irriDate: A Smart Solution for Palm Tree Irrigation and Health Monitoring

September 27, 2024

Table of Contents

Introduction

irriDate is an innovative application designed to revolutionize palm tree cultivation through smart irrigation, disease detection, and community engagement. By integrating advanced technologies like neural networks, computer vision, and IoT devices, irriDate addresses the critical challenges faced by palm tree growers in maintaining optimal soil moisture levels and monitoring tree health.

Problem Statement

Palm tree cultivation requires precise irrigation management and early detection of diseases to ensure healthy growth and high yield. Traditional methods of irrigation often lead to water wastage or under-irrigation, adversely affecting the trees. Additionally, manual disease detection is time-consuming and requires expert knowledge, which is not always readily available to farmers.

Solution Overview

irriDate offers a comprehensive solution comprising three main features:

1. **Smart Irrigation System**
2. **Palm Disease Analyzer**
3. **Community Platform**

By leveraging machine learning models, sensor data, and user-friendly interfaces, irriDate empowers farmers to make informed decisions, optimize resource usage, and engage with a community of peers.

Methodology and Technical Details

1. Smart Irrigation System

How It Works

- **Sensor Integration**: A NodeMCU microcontroller collects real-time data on soil moisture and temperature from sensors placed in the palm tree plantation.
- **Data Transmission**: The sensor data is transmitted to the application via Wi-Fi.
- **Neural Network Model**: A neural network processes the sensor data to predict the optimal irrigation schedule.
- **User Notification**: When irrigation is needed, the app sends a notification to the user's phone.

Technical Details

- **Hardware**: NodeMCU microcontroller connected to soil moisture and temperature sensors.
- **Software**:
  - **Python**: Used for developing the neural network model with TensorFlow.
  - **TensorFlow**: Provides the framework for building and training the neural network.
  - **Expo React Native**: Builds the mobile application interface.

Innovative Techniques

- **AI-Driven Predictions**: The neural network adapts to changing environmental conditions, improving irrigation efficiency over time.
- **Real-Time Monitoring**: Users can view live graphs of temperature and soil moisture within the app.

2. Palm Disease Analyzer

How It Works

- **Image Capture**: Users can take a photo of a palm tree or upload one from their album.
- **Image Classification**: The app classifies the image into one of ten categories:
  - **Eight Disease Classes**
    - **Potassium Deficiency**
    - **Manganese Deficiency**

        - **Magnesium Deficiency**
        - **Black Scorch**
        - **leaf spots**
        - **Fusarium Wilt**
        - **Rachis Blight**
        - **Parlatoria Blanchardi**
    - **Healthy**
    - **No Plant Detected**
- **Information and Treatment**: If a disease is detected, the app provides a link to resources on disease management and treatment.

Technical Details

- **Data Source**: Images were sourced from this dataset.
- **Machine Learning Model**:
    - **Convolutional Neural Network (CNN)**: Built using TensorFlow to handle image classification tasks.
- **Software**:
    - **Python**: Used for model development.
    - **Expo React Native**: Integrates the model into the mobile app.

Innovative Techniques

- **Transfer Learning**: Utilized pre-trained models to improve accuracy and reduce training time.
- **User-Friendly Interface**: Simplifies complex diagnostics into an accessible tool for farmers.

3. Community Platform

How It Works

- **User Interaction**: Users can create and share posts, allowing for knowledge exchange and community support.

- **Content Management**: Posts can include text, images, and links relevant to palm tree cultivation.

Technical Details

- **Backend Development**: Managed using a Node.js server to handle user data and posts.
- **Database**: Implemented using MongoDB for scalable data storage.
- **Software**:
  - **Expo React Native**: Provides the front-end interface for user interaction.

Innovative Techniques

- **Real-Time Updates**: Users receive instant updates on new posts.

Installation Instructions

Prerequisites

- **Hardware**:
  - NodeMCU microcontroller
  - Soil moisture sensor
  - Temperature sensor
- **Software**:
  - Python 3.x installed
  - esptool.py installed (pip install esptool)
  - ampy (Adafruit MicroPython Tool) installed (pip install adafruit-ampy)
  - Silicon Labs CP210x USB to UART Bridge VCP Drivers installed
  - Download for Windows
  - Download for macOS
  - Expo CLI installed globally (npm install -g expo-cli)
  - Git installed (for cloning repositories)
  - Mobile Device with the Expo Go app installed (available on iOS and Android)

Steps

1. Clone the Repository

Since the backend code, NodeMCU code, and the irriDate Expo app are in separate repositories, you need to clone each one individually.

a. Clone the irriDate App Repository

git clone https://github.com/yourusername/irriDate-expo-app.git

b. Clone the Backend Repository

git clone https://github.com/yourusername/irriDate-backend.git

c. Clone the NodeMCU Code Repository

git clone https://github.com/yourusername/irriDate-NodeMCU-code.git

2. Install Silicon Labs USB to UART Drivers

a. Download the Drivers

- Windows and macOS:
  - https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
- Linux:
  - The drivers are usually included in the Linux kernel. No action is required.

b. Install the Drivers

- Run the installer and follow the on-screen instructions.
- After installation, plug in your NodeMCU to ensure the drivers are correctly recognized.

