, urban gardening, or landscape management. The following report details both the hardware and software necessary to build the system, so others can replicate and further adapt it to address environmental needs in their own communities.

Materials and Methods

Materials

- ESP32 Development Board: Primary controller running FreeRTOS.
- VL53L8CX ToF Sensor Module: Delivers 8×8 grid distance measurements for object/plant detection.
- Push-Button Switch: User input, event annotation.
- LED, relay, or actuator (connected to GPIO7): Controls irrigation.
- Wiring/Breadboard or PCB: For sensor, switch, and actuator connections.
- Client computer/device: Receives sensor data via TCP.
- Power Supply: For ESP32 and peripherals.
- Wi-Fi Network: For ESP32 to connect and transmit data.

Pin Assignments

Function	ESP32 GPIO	Notes
I2C SCL	9	ToF sensor
I2C SDA	8	ToF sensor
ToF sensor reset	5	XSHUT line
Output (Actuator)	7	Controls valve/LED/relay
Input (positioning marks reader)	4	With internal pull-up enabled

Methods

System Design

The system combines real-time sensing, network connectivity, and actuator control:

- Wi-Fi Setup: ESP32 connects as a station to a Wi-Fi network.
- TCP Server: Listens for clients on port 5055, streams data on connection.
- Sensor Initialization: VL53L8CX set to 8×8 mode, 10Hz data rate, using I2C.
- Distance Measurement: ESP32 polls the sensor, collects a 64-value frame representing distances in mm.

Object/Plant Detection:

- Calculates background distance (median).
- Compares each pixel to background; detects objects (plant presence) using a threshold.
- Calculates weighted centroid of detected zone.

Actuator Control:

- If detected object (plant) is close to the center, triggers GPIO7 (e.g., opens/closes irrigation valve).
- Otherwise, actuator remains off, preventing unnecessary watering.

Button Interrupts: Debounced, stamp events for annotation (e.g., manual override/mark).

Data Streaming: Sensor data frames, timestamps, and event marks are sent to network clients for monitoring or logging.

Software Flow

Written in C using ESP-IDF framework.

- Uses FreeRTOS for multitasking: independent tasks for TCP communication, sensor polling, and button handling.

- Hardware timer (GPTimer) ensures accurate event timestamps and debouncing.
- All configuration parameters (SSID, pins, thresholds) are user-adjustable.

Assembly

- Wire the ESP32 to the VL53L8CX using I2C (SCL: 9, SDA: 8) and sensor XSHUT to GPIO5.
- Connect button to GPIO4 (with internal or external pull-up to 3.3V).
- Connect actuator (e.g., relay valve) control input to GPIO7.
- Flash the ESP32 with the provided code, making any adjustments for your setup.
- Start ESP32; ensure it connects to Wi-Fi.
- Connect a client to ESP32's IP on TCP port 5055 to view or log streaming data.