



School of Computer Sciences
Semester 1, 2024/2025
CPC357 IoT Architecture and Smart Applications

Project
Technical Documentation

TirAir:
An IoT-Based Rainwater Harvesting and
Roof Cooling System

MAISARAH QISTINA BINTI MEOR SHA'AZIZI (155346)
&
RAJIHAH BINTI MUHD ROSYDI (155442)

GitHub Repo: <https://github.com/rajihahr/TirAir>

TABLE OF CONTENTS

TABLE OF CONTENTS	ii
1 INTRODUCTION	3
1.1. Background	3
1.2. Problem Statement	4
1.3. System Objectives.....	4
1.4. System Overview.....	5
1.5. SDG Alignment	6
2 SYSTEM ARCHITECTURE.....	7
2.1. Architecture Layers	7
2.2. Hardware Setup.....	10
2.3. Software Setup (Coding).....	12
2.4. Final Product	20
3 CONCLUSION	23
REFERENCES	24

1 INTRODUCTION

1.1. Background

In traditional rural Malay communities, zinc roofing remains a popular choice for housing due to its affordability, durability, and ease of construction [1]. These characteristics make it an optimal choice for low-income households. However, this type of roofing comes with a significant drawback which is poor thermal comfort. Zinc is a highly efficient conductor of heat that absorbs solar radiation during the day and transfers it into the living space below. This conduction process causes the inside of the house to become excessively hot, particularly in the afternoons, when temperatures increase [2]. The lack of proper ventilation in many of these homes further exacerbates the issue as the heat gathers and accumulates within the living space. This problem is further caused by global warming which has already raised ambient temperatures beyond normal levels. Traditional solutions such as ceiling fans only recirculate the hot air [3] while air conditioning systems are financially out of reach for most rural households.

At the same time, Malaysia experiences significant seasonal rainfall in states such as Kelantan, Terengganu and Pahang create another critical issue in rural areas which is water shortage [4]. According to Eco-Business [5], Malaysia is projected to encounter a water crisis by 2030 due to increasing demand. Between 2018 and 2022, the country lost an estimated amount of USD 1.7 billion from wasted water. The combination of climate-related challenges and the perception of water being a low-cost resource increases the issue of water scarcity. Rainwater harvesting (RWH) systems have proven to be an optimal method for mitigating water shortages globally [6] and can be particularly beneficial in areas that face intense heat and periodic droughts. By integrating rainwater harvesting with an effective cooling mechanism for zinc roofs, there is potential to create a sustainable and low-cost solution that improves thermal comfort while conserving water resources for future use. This system leverages the abundance of natural rainfall to address two pressing issues faced by rural Malay communities.

1.2. Problem Statement

Rural Malay households face two interconnected challenges that significantly affect their quality of life. The first issue is poor thermal comfort in homes with zinc roofing, which absorbs and conducts heat efficiently despite its affordability and durability [1,2]. The situation is worsened by the lack of proper ventilation in many homes that cause heat to accumulate. Global warming further increases this issue by the rise of temperatures and making traditional solutions such as ceiling fans, ineffective. Air conditioning systems are considered more effective [3], but it remains unaffordable for most low-income households since the focus is rural households.

The second issue is water shortage which is a growing concern in Malaysia despite significant seasonal rainfall in states like Kelantan, Terengganu, and Pahang. Experts project a water crisis by 2030 due to climate-related pressures and the undervaluation of water as a resource [5]. Rural areas are particularly vulnerable to these challenges as they experience periods of heavy rain followed by water shortages during drier months [4].

To address these issues, an integrated system combining rainwater harvesting and roof cooling presents an innovative and cost-effective solution. By utilizing the source of rainfall to cool zinc roofs and conserve water resources, this system has the potential to improve thermal comfort and mitigate water shortages to further enhance the living conditions of rural Malay communities.

1.3. System Objectives

This system aims to achieve the following objectives:

- To develop an automated roof cooling system that utilizes harvested rainwater to reduce the indoor temperature of homes with zinc roofs.
- To implement a rainwater harvesting system to collect and store rainwater for efficient utilization for roof cooling and minimizing water wastage in rural communities.
- To design an affordable and low-maintenance integrated system that addresses thermal discomfort and water shortage.

1.4. System Overview

TirAir is a combination of “Tirai” which is the Malay word for curtain, and “Air” which means water. The name reflects the system’s dual function that provides a protective shield or "curtain" for the home by cooling zinc roofs with rainwater and using water as a natural resource to enhance thermal comfort. Zinc is a highly conductive metal that readily absorbs heat from sunlight leading to the excessive heat inside the house [9].