3. Hardware Setup

a. Assemble the Hardware

- **Connect the Sensors to NodeMCU**:
  - Refer to the wiring picture provided in the irriDate-NodeMCU-code/hardware directory or documentation.

b. Flash MicroPython Firmware to NodeMCU

i. Download MicroPython Firmware

- Download the latest stable MicroPython firmware for ESP8266 from MicroPython Downloads.
- Choose the .bin file, e.g., esp8266-20210902-v1.17.bin.

ii. Erase Existing Firmware (Optional)

Open a terminal and run:

```
esptool.py --port /dev/tty.SLAB_USBtoUART erase_flash
```

Note: Replace /dev/tty.SLAB_USBtoUART with the correct port. On Windows, it might be something like COM3.

iii. Flash MicroPython Firmware

Run the following command:

```
esptool.py --port /dev/tty.SLAB_USBtoUART --baud 460800 write_flash
--flash_size=detect 0 esp8266-20210902-v1.17.bin
```

Note:

Replace /dev/tty.SLAB_USBtoUART with your port.

Replace esp8266-20210902-v1.17.bin with the name of the firmware file you downloaded.

c. Upload the MicroPython Code to NodeMCU

i. Navigate to the NodeMCU Code Directory

```
cd irriDate-NodeMCU-micropython
```

ii. Install Required Python Packages (On Your Computer)

- The NodeMCU will use MicroPython packages. Ensure any required modules are uploaded.

iii. Use ampy to Upload Files

Upload main.py:

```
ampy --port /dev/tty.SLAB_USBtoUART put main.py
```

iv. Verify Files on NodeMCU

List files on the NodeMCU to verify:

```
ampy --port /dev/tty.SLAB_USBtoUART ls
```

- You should see main.py and other uploaded files.

d. Configure Wi-Fi Setup via App

- **Note**: The NodeMCU code is set up to enter Wi-Fi Access Point (AP) mode on startup, allowing you to configure Wi-Fi credentials via the mobile app.
- **Verify Code Implementation**:
  - In main.py, ensure the code initializes Wi-Fi in AP mode and listens for configuration data.

e. Reboot NodeMCU

- Disconnect and reconnect the NodeMCU to power cycle it.
- The NodeMCU should now be running your MicroPython code.

4. Backend Setup (Flask)

a. Navigate to the Backend Directory

```
cd irriDate-backend-flask
```

b. Create a Virtual Environment (Optional but Recommended)

```
python -m venv venv
```

- Activate the virtual environment:

On Windows:

```
venv\Scripts\activate
```

On macOS/Linux:

```
source venv/bin/activate
```

c. Install Python Dependencies

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

d. Start the Flask Server

To start the server locally, simply run:

```
python app.py
```

The server should now be running on http://localhost:5000 or the port specified in your app.py.

e. Verify Backend is Running

- Open a web browser and navigate to http://localhost:5000/api/status.
- You should see a JSON response indicating the server is running.

5. Mobile App Setup

a. Navigate to the Mobile App Directory

```
cd irriDate-expo-app
```

b. Install Dependencies

```
npm install
```

c. Change the server IP

- Go to 'Backend_Server_IP.js' and change 'Backend_Server_IP' with the IP address where the Flask backend server is running.
    - For example, if running on your local machine:
        - Find your local IP address (e.g., 192.168.1.100).
        - Set 'Backend_Server_IP' to http://192.168.1.100:5000.

d. Start the Expo Development Server

Open a terminal and run:

```
expo start
```

This will open the Expo developer tools in your browser.

e. Run the App on Your Device

- **Using a Physical Device**:
    - Ensure your mobile device is connected to the same Wi-Fi network as your development machine.
    - Open the **Expo Go** app on your device.
    - Scan the QR code displayed in the Expo developer tools.

The app should load on your device or emulator

Conclusion

By following these steps, you can set up the **irriDate** application, including the hardware, backend server, and mobile app. This comprehensive solution enables you to monitor soil conditions, detect palm tree diseases, and engage with a community of palm tree growers, all through a user-friendly interface.