Automated Rain-Responsive Clothesline System

1. **Project Aim:**

Drying clothes outside is a common practice in households without electric dryers.

However, sudden rainfall can soak freshly washed clothes, leading to inconvenience,

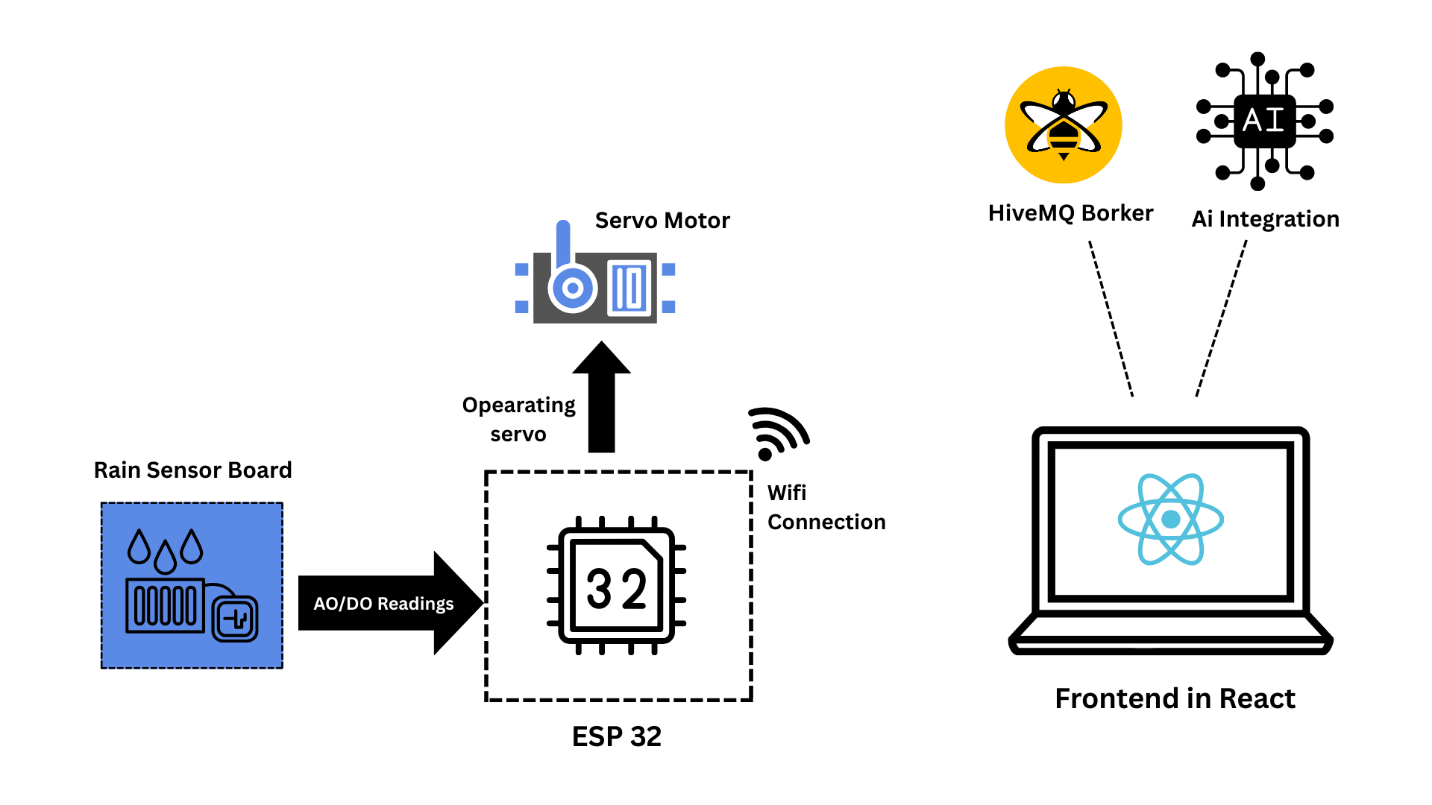
rewashing, and delays. This project proposes a smart solution using an ESP32-based

rain-detecting system that not only alerts users but also automatically moves the

clothesline under a protective shed using a servo. Once the rain stops, the system

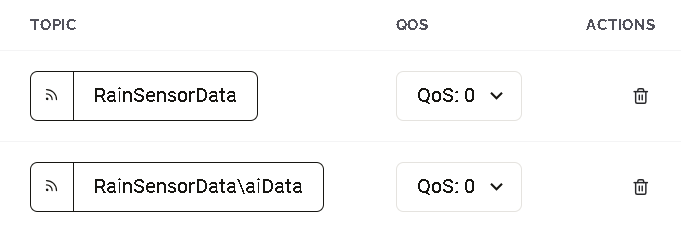
will return the clothesline to the open air for continued drying. Moreover, We will provide rain intensity and Ai based predications related to the duration of rain, so that the user can decide whether to get the clothes of the linings or not.

1. **Block Diagram :**



1. **MQTT Topics:**

We are using HiveMQ Cloud Broker , Following are our topics we are using to transferring data for user and Ai



1. **Detailed Cloud Ai Model:**

**Model Type**

* **Random Forest Regressor** (from sklearn.ensemble.RandomForestRegressor)
* Used for regression (predicting continuous values)

**Model Training Details**

* **Features used**:
  + intensity (rain intensity, 0-100%)
  + current\_duration (how long it has been raining so far in minutes)
* **Target variable**:
  + total\_duration (total rain duration in minutes)
* **Data**:
  + Synthetic/mock dataset generated with 1000 samples
  + Intensity mapped to durations with some noise added to simulate real data
* **Train/Test split**: 80% train, 20% test
* **Hyperparameters**:
  + n\_estimators=100 (number of trees)
  + max\_depth=10
  + random\_state=42

**Model Performance**

* **Training R² score:** Approximately 0.95 (or similar, based on print statement — actual value shown when you run)
* **Testing R² score:** Approximately 0.93 (or similar)

These values indicate the model fits the training data well and generalizes reasonably on test data.

**Usage**

* The model predicts the **total rain duration** based on current rain intensity and duration so far.
* The API subtracts current duration from predicted total to give **predicted remaining rain duration**.
* If the model file is missing, a simple rule-based fallback is used for prediction.

**Deployment**

* The model is loaded in a Flask API (flash\_api.py) which exposes a /ai-predict POST endpoint.
* Input JSON requires:
  + rain\_intensity (percentage)
  + duration (string like "1m 19s", parsed into minutes)
* Returns:
  + predicted\_remaining\_minutes (float)
  + confidence (based on duration length)
  + Echoed input intensity and duration

1. **User guide for front-end**

As our app is home based and it’s a model. We have integrated a Screen containing all the Displayed data in clear understanding

Accessing the Interface

* Open the web application in a browser.
* The main screen shows a card with rain status, intensity, duration, and predictions.

**Features and Usage**

**1. Real-Time Rain Status**

* The interface shows whether it is currently **Raining** or **Dry**.
* Status is updated automatically every 10 seconds.

**2. Rain Intensity**

* Displays current rain intensity as a percentage.
* Visualized with a progress bar showing the level of rain intensity.

**3. Rain Duration**

* Shows how long the rain has been ongoing in a readable format (e.g., "1m 19s").
* Updated live alongside intensity.