This project is designed to efficiently utilize rainwater for cooling zinc roofs of houses during hot weather to regulate the house temperature. The system integrates various sensors and actuators to trigger the cooling process based on predefined conditions. Additionally, a dashboard is implemented to display the collected data and allow users to manage the system's actions. The prototype model is designed to replicate a house with a zinc roof that incorporates a rainwater harvesting tank which supplies water to the sprinkler system for cooling. The process of the overall system is explained as follows:

1. Rainwater Harvesting Activation:

- The rain sensor module detects the presence of rain when it starts raining.
- The SG90 micro servo will be activated upon detection to open the water tank lid to allow rainwater to fill up the tank for storage.
- Once the rain stops, the rain sensor module detects the absence of rain signalling the micro servo to close the water tank lid.

2. Temperature Sensing for Cooling:

- The system monitors the temperature on top of the roof using a DHT11 sensor.
- If the temperature rises above the predefined threshold hence indicating hot weather, the system will trigger the solenoid valve to open.
- After the temperature decreases and regulates, the solenoid valve will close.

3. Water Flow for Cooling:

- The stored rainwater from the tank is released to the sprinkler system when the solenoid valve opens.
- The water sprinkles over the zinc roof which cools it down by evaporative cooling. It reduces the heat absorbed by the roof and ultimately lowering the overall temperature inside the house [1].

1.5. SDG Alignment

This project was constructed to align with two United Nations Sustainable Development Goals (SDGs) as outlined below:

- **SDG 6: Clean Water and Sanitation [7]**

The project addresses sustainable water management by incorporating a rainwater harvesting system for roof cooling. This innovative approach helps reduce water wastage and alleviate water scarcity. The system ensures that treated water is reserved for essential uses such as drinking and utilizes rainwater for tasks like cooling.

- **SDG 12: Responsible Consumption and Production [8]**

The project promotes the responsible use of natural resources by efficiently utilizing harvested rainwater as a readily available resource in regions like Malaysia. This approach reduces reliance on conventional water sources and encouraging environmentally responsible practices.

2 SYSTEM ARCHITECTURE

2.1. Architecture Layers

The architecture of TirAir organized into four key layers where each layer serves a distinct purpose for efficient operation and user interaction.

