Intelligent Irrigation System for Low-cost Autonomous Water Control in Small-scale Agriculture

Deliverable D2.4

Advanced AI edge-processing framework for Plug-&-Sense smart irrigation system

Responsible Editor: WAZIUP e.V.

Version: 1.1

Date: January 2024

Contributors table

DOCUMENT SECTION	AUTHOR(S)
SECTION 1	A. Rahim
SECTION 2	F. Markwordt
SECTION 3	F. Markwordt
SECTION 4	A. Rahim

Document revision history

Version	Date	Changes
V1.1	Jan 17th, 2024	Public release
V1.0	Dec 22nd, 2023	First draft version for internal approval
V0.1	Dec 1th, 2023	First release for review

Executive Summary

In this report, Deliverable D2.4 delves into the intricacies of the INTEL-IRRIS Edge-AI application designed specifically for the WaziGate. It goes beyond a mere overview by offering detailed insights into the underlying hardware architecture. Furthermore, the report elucidates the step-by-step process of developing machine learning models, shedding light on their practical deployment.

The document serves as a valuable resource, not only outlining the technical aspects but also emphasising the real-world applications of these models. By elucidating how the generated results are harnessed, the report illustrates the tangible benefits for farmers. It is a comprehensive exploration that aims to provide a thorough understanding of the integration of advanced technologies in agricultural practices.

Table of Contents

1. INTRODUCTION

The INTEL-IRRIS Edge-AI irrigation system introduces an innovative and cost-effective IoT and AI-based irrigation system, leveraging the Edge computing architecture. This report revolves around the development of an advanced irrigation system that adeptly operates at the edge level. Integral to this system is an inference engine that facilitates the precise calculation of optimal water quantities and irrigation schedules.

This report provides an extensive exploration of the AI edge-processing framework architecture and design, including the Edge gateway, Edge application, embedded software and AI model.

2. The INTEL-IRRIS Edge-AI irrigation system

2.1 WaziGate

WaziGate is a versatile IoT LoRa Gateway designed to meet diverse remote IoT needs. It can connect up to 100 IoT sensors and actuator nodes using a robust LoRa radio network, making it suitable for various applications like weather monitoring, soil analysis, GPS tracking and much more.

What sets WaziGate apart is its edge capacity, allowing you to host applications directly within the gateway for a responsive and tailored environment. It not only gathers data but also enables control of actuators such as electro-valves.

Flexibility is a key feature, with the ability to host applications within the gateway and enhanced connectivity through WiFi. It boasts a long LoRa communication range of up to 10-12 kilometers and establishes a WiFi hotspot for device connectivity.

WaziGate adapts to different connectivity options, supporting WiFi, 3G, and Ethernet connections, ensuring continuous connectivity regardless of infrastructure availability. It's energy-efficient, optimizing performance while conserving power.

Automation capabilities streamline IoT network operations, enabling devices to communicate intelligently. Remote management allows real-time adjustments, even without internet access, thanks to its embedded database and web-based visualization module.

In summary, WaziGate is a powerful IoT LoRa Gateway that combines connectivity, control, data access, and advanced features like edge hosting, extended communication range, low power consumption, and remote management. Its versatility and resilience in limited connectivity situations make it an essential asset in the IoT landscape.

2.2 WaziApp

WaziApps are applications developed specifically for the Wazigate platform, which is an IoT LoRa Gateway designed for remote IoT applications. These apps are designed to enhance the capabilities of the Wazigate by extending its functionality to cater to various use cases. The core functionality of WaziApps revolves around leveraging the microservice architecture of Wazigate to create independent, isolated applications that can run on the gateway.

Key aspects of WaziApps' core functionality include:

- **Microservice Architecture:** WaziApps operate using a microservice architecture. Each app functions as an independent microservice, running in its own isolated container. This architecture enhances development, maintenance, and scalability by allowing apps to operate autonomously.
- **Docker Containers:** WaziApps are encapsulated within Docker containers. This encapsulation ensures that each app runs independently of other apps, preventing potential conflicts or interference. These containers are conveniently accessible through the Wazigate's user interface, allowing users to directly retrieve them from Docker Hub. This streamlined process eliminates the need for manual configuration and enables users to easily deploy the desired containers for their WaziApps. By offering this direct integration with Docker Hub, Wazigate simplifies the deployment of applications and enhances the user experience.
- **Custom Development:** WaziApps offer developers the flexibility to create custom applications to suit their specific needs. Whether it's a weather station, soil monitoring, GPS application, or any other IoT-related functionality, WaziApps can be tailored accordingly.

