Smart Wall Art – Interactive Art That Changes Based on Environmental Input

Simone Reale
Alma Mater Studiorum University of Bologna
simone.reale3@studio.unibo.it

***Abstract*—The Smart Wall Art project seeks to merge art and technology to create dynamic and interactive experiences that adapt to the user's environment. The objective is to make art more engaging, creating a link between external stimuli and visual representations. The presented architecture implements this connection through an Internet of Things (IoT) pipeline made up of three nodes, each responsible for a specific part of the project. The presented edge-to-cloud system is described in all its details, from the data acquisition to the cloud ingestion and data analytics. The setup is fully configurable, in terms of communication protocols for the data interchange, sampling rate for data collection, and motion alert for observers detection.**

***Index Terms*—Internet of Things (IoT), ESP32, Smart Art, Prophet**

I. Introduction

The project delves into the implementation of an ESP32-based Smart Wall Art system, orchestrating heterogeneous protocols (MQTT, CoAP, HTTP) for optimized environmental and motion data transmission. The architecture integrates a robust pipeline from sensor acquisition to InfluxDB storage, enriched by predictive models for future trend analysis displayed on Grafana. System responsiveness is ensured by immediate MQTT motion event processing, triggering real-time dynamic geometric pattern generation. This solution demonstrates the validity of a modular IoT approach, reliably adapting artistic expression to physical context variations. Ultimately, the prototype confirms the effectiveness of merging embedded technologies with data analytics to create interactive and resilient user experiences.

The report is organized as follows: (i) section II defines the backbone and the design choices; (ii) section III describes the firmware; (iii) section IV presents a performance analysis of the system.

II. Architecture

The architecture is designed as a distributed IoT pipeline comprising three distinct logical nodes, intended to run on different devices:

- *Sensing Node (Edge)*
- *Data Proxy and Analytics Node (Middleware)*
- *Display Node (Actuator)*

This modular architecture (Figure 1) ensures scalability and clearly separates data acquisition from high-level processing and visualization.

A. Edge Device: Sensing Node

The core of the sensing layer is an *ESP32 microcontroller*, equipped with a *DHT22* temperature/humidity sensor, a *TEMT6000* ambient light sensor, both powered at 3.3V for compatibility with the GPIO ports, and a *HCSR505* PIR sensor, powered at 5V (Figure 2[1]). The first two periodically acquire data with a frequency set through the configurable parameter `sampling_rate`, for environmental monitoring. The third sensor triggers a `motion_alert` if there is human presence for more than a certain number of seconds, as regulated by the relative parameter.

B. Middleware: Data Proxy and Visualization/Storage

The backend infrastructure is managed by a *Data Proxy*, an application running on a separate machine, hosting also the *Mosquitto Broker* for MQTT messages. This is the central node of the architecture, as shown in figure 1, and acts as a bridge between the IoT edge and the storage layer. It enables remote configuration of the sensing node and forwards the telemetry information to an *InfluxDB* instance.

Additionally, a self-adaptive *Data Analytics Module* is integrated in the workflow in order to process historical data and forecast future environmental values, evaluating the accuracy across different metrics.

The historical and forecasted time series are then visualized through a *Grafana dashboard* that is connected to the InfluxDB database. It shows both the current temperature, humidity and ambient light values and the timeline panel of past and future data.

[1]AI generated image.

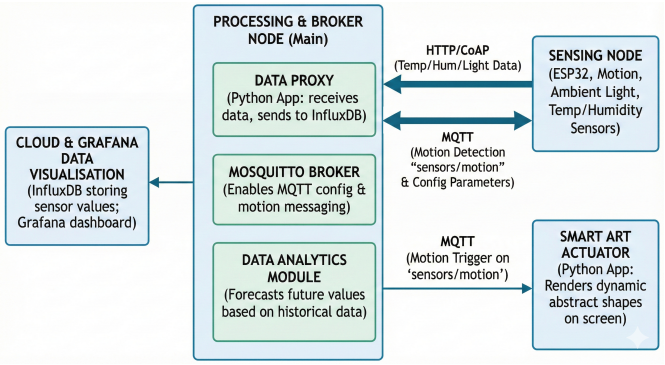

Fig. 1: Architecture Graph

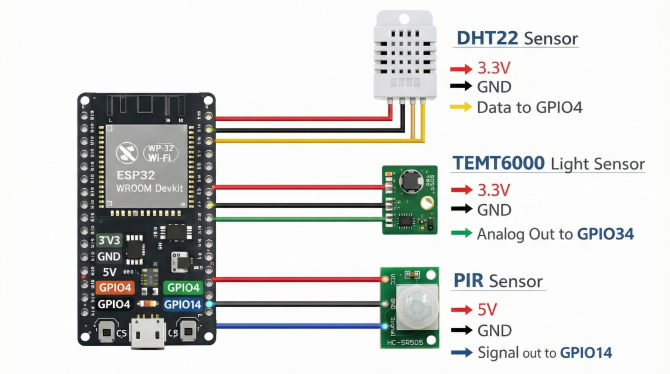

Fig. 2: Sensing Node wiring schema.