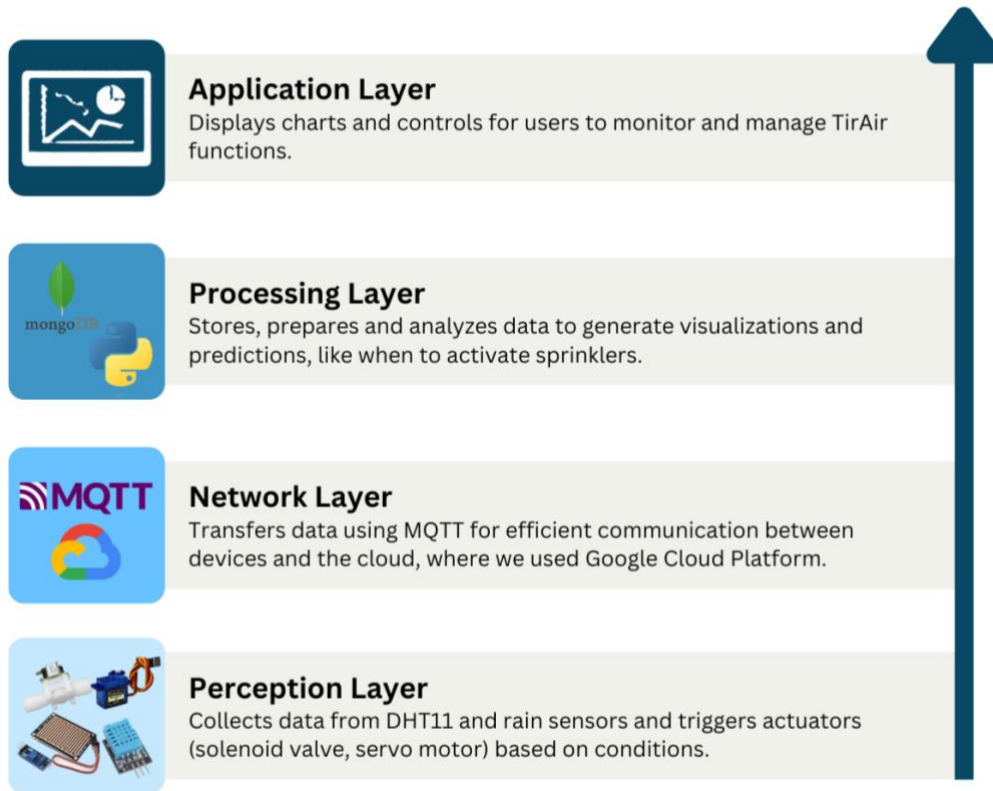


Figure 2-1: TirAir Architecture Diagram

i) Perception Layer

The purpose of the perception layer is to collect real-time environmental data from sensors and control the system's physical components which are actuators, in this case. In TirAir, this layer plays a crucial role in monitoring the temperature, humidity, and rainfall, which are key factors in managing thermal comfort and water usage. The sensors used are DHT11 and rain sensor whereas the actuators used are solenoid valve and servo motor. The DHT11 sensor measures the temperature and humidity of the environment. When the DHT11 sensor detects that the temperature or humidity has crossed a threshold of 25C, the system will send a signal to activate the solenoid valve. This opens the valve that allows water to flow to the sprinklers which irrigates the roof and cools it down, thereby addressing the thermal discomfort. The solenoid valve acts as an actuator

that physically responds to the data provided by the DHT11 sensor so that the sprinklers are only turned on when necessary. The rain sensor is responsible for detecting rainfall in the environment. It plays a role in rainwater harvesting by signaling when it is raining to collect rainwater for storage. When rain is detected from the sensor, the servo motor activates to open the water tank lid which allows the rainwater to flow into the tank and be harvested.

ii) Network Layer

The purpose of the network layer is to facilitate communication between the sensors, actuators, and the data processing components of the system. It ensures that the collected data is transmitted to the backend cloud which in this case uses Google Cloud Platform for further data processing. MQTT is used here because it is lightweight and optimized for IoT environments that allows TirAir to send small packets of data from the sensors in the previous layer with low bandwidth usage. The ESP32 microcontroller acts as a publisher that sends sensor readings (e.g., temperature, humidity, and rain status) to a specific topic on an MQTT broker. A backend server which is a Virtual Machine instance on GCP subscribes to the topic and receives the data in real time.

iii) Processing Layer

The processing layer is responsible for storing, organizing, and preparing the raw data collected from the sensors for insights and visualization. Once the ESP32 sends the sensor data (e.g., temperature, humidity, and rain detection) to the backend via MQTT, this layer processes the data to make it usable in the next layer. It performs tasks like cleaning the data and preparing it for storage. The processed data is then stored in a MongoDB database. This layer is essential as it bridges the gap between raw sensor data and actionable insights used in the next application layer.

iv) Application Layer

The application layer is the user-facing component of TirAir that is responsible for presenting data and system controls in an intuitive and accessible way. This layer visualizes the processed data stored in

MongoDB through various charts and metrics that allows users to monitor and interact with the system effectively. Key visualizations include time series charts for temperature and humidity, predictions for the next sprinkler activation and current environmental conditions, such as whether it is raining. Additionally, the application layer provides manual control buttons to turn the water tank lid and sprinkler on or off so users can directly control the system when needed. The primary purpose of this layer is to enable users to understand the system's status and make informed decisions. For instance, the time series charts help users track trends over time, while the rain status and sprinkler predictions allow proactive water management. By combining real-time updates, historical data, and manual controls, the application layer ensures that users have a clear and comprehensive view of the TirAir system so it's easy to manage and optimize operations for thermal comfort and water harvesting.

Each of these layers is integral to TirAir. The sensors and actuators provide real-time data collection and environmental control, while the network layer ensures seamless communication using MQTT. The data processing layer supports predictive capabilities and analytics, and the application layer delivers an intuitive interface for user interaction.

2.2. Hardware Setup

To set up TirAir system, the components used are:

- Maker Feather AIoT S3 board
- Breadboard
- Rain Sensor Module
- SG90 Micro Servo
- DHT11 Sensor
- OctoCoupler Relay Module
- Liquid Solenoid Valve
- 12V Power Adapter
- DC Power Jack
- Qwiic Cable x 2
- Male to Male Jumper Wire x 11