- **Multi-Language Support:** WaziApps can be developed using various programming languages, including Python, GoLang, and Javascript. This enables developers to work with the language they are most comfortable with, promoting productivity and creativity.
- **API Creation:** WaziApps allow developers to create APIs to interact with the app's functionality. Also the WaziGates API enables communication between the app and other components of the system, facilitating data exchange and control.
- **Local Hosting:** WaziApps can host their applications directly on the Wazigate. This means that the gateway can operate as a server for the developed applications, providing a self-contained environment for running and managing apps.
- **Remote Management:** Apps running on the Wazigate can be managed remotely. This enables developers to update, maintain, and monitor their applications from a distance.
- **Data Visualization:** WaziApps can provide data visualization capabilities, allowing users to visualize collected data through web-based interfaces. This visualization aids in data analysis and decision-making.
- **Extensibility:** As the possibilities are endless, WaziApps can be extended to encompass various functionalities and use cases. Whether it's gathering environmental data, controlling actuators, or other IoT-related tasks, WaziApps can adapt to different scenarios.

In summary, WaziApps uses Wazigate microservice architecture to enable developers to create customized, independent applications for a wide range of IoT applications. This versatility empowers users to develop solutions that cater to their specific requirements, enhancing the overall capabilities of the Wazigate platform.

2.3 INTEL-IRRIS soil sensor device software architecture

At the core of the INTEL-IRRIS soil sensor device’s functionality stands its embedded software, a pivotal element that orchestrates a series of essential tasks. The embedded software undertakes the following key responsibilities:

1. Measurement Execution:

The software efficiently manages the process of measuring soil water tension through the device’s attached water tension sensor. It controls sensor activation, oversees data acquisition, and oversees the conversion of raw data into meaningful measurements.

2. Calibration for Accuracy:

Following the acquisition of raw data, the software applies calibration algorithms to transform the measurements into calibrated centibar values. This calibration process ensures the accuracy of soil tension readings, providing dependable insights for agricultural analysis.

3. LoRa Data Transmission:

Once calibrated centibar values are determined, the software leverages LoRa communication technology for wireless data transmission to a LoRa Gateway. It formats the data into LoRa-compatible packets and manages the transmission process to ensure efficient and reliable data transfer.

4. Efficient Energy Management:

The software incorporates energy management strategies to optimize the device’s battery life. It monitors the battery voltage level and implements energy-saving measures to extend the operational duration.

5. Low Battery Adaptation:

When the battery level reaches a low threshold, the software adapts by adjusting the transmission interval. It reduces the frequency of data transmission to conserve energy and extend the device's operational capability, ensuring functionality even under limited power conditions.

By seamlessly handling these tasks, the embedded software within the soil device plays a pivotal role in enabling accurate measurements, dependable data transmission, and effective energy management. It ensures that the device operates optimally, providing valuable soil water tension data that contributes to informed agricultural monitoring and decision-making processes.

The microcontroller code is written in the C programming language. For easy access and collaboration, we have created a GitHub repository where you can find the code using the following link: PRIMA INTEL-IRRIS GitHub repository

2.4 INTEL-IRRIS Edge-AI WaziApp

The initial version of the INTEL-IRRIS Edge-AI WaziApp application for the WaziGate is focused on providing farmers with actionable guidance and predictions for optimal watering practices within a reasonable forecast horizon. This predictive functionality revolves around the **farmer's pre-set soil tension threshold.**

The application integrates various crucial factors into its decision-making process. It begins by collecting data from the soil sensor devices, encompassing soil moisture and temperature readings. From these readings, **the volumetric water content is derived**. This calculation can be customized based on the specific soil type or through the use of a tailored soil water retention curve. Additional features are also being engineered, including the status of the pump and various moving averages derived from the sensor data.

Supplementary data is fetched through an API, providing **historical weather information** for the farm's GPS location. This data encompasses temperature, humidity, rainfall, cloud cover, shortwave radiation, wind speed, wind direction, soil temperature, soil moisture, and evapotranspiration (Et0). **Forecasts for this meteorological data are also accessible**, contributing to the application's predictive capabilities.

Following an initial warm-up period, during which the farmer has the option to train the model with pump state data (later automated detection can be implemented), the system becomes proficient in making predictions when specific soil moisture thresholds are reached. This prediction process takes into account various environmental variables.

The model undergoes periodic training cycles, during which multiple models are compared, tuned, and evaluated using ensemble techniques. The best-performing model is selected based on its performance on an evaluation set that was not used during training. This iterative process ensures that the predictions stay updated and incorporate the latest changes in the environment.

