Agricultural Automation Scripts

This repository contains three Python scripts for agricultural automation and monitoring systems.

Scripts Overview

1. 2_combo_proximity_bridge_debug.py

Purpose: Debug and test proximity sensor bridge combinations for agricultural equipment.

Features:

- Proximity sensor calibration and testing
- Bridge communication debugging
- Sensor data validation
- Error detection and reporting

Use Case: Debugging proximity sensors on farming equipment to ensure accurate distance measurements for automated navigation and obstacle detection.

2. 8_row_following_system.py

Purpose: Automated row following system for agricultural vehicles.

Features:

- Crop row detection and tracking
- Automated steering control
- Path correction algorithms
- GPS integration support

Use Case: Enables tractors and other farm equipment to automatically follow crop rows for precision farming operations like planting, spraying, or harvesting.

3. 9_crop_monitoring_system.py

Purpose: Comprehensive crop health and growth monitoring system.

Features:

- Crop health assessment
- Growth tracking and analysis
- Environmental data collection
- Alert and notification system

Use Case: Monitors crop conditions, detects issues early, and provides data for making informed farming decisions.

Requirements

System Requirements

- Python 3.7 or higher
- Windows/Linux/MacOS compatibility
- Hardware sensors (proximity, GPS, cameras as applicable)

System-Level Dependencies

Before installing Python packages, install these system libraries:

Ubuntu/Debian:

```bash
```

```
# Install Python 3 and pip3
sudo apt install python3 python3-pip python3-dev

# Core development tools
sudo apt update
sudo apt install build-essential cmake pkg-config

# OpenCV dependencies
sudo apt install libopencv-dev python3-opencv
sudo apt install libjpeg-dev libtiff5-dev libpng-dev
sudo apt install libavcodec-dev libavformat-dev libswscale-dev
sudo apt install libgtk2.0-dev libcanberra-gtk-module

# Intel RealSense camera support
sudo apt install librealsense2-dev librealsense2-utils

# GPS and serial communication
sudo apt install gpsd gpsd-clients
sudo apt install libgps-dev

# I2C and SPI support
sudo apt install i2c-tools libi2c-dev
sudo apt install spi-tools

# Audio support
sudo apt install libasound2-dev portaudio19-dev

# Bluetooth support
sudo apt install libbluetooth-dev
```

Windows:

bash

```
# Install Visual Studio Build Tools
# Download from: https://visualstudio.microsoft.com/visual-cpp-build-tools/

# Intel RealSense SDK 2.0
# Download from: https://github.com/IntelRealSense/librealsense/releases

# OpenCV (install via pip should work, but for development):
# Download from: https://opencv.org/releases/
```

macOS:

bash

```
# Install Homebrew first: https://brew.sh
brew install cmake pkg-config
brew install opencv
brew install librealsense
brew install portaudio
```

Python Dependencies

```bash
pip install -r requirements.txt
```

Common dependencies include:

- `numpy` - Numerical computations
- `opencv-python` - Computer vision (for crop monitoring)
- `pyserial` - Serial communication with sensors
- `matplotlib` - Data visualization
- `pandas` - Data analysis and manipulation
- `pyrealsense2` - Intel RealSense camera interface

Installation

1. Clone or download the scripts to your local machine
2. Install Python 3.7+ if not already installed
3. Install required dependencies:

```bash
  pip install numpy opencv-python pyserial matplotlib pandas
```

4. Connect and configure your hardware sensors

Usage

Running Individual Scripts

Proximity Bridge Debug:

```bash
python 2_combo_proximity_bridge_debug.py
```

Row Following System:

```bash
python 8_row_following_system.py
```

Crop Monitoring System:

```bash
python 9_crop_monitoring_system.py
```

Configuration

Each script may require configuration for:

- Hardware sensor connections (COM ports, GPIO pins)
- Calibration parameters
- Field-specific settings
- Output file locations

Edit the configuration section at the top of each script before running.

Hardware Setup

Proximity Sensors

- Connect proximity sensors to designated ports
- Ensure proper power supply (typically 12V or 24V)
- Calibrate sensor range and sensitivity

GPS Module

- Connect GPS module for row following system
- Ensure clear sky view for accurate positioning
- Configure baud rate and communication protocol

Camera System

- Mount camera with clear field view for crop monitoring
- Adjust focus and lighting conditions
- Configure resolution and frame rate settings

Troubleshooting

Common Issues

Script won't start:

- Check Python version compatibility
- Verify all dependencies are installed
- Ensure hardware connections are secure

Sensor readings incorrect:

- Recalibrate sensors
- Check for electromagnetic interference
- Verify power supply stability

Poor row following performance:

- Check GPS signal quality
- Calibrate steering system
- Adjust detection sensitivity parameters

Safety Considerations

⚠️ **Important Safety Notes:**

- Always test systems in safe, controlled environments
- Maintain manual override capabilities
- Regular equipment maintenance and calibration
- Follow local agricultural safety regulations

Support and Maintenance

Regular Maintenance

- Clean sensors and cameras regularly
- Update calibration parameters seasonally
- Check connection integrity
- Backup configuration and data files

Updates

- Check for script updates regularly
- Test new versions in non-critical environments

- Document any custom modifications

License

This software is provided as-is for agricultural automation purposes. Please ensure compliance with local regulations regarding automated farming equipment.

Contributing

To contribute improvements or report issues:

1. Document the problem or enhancement
2. Test thoroughly in controlled environment
3. Submit detailed reports with system specifications

Last Updated: September 2025
Version: 1.0
Compatibility: Python 3.7+