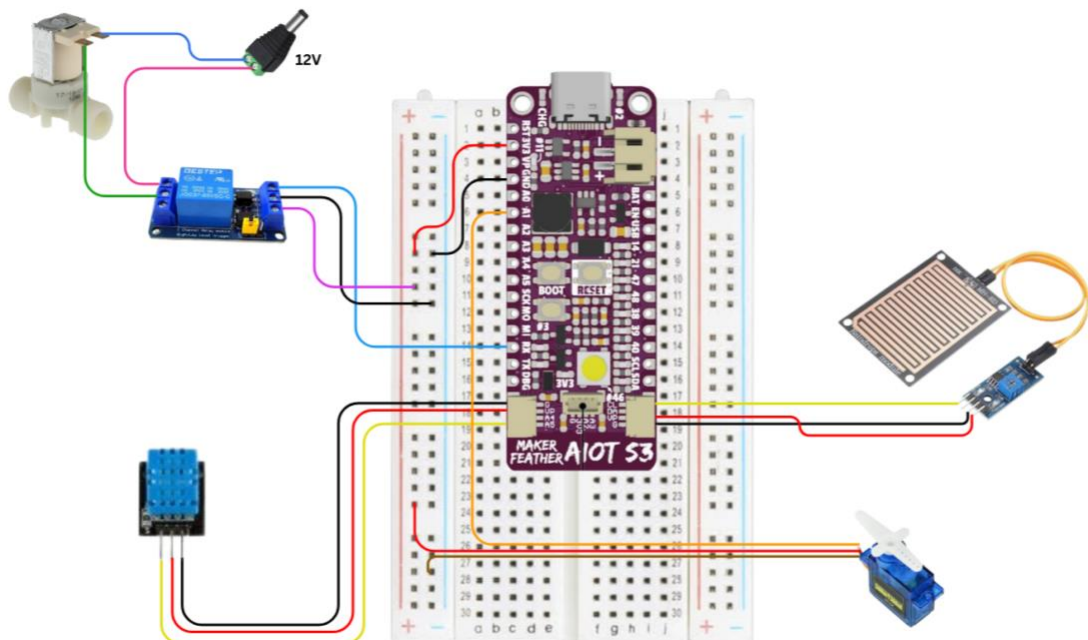


Figure 2-2: TirAir Circuit Diagram

To further understand the hardware setup, the liquid solenoid valve is connected to a PVC pipe that is connected to water sprinklers. The valve controls the water flow to activate the sprinklers. The servo motor is attached to the water tank lid that has the rain sensor connected on top of it so that it

harvests or stores water. When rain is detected, the servo motor rotates to open the lid and closes when rain is not detected (full prototype model in Figure 2-3).

Figure 2-2 illustrates the circuit diagram of TirAir system. The connections for these components are as follows:

- Maker **3V3** -> Breadboard **Positive Power Rail** (left)
- Maker **GND** -> Breadboard **Negative Power Rail** (left)
- OctoCoupler Relay **In1** -> Maker **Pin D16**
- OctoCoupler Relay **DC+** -> Breadboard **Positive Power Rail** (left)
- OctoCoupler Relay **DC-** -> Breadboard **Negative Power Rail** (left)
- 12V Power Adapter **Positive Terminal** -> OctoCoupler Relay **NO**
- Liquid Solenoid Valve Terminal -> 12V Power Adapter **Negative Terminal**
- Liquid Solenoid Valve Terminal -> OctoCoupler Relay **COM**
- DHT11 **SIG** -> Maker Port (left) **Pin D7**
- DHT11 **VCC** -> Maker Port (left) **VP**
- DHT11 **GND** -> Maker Port (left) **G**
- Rain Sensor Module **AO** -> Maker Port (right) **Pin D41**
- Rain Sensor Module **VCC** -> Maker Port (right) **VP**
- Rain Sensor Module **GND** -> Maker Port (right) **G**
- SG90 Micro Servo **SIG** -> Maker **Pin D9**
- SG90 Micro Servo **VCC** -> Breadboard **Positive Power Rail** (left)
- SG90 Micro Servo **GND** -> Breadboard **Negative Power Rail** (left)

2.3. Software Setup (Coding)

Note: The full code can be accessed via the GitHub repository linked on the cover page.

The general workflow for the software part includes four main steps as described below:

i) Set Up ESP32 for Data Collection (`sensor_reading.ino`)

- Initialize necessary libraries and define pins for sensors and actuators.
- Connect to WiFi using specified SSID and password and configure the MQTT client with **Port 1883** to publish sensor data and listen for control commands through topic **cpc357**. All MQTT communications use the **PubSubClient** library

```
const char* WIFI_SSID = "";
const char* WIFI_PASSWORD = "";
const char* MQTT_SERVER = "34.29.205.132";
const char* MQTT_TOPIC = "cpc357";           // MQTT topic for sensor data
const char* CONTROL_TOPIC = "cpc357/control"; // MQTT topic for control commands
const int MQTT_PORT = 1883;                 // Non-TLS communication port
```

- Read temperature, humidity, and rain status from connected sensors.

```
void loop() {
    // Read temperature from DHT11 sensor
    float t = dht.readTemperature();
    float h = dht.readHumidity(); // Read humidity

    // Read rain sensor
    int rain_sensor_value = digitalRead(RAIN_SENSOR_PIN);

    if (isnan(t) || isnan(h)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }
}
```