3. IMPLEMENTATION

3.1 Software components

Regarding the INTEL-IRRIS Edge-AI WaziApp irrigation application, the backend of this application is developed using Python, harnessing a suite of powerful packages to achieve its functionality. Here's a brief overview of the key packages utilized:

- **PyCaret:** PyCaret is a versatile machine learning library that streamlines the process of model training and deployment. It offers automation for various machine learning tasks, making it an efficient choice for building predictive models in a user-friendly manner.
- **Scikit-learn (sklearn):** Scikit-learn is a renowned machine learning library that provides a comprehensive range of tools for data analysis, modeling, and predictive analysis. It offers an array of algorithms for classification, regression, clustering, and more.
- **XGBoost:** XGBoost is a popular gradient boosting library that excels in boosting and ensemble learning. It's particularly effective for improving the performance of machine learning models, making it a valuable addition to the application's toolkit.
- **Pandas:** Pandas is a fundamental library for data manipulation and analysis. It offers data structures and functions that simplify working with structured data, enabling efficient data preprocessing and transformation.
- **NumPy:** NumPy is a fundamental package for numerical computing in Python. It provides support for arrays, matrices, and mathematical functions, making it essential for efficient handling of large datasets.
- **Matplotlib:** Matplotlib is a widely used plotting library that facilitates the creation of various types of visualizations, aiding in data exploration and presentation.
- **Subprocess:** The Subprocess module allows the application to spawn new processes, connect to their input/output/error pipes, and obtain return codes. It's useful for executing external commands and processes from within the Python code.
- **JSON:** JSON (JavaScript Object Notation) is a lightweight data interchange format. It's employed for encoding and decoding structured data, enabling seamless communication between different components of the application.
- **Datetime:** The Datetime module provides classes for manipulating dates and times. It's crucial for managing timestamps, durations, and other temporal data within the application.
- **Missingno:** Missingno is a visualization library tailored for identifying and visualizing missing data in datasets. This aids in understanding data completeness and quality.

On the frontend side, the INTEL-IRRIS Edge-AI WaziApp irrigation application is implemented as a **Flask web application**. This micro web framework enables the creation of interactive and dynamic web interfaces. The application's frontend will utilize **Apex Charts**, a visualization library, to render a variety of informative charts and graphs, enhancing the user experience. Configuration settings will be stored in a **JSON file**, allowing users to easily customize and tailor the application's behavior to their specific requirements.

This integration of Python packages and frontend technologies results in a robust and user-centric smart irrigation application, designed to provide accurate predictions and guidance for optimal irrigation practices.

3.2 Next steps and future prototypes

In the roadmap for future prototypes of the INTEL-IRRIS Edge-AI WaziApp, the aim is to achieve a comprehensive level of automation for the irrigation system, streamlining the process and reducing manual intervention. This progression involves incorporating key components to enhance system functionality:

WaziAct Integration: The addition of a WaziAct module serves as a pivotal component to automate the irrigation process. The WaziAct module controls the pump, ensuring efficient and timely water delivery to the plants. This integration marks a significant step toward achieving a fully automated irrigation system.

Flow Meter Integration: The inclusion of a flow meter introduces the capability to accurately measure the amount of water delivered to the plants. This data provides crucial insights into water consumption and usage, aiding in optimizing irrigation strategies and resource management.

Complete Automation: With the combination of the WaziAct module and the flow meter, the irrigation system can be fully automated. The system will autonomously monitor soil moisture levels, weather forecasts, and other relevant factors to determine the optimal irrigation schedule. This minimizes the need for constant manual oversight and intervention by farmers.

Visual Detection via Camera: To further enhance the automation, the integration of a camera system could be considered. This camera could be strategically positioned to oversee specific plants. By leveraging object detection techniques on a real-time video stream, the system can identify and assess plant conditions. This enables remote monitoring and eliminates the need for physical presence for visual inspections.

Plant Stress Detection: Another potential advancement is the training of models to detect plant stress indicators through image analysis. By incorporating machine learning algorithms, the system can identify signs of plant stress, such as discoloration or wilting. These findings can serve as additional inputs for the predictive model, allowing it to refine its irrigation recommendations based on real-time plant health data.

Continuous Improvement: The iterative nature of the system enables continuous improvement. As the system gathers more data over time, it can refine its predictions and recommendations. By incorporating feedback loops and machine learning techniques, the overall system becomes more accurate and tailored to the specific needs of the plants.

By integrating these components and capabilities, the future iterations of the INTEL-IRRIS Edge-AI WaziApp aim to create a highly automated and intelligent irrigation system. This evolution not only optimizes water usage but also empowers farmers with valuable insights and control over their agricultural practices, paving the way for increased efficiency and sustainability in irrigation management.

4. CONCLUSION

This report detailed the architecture and implementation of the AI edge-processing framework for Plug-&-Sense smart irrigation system. We presented the "WaziGate" Edge gateway, able to host AI-edge applications such as the Intel-Irris Edge application. We also detailed the software deployed in the sensing devices, and AI model.