C. Actuator: Display Node

This application represents the interactive output of the system. When user presence is detected, it renders a "Blinking Eye", whose characteristics are non-static and modulated in real-time, depending on the last measurement perceived for its relative node id.

D. Communication Protocols

The sensor readings are transmitted to the remote server node using either the *CoAP* or *HTTP* protocol. This selection is determined by a configurable communication mode through the parameter `protocol`, allowing the system to adapt to different network constraints.

On the other side, *MQTT* is used for asynchronous, bi-directional communication of both the system configuration and the motion alert (Figure 1). Quality of Service (QoS) 1 is used to have a balance between efficiency of the communication and robustness to data losses. In addition, it has been decided to use retained messages in order to have a system that is able to recover automatically from errors that cause single nodes to stop if any, guaranteeing continuity.

III. Implementation

The firmware for the different nodes is written in Python for the Middleware and the Actuator, while the ESP32 microcontroller is programmed using the *ESP-IDF framework* for the Arduino language.

The Sensing Node is designed to handle asynchronous tasks for data acquisition, communication and movement perception through *freeRTOS*. The main loop is used for main objective of the node, i.e. the temperature, humidity and light acquisition, while a parallel task is run for motion alerts.

The forecasting engine of the Data Analytics Module is implemented using the *Facebook's Prophet* model [1], selected for its scalability and inherent robustness against missing or spurious data points, common in IoT networks. It is set to follow a *logistic growth*, which allows to define a bounded capacity for limiting the prediction. For each measure, the ranges between *'floor'* and *'cap'* values are chosen according to sensors' specifics and so 0-80 °C for temperature, 0-100 %H for humidity and 0-660 lux for ambient light. It is defined to identify a daily seasonal pattern, since the application does not present any other specific patterns, with a curve that must not be prone to noise. In addition, the model is fed a rolling window of one week of historical data and operates in *logspace* to stabilize variance and reduce the impact of outliers. This module is intended to run in parallel to the Data Proxy and is self-adaptive, meaning that the measures' models are periodically fitted on the previous week of historical data and saved. The initial conditions for the fit are set according to defaults models, that are referred to a default dataset, and each time performance measures are evaluated in the logs.

The rendered output is a "Blinking Eye" (Figure 3), a combination of geometrical shapes (lines, circles and ellipses) that generate an eye shape, realized through the *PyGame* framework. Its characteristics are not static and change in real-time depending on environmental measures as follows:

- the temperature determines the color of the iris, ranging from green for low values to red for higher ones.
- the humidity determines the radius base unit of the pupil and so the size of the eye, as long as the framing rate of the blinking.
- the light illuminance determines the rotation speed of the rays on the outside.

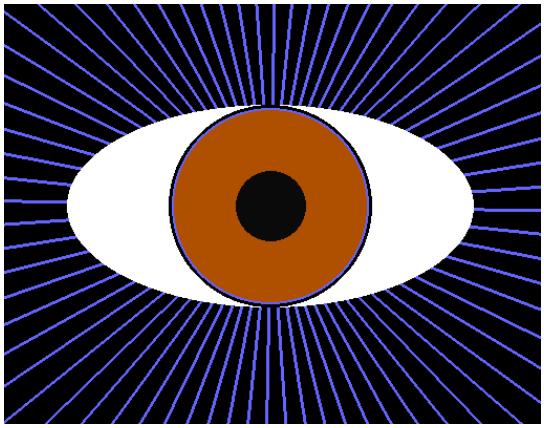

Fig. 3: *The Blinking Eye*

IV. Results

Beyond its visual and creative impact, this project fundamentally serves as a rigorous technical case study in distributed IoT architectures, validating the efficiency of hybrid communication protocols and real-time predictive analytics.

In particular, the performance of the predictive module was evaluated to ensure the system can reliably anticipate environmental trends and changes. Obviously, numbers and curves change according to which data the model is fed with, so the following analysis is related to the default dataset. They are collected in a partially controlled and unstable environment, which is different from the possible real application of the project, so outcomes are influenced by high frequency noise and unpredictable variances.