- Based on the sensor readings, control the servo motor for the water tank lid and the valve for the sprinkler system. If rain is detected, the

water tank lid handled by the servo motor turns **180 degrees**. If temperature is above **25 Celsius**, the water sprinkler handled by the solenoid valve is opened.

```
if (rain_sensor_value == LOW && !manualControl) { // Assuming LOW means rain detected
    if (rain_start_time == 0) {
        rain_start_time = millis();
    }

    if (millis() - rain_start_time > rain_duration_threshold) {
        tap_servo.write(180); // Move servo to 180 degrees when rain detected
        isRaining = true;
    }
}

if (t >= TEMP_THRESHOLD) {
    digitalWrite(RELAY_PIN, HIGH); // Open valve when temperature above 25°C
    valveOpened = true;
} else {
    digitalWrite(RELAY_PIN, LOW);
    valveOpened = false;
}
```

- Manual control handling through MQTT

```
void callback(char* topic, byte* payload, unsigned int length) {
    if (String(topic) == CONTROL_TOPIC) {
        if (message == "open_lid") {
            tap_servo.write(180);
            manualControl = true;
        } else if (message == "close_lid") {
            tap_servo.write(0);
            manualControl = true;
        } else if (message == "open_sprinklers") {
            digitalWrite(RELAY_PIN, HIGH);
            valveOpened = true;
        } else if (message == "close_sprinklers") {
            digitalWrite(RELAY_PIN, LOW);
            valveOpened = false;
        }
    }
}
```

- Publish the sensor data to the MQTT broker at regular intervals.

```
sprintf(buffer, "Temperature: %.2f degree Celsius, Humidity: %.2f %, Raining: %s, Valve: %s",
        t, h, isRaining ? "Yes" : "No", valveOpened ? "Opened" : "Closed");
client.publish(MQTT_TOPIC, buffer);
```

ii) Set Up the Data Ingestion into MongoDB (store_mongo.py)

MongoDB is a NoSQL database known for its flexibility and storing data in a JSON-like format. It is an excellent choice for IoT applications that produces vast amounts of data hence needing efficient storage. By integrating MQTT, data can be transmitted between clients (data from IoT sensors) and then saved into MongoDB upon reception. Below are the key steps to do so:

- Connect to a local MongoDB server at **localhost:27017**.

```
mongo_client = pymongo.MongoClient("mongodb://localhost:27017/")
```

- Specify the database (**TirAir**) and collection (**sensor_data**) for storing data.

```
db = mongo_client["TirAir"]
collection = db["sensor_data"]
```

- Connect to an MQTT broker at the specified IP address on **Port 1883** with a 60-second keepalive for it to subscribe to the **cpc357** topic.

```
# Connect to the MQTT broker
client.connect(mqtt_broker_address, 1883, 60)
```

- **on_connect** is called when the client connects to the MQTT broker. It subscribes to the specified topic.

```
def on_connect(client, userdata, flags, reason_code, properties):
    if reason_code == 0:
        print("Successfully connected")
        client.subscribe(mqtt_topic)
    else:
        print("Connection failed with code:", reason_code)
```

- **on_message** is called when a message is received on the subscribed topic. It processes the received data, converts it to the appropriate format and inserts it into MongoDB.

```

def on_message(client, userdata, message):
    payload = message.payload.decode("utf-8")
    print(f"Received message: {payload}")

    # Parse the message
    data = {}
    parts = payload.split(", ")
    for part in parts:
        key, value = part.split(": ")
        data[key.lower()] = value

    # Convert MQTT timestamp to datetime
    timestamp = datetime.now(timezone.utc)
    datetime_obj = timestamp.strftime("%Y-%m-%dT%H:%M:%S.%fZ")

    # Process the payload and insert into MongoDB with proper timestamp
    document = {
        "timestamp": datetime_obj,
        "temperature": float(data["temperature"].split(" ")[0]),
        "humidity": float(data["humidity"].split(" ")[0]),
        "raining": data["raining"],
        "valve": data["valve"]
    }

    collection.insert_one(document)
    print("Data inserted into MongoDB")

```

iii) Train the Machine Learning Model (train_model.py)

The machine learning model predicts the time until the next sprinkler activation based on temperature data to optimize water usage and automate the irrigation process.

- Retrieve historical sensor data from the MongoDB database (***TirAir***) and collection (***sensor_data***).

```

def fetch_data():
    data = list(collection.find())
    for doc in data:
        doc['_id'] = str(doc['_id']) # Convert ObjectId to string
    return data

```

- Compute necessary features (e.g., temperature, time difference between readings) and target variables (e.g., time until the next sprinkler activation).
- Scale the features to ensure they are in the right format for training.
- Train a machine learning model which is Linear Regression using the data in the mongoDB.

```
def train_model():
    data = fetch_data()
    df = pd.DataFrame(data)

    # Ensure data is properly formatted
    df['temperature'] = pd.to_numeric(df['temperature'], errors='coerce')
    df['timestamp'] = pd.to_datetime(df['timestamp'])
    df['valve_open'] = df['valve'].apply(lambda x: 1 if x == "Open" else 0)

    # Prepare features and target
    df['time_diff'] = df['timestamp'].diff().dt.total_seconds() / 3600 # Time difference in hours
    df['time_diff'] = df['time_diff'].fillna(0)

    X = df[['temperature']]
    y = df['time_diff']
    # Scale features
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Train model
    model = LinearRegression()
    model.fit(X_scaled, y)
```

- Save the trained model (***sprinkler_model.pkl***) and scaler (***scaler.pkl***) to files for user interface application.

```
with open('sprinkler_model.pkl', 'wb') as model_file:
    pickle.dump(model, model_file)
with open('scaler.pkl', 'wb') as scaler_file:
    pickle.dump(scaler, scaler_file)
```

iv) Run the Dashboard (dashboard.py)

- Load the trained machine learning and scaler model.


```

with open('sprinkler_model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

with open('scaler.pkl', 'rb') as scaler_file:
    scaler = pickle.load(scaler_file)

```

- Set up a **Dash** application with UI components to display real-time sensor data and control the water tank lid and sprinkler system.
- Periodically fetch the latest sensor data from MongoDB and update the dashboard.
- The main components of the dashboard include:
 - **Time series line chart** of the temperature and humidity
 - **Current temperature and humidity** for easier readability
 - **Current rain status** whether it's raining or not
 - Manual **open/close buttons** the water tank lid and sprinkler system by sending MQTT commands.
 - Display the **next sprinkler activation** based on the trained model (previous code)

```

@app.callback(
    [
        Output("live-graph", "figure"),
        Output("current-temp", "children"),
        Output("current-humidity", "children"),
        Output("rain-status-text", "children"),
        Output("rain-status-icon", "children"),
        Output("next-sprinkler-prediction", "children")
    ],
    [
        Input("interval-component", "n_intervals"),
        Input("btn-open-sprinklers", "n_clicks"),
        Input("btn-close-sprinklers", "n_clicks"),
        Input("btn-open-lid", "n_clicks"),
        Input("btn-close-lid", "n_clicks")
    ]
)

def update_dashboard(n, open_sprinklers, close_sprinklers, open_lid, close_lid):
    # Fetch data from MongoDB
    data = fetch_data()
    df = pd.DataFrame(data)

```

```

# Create the temperature and humidity chart
fig = px.line(df, x='timestamp', y=['temperature', 'humidity'],
              title='Temperature & Humidity Over Time')
fig.update_layout(
    xaxis_title="Time",
    yaxis_title="Value",
    legend_title="Metrics",
    plot_bgcolor="#E7FAFF",
    paper_bgcolor="#E7FAFF"
)

# Get the latest data
latest_data = df.iloc[-1]
current_temp = f"{latest_data['temperature']} °C"
current_humidity = f"{latest_data['humidity']} %"

# Set rain status with text and icon
if latest_data['raining']:
    rain_status_text = "Currently Raining"
    rain_status_icon = "☔"
else:
    rain_status_text = "No Rain Detected"
    rain_status_icon = "☀️"

# Handle button clicks for controls
ctx = dash.callback_context
if ctx.triggered:
    button_id = ctx.triggered[0]['prop_id'].split('.')[0]
    if button_id == 'btn-open-sprinklers':
        client.publish(control_topic, "open_sprinklers")
    elif button_id == 'btn-close-sprinklers':
        client.publish(control_topic, "close_sprinklers")
    elif button_id == 'btn-open-lid':
        client.publish(control_topic, "open_lid")
    elif button_id == 'btn-close-lid':
        client.publish(control_topic, "close_lid")

# Make ML prediction
temperature = latest_data['temperature']

```

```

input_data = pd.DataFrame([[temperature]], columns=['temperature'])
input_data_scaled = scaler.transform(input_data)
prediction = model.predict(input_data_scaled)
next_activation = f"In {max(prediction[0], 0):.2f} hours"

return fig, current_temp, current_humidity, rain_status_text, rain_status_icon,
next_activation

```

- Run the server

```

if __name__ == "__main__":
    app.run_server(debug=True, host="0.0.0.0", port=8050)

```

Below are the steps to run the dashboard:

i) Start the VM instance to open up the SSH-in-browser.