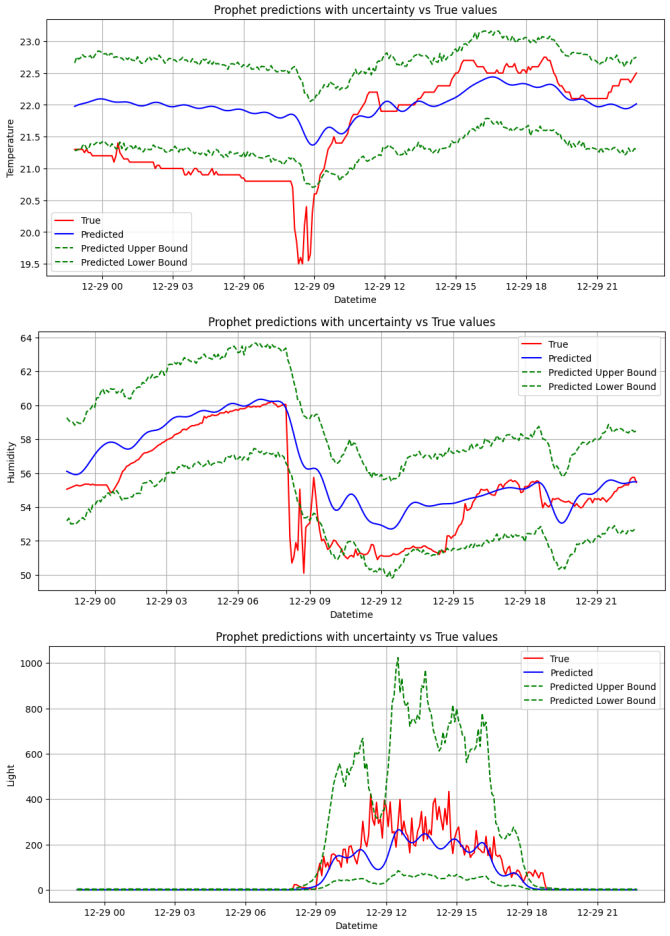

Fig. 4: Ground truth vs Prediction, with error intervals (a. temperature b. humidity c. light).

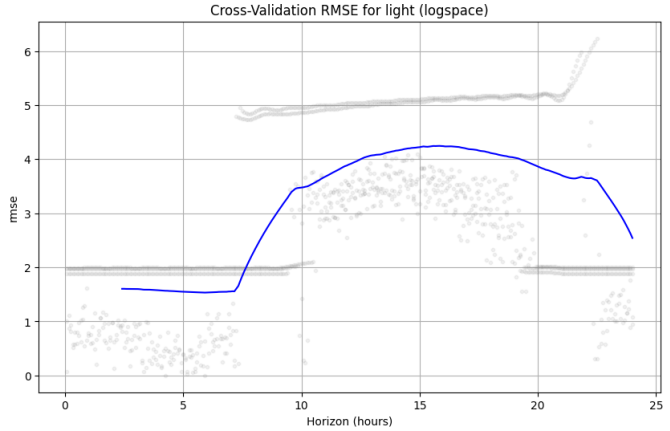

Fig. 5: RMSE evolution plot with a horizon of 1 day.

The evaluation is conducted over description plots and three metrics, *Root Mean Squared Error (RMSE)*, *Mean Absolute Error (MAE)* and R^2 *Score*. The default dataset spans over a week of data, split into approximately *6 days* for the training set and *1 day* for the validation set, which is the same time interval used for periodic predictions. The resulting performance numbers are shown in the table [2]:

Measure	RMSE	MAE	R^2
temperature	0.7230	0.5674	**0.0674**
humidity	1.8260	1.3061	**0.5859**
light	60.135	27.585	**0.7256**

In general, the Prophet model is tuned to avoid noise and capture the daily trends of environmental sensors, as said in section III.

Looking at the evolution timeline of historical data, temperature and humidity values show a more uniform curve overall. However, especially in the temperature case, they present more unpredictable spikes and anomalies, due to opening and closing windows or the heating system for instance. This behaviour is reflected in the prediction capabilities of the model in terms of R^2 Score, where the temperature model has a significant low score while humidity seems to be affected much less, showing a discrete performance value. Anyway, looking at the figure 4, they do not struggle to catch the evolving trends with a constant and restricted error interval going forward in time.

On the other side, ambient light predictions have opposite characteristics, with a much bigger error interval, caused by a higher noise in observed data. In addition, the cross-validation plot of the RMSE evolution over future timestamps (Figure 5) revealed that prediction accuracy degrades significantly after a horizon of 6 to 7 hours. For this reason, to ensure the art installation responds to accurate future trends rather than stale data, the Data Analytics Module requires a retraining pipeline that executes every 6 hours. It has been considered also to retrain the models at each data ingestion, but latency would have been subject to a big increase and therefore the solution above has been chosen. However, day-night cycles and trends for ambient light measurements are clearly defined, such that Prophet succeeds in catching their patterns and this is reflected in the best R^2 Score among all the measures.

REFERENCES

[1] Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

[2]Further details of the exploration are in the attached notebook.