ii) Make sure all needed libraries are installed.

```
$ pip3 install dash plotly pandas pymongo paho-mqtt  
scikit-learn
```

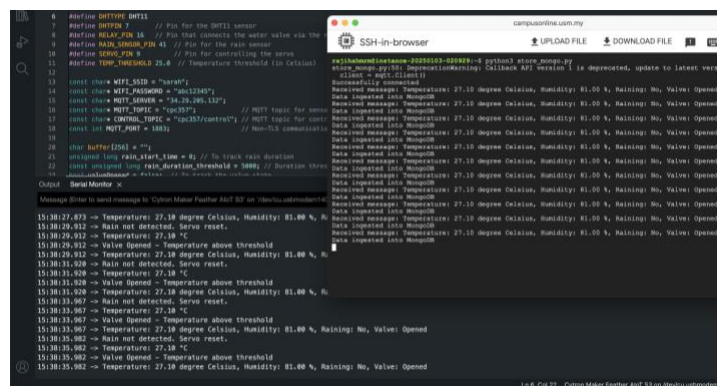
iii) Create or upload the python scripts to the current directory:

store_mongo.py, train_model.py, dashboard.py.

iv) Upload and run the Arduino code: ***sensor_reading.ino.***

v) Start data ingestion to allow data from sensor readings to store into MongoDB.

```
$ python3 store_mongo.py
```

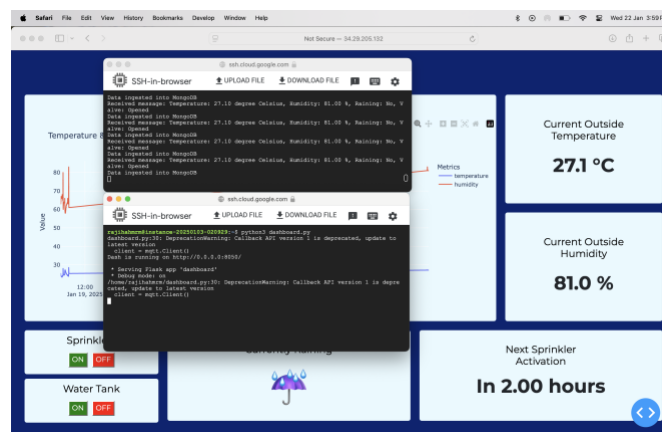


vi) Train the machine learning model after gathering data in MongoDB.

```
$ python3 train_model.py
```

vii) Start the dashboard and keep data running in MongoDB.

```
$ python3 dashboard.py
```



2.4. Final Product



Figure 2-3: Model Prototype

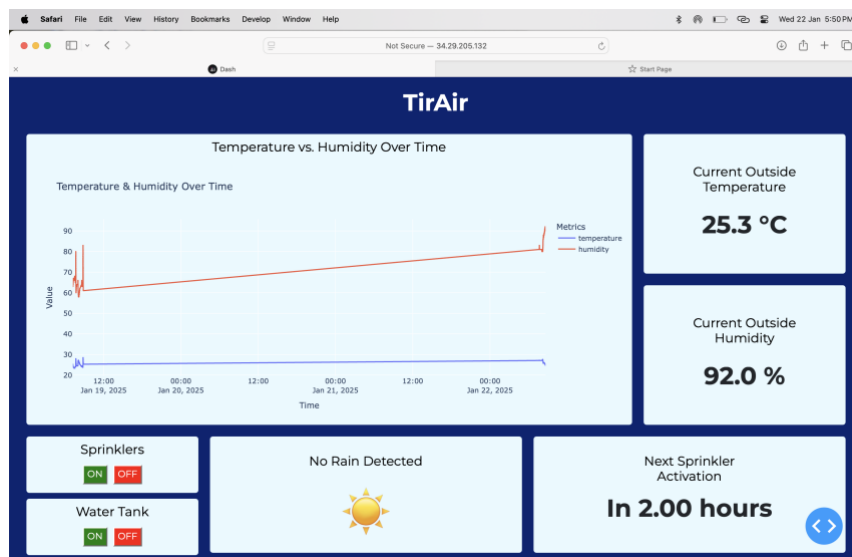


Figure 2-4: TirAir Dashboard

Figure 2-3 illustrates the prototype model of a house with a zinc roof equipped with the TirAir system for temperature regulation in hot conditions. The model features a water tank designed to collect and store rainwater. The system operates based on a temperature threshold. When the temperature exceeds the set limit, the solenoid valve connecting to a pipe is triggered to open which activates the sprinklers to cool the roof effectively.

Users can monitor and interact with the TirAir system through a user-friendly dashboard as depicted in Figure 2-4. The dashboard provides a comprehensive overview of the system's performance and allows users to manage various features conveniently. Key elements of the dashboard include:

- **Time Series Line Chart:** Displays real-time and historical data on the roof's temperature and humidity. This feature helps users to identify temperature trends and understand when the sprinkler system is activated based on these thresholds.
- **Current Roof Temperature and Humidity:** These values are prominently displayed for quick and easy readability for users to observe.
- **Current Rain Status:** A clear visual indicator shows whether it is currently raining or not. It provides users with information about the rainwater harvesting and whether the sprinkler system may be unnecessary due to natural cooling.
- **Manual Control Buttons:** Interactive buttons enable users to manually open or close the water tank lid and activate or deactivate the sprinkler system. This allows users to override automatic functions when necessary such as during maintenance or unexpected weather conditions.
- **Next Sprinkler Activation Display:** This feature informs users about the scheduled or anticipated next activation of the sprinkler system based on the data collected.

3 CONCLUSION

To conclude, the implementation of TirAir has successfully demonstrated an innovative approach to home temperature regulation through its integrated smart features. The system addresses the challenge of poor thermal comfort in homes with zinc roofing which tends to absorb and conduct heat efficiently. Additionally, it mitigates the drawbacks of traditional cooling systems that rely on water which can exacerbate water scarcity especially for critical uses such as drinking and bathing. TirAir resolves these issues by harvesting rainwater that will be used to sprinkle over the roof to reduce temperatures during hot conditions. This approach aligns with two key Sustainable Development Goals (SDGs) which are SDG 6 (Clean Water and Sanitation) and SDG 12 (Responsible Consumption and Production).

Key features of TirAir include a rainwater harvesting system using a water tank, temperature sensing to control sprinklers, and precise water flow management to cool the zinc roofs. These features collectively reduce roof temperatures, hence ensuring optimal indoor comfort.

The system's integration is achieved through IoT components structured into four layers including the perception layer to collect real-time environmental data from sensors and control actuators, the network layer to facilitate communication, the processing layer for data handling and the application layer to provide data visualization for users.

Looking ahead, the system represents a significant step toward sustainable home cooling solutions beyond its primary function of temperature regulation. By harnessing rainwater and utilizing smart technology, TirAir not only enhances living comfort but also promotes environmental responsibility through reduced energy consumption.

REFERENCES

- [1] Rahman, R. A., Kaamin, M., Suwandi, A. K., Kesot, M. J., & Zan, N. M. (2014). Cooling system of zinc roofed house by using circulated water. *Advanced Materials Research*, 935, 84-87. <https://doi.org/10.4028/www.scientific.net/amr.935.84>
- [2] Rizzo, G., Beccali, M., & Nucara, A. (2004). Thermal Comfort. In *Elsevier eBooks* (pp. 55–64). <https://doi.org/10.1016/b0-12-176480-x/00551-9>
- [3] Ho, S. H., Rosario, L., & Rahman, M. M. (2008). Thermal comfort enhancement by using a ceiling fan. *Applied Thermal Engineering*, 29(8–9), 1648–1656. <https://doi.org/10.1016/j.applthermaleng.2008.07.015>
- [4] Alam, M. M., Siwar, C., Murad, W., & Toriman, M. E. B. (2019). Impacts of climate change on agriculture and food security issues in Malaysia: An Empirical study on farm level assessment. *International Digital Organization for Scientific Information*. <https://doi.org/10.31219/osf.io/mdqpz>
- [5] Selan, S. (2024, May 20). Malaysia likely to face water crisis by 2030 amid growing demand: officials. *Eco-Business*. <https://www.eco-business.com/news/malaysia-likely-to-face-water-crisis-by-2030-amid-growing-demand-officials/>
- [6] Feloni, E., & Nastos, P. T. (2024). Evaluating Rainwater Harvesting Systems for Water Scarcity Mitigation in Small Greek Islands under Climate Change. *Sustainability*, 16(6), 2592. <https://doi.org/10.3390/su16062592>
- [7] United Nations. (n.d.). *Goal 6 | department of economic and social affairs*. [Sdgs.un.org; United Nations. https://sdgs.un.org/goals/goal6](https://sdgs.un.org/goals/goal6)
- [8] United Nations. (2024). *Goal 12 | Ensure sustainable consumption and production patterns*. United Nations. <https://sdgs.un.org/goals/goal12>

[9] *Answers to: Why is corrugated zinc roof is very hot in summer and very cold in winter.* (2023). Class Ace.

https://www.classace.io/answers/why-is-corrugated-zinc-roof-is-very-hot-in-summer-and-very-cold-in-winter#google_vignette