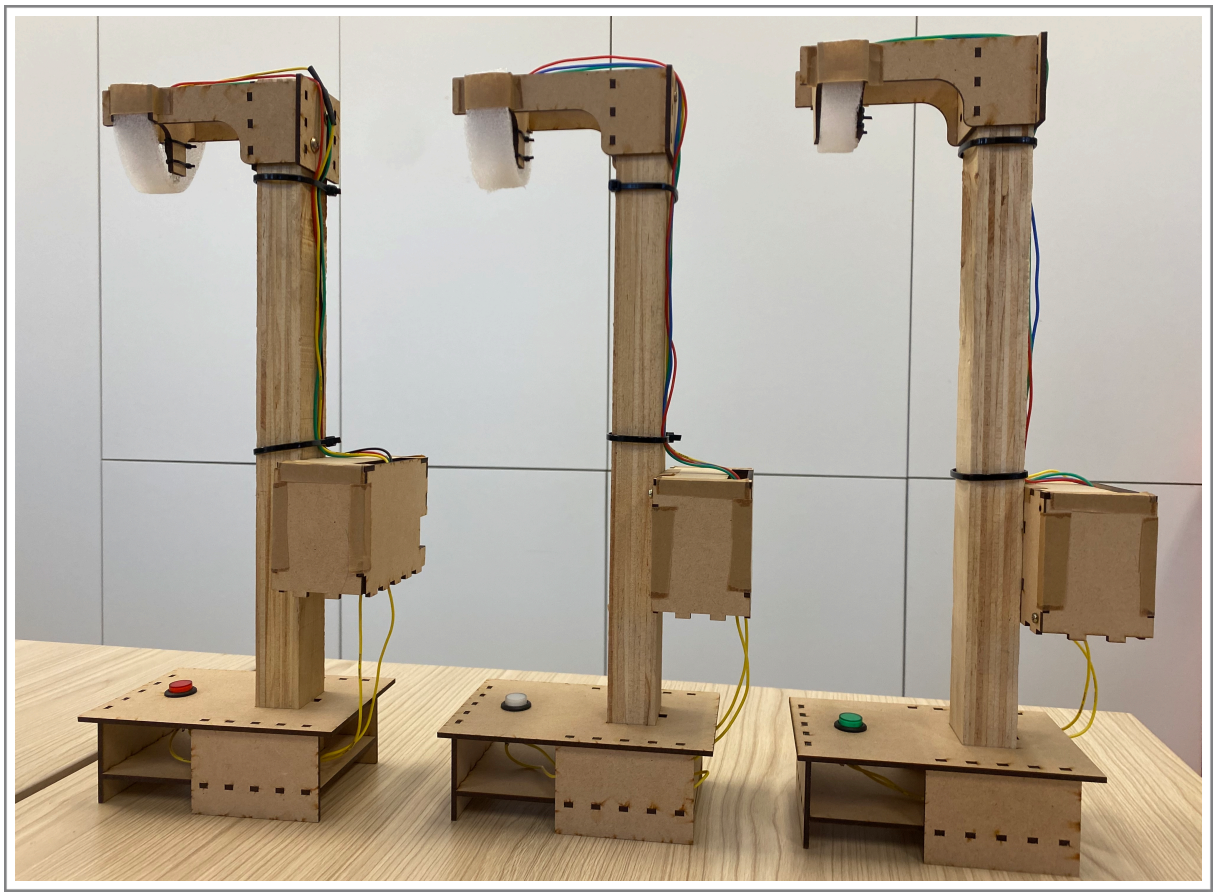
Genie's lamp

IoT Intelligent Streetlight Network



陳宏恩, 莊加旭, 黃柏睿

Fall 2022

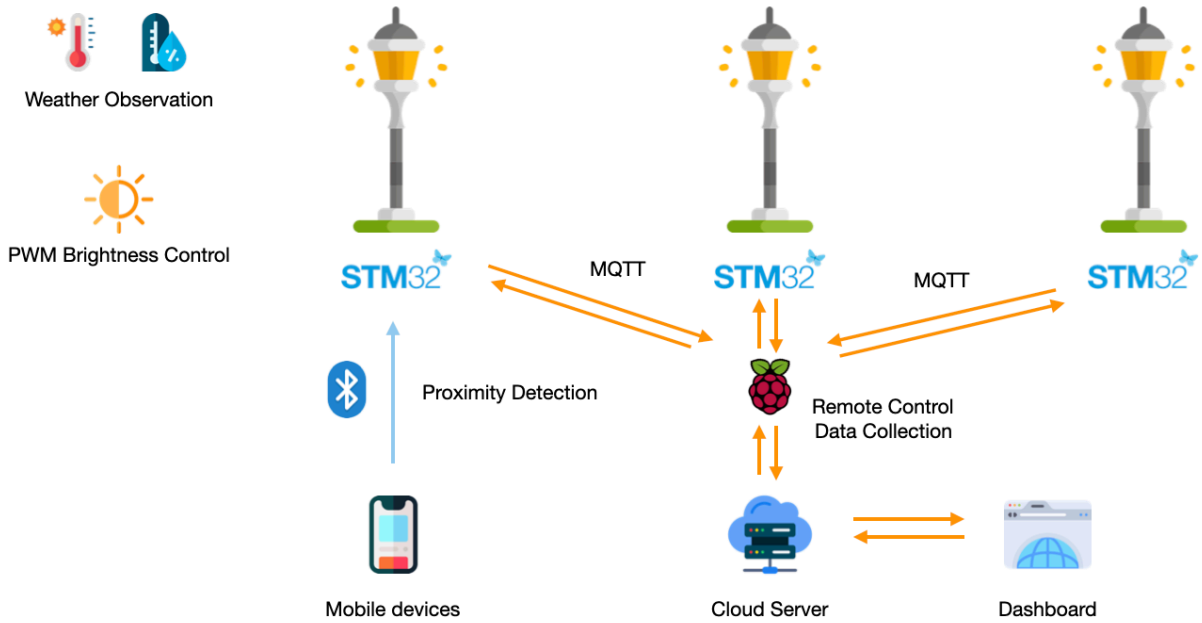
Genie's lamp

IoT Intelligent Streetlight Network

Introduction

Streetlight is an essential element in a modern city at night. It illuminates the road and gives pedestrians a sense of safety. However, it constantly consumes energy even when there's none nearby. Therefore, we propose Genie's Lamp, an IoT intelligent streetlight network that can conserve energy via smart brightness control. In addition, this streetlight can serve as a real-time weather observation network with high resolution. This streetlight network can also be an emergency report system for people at night. The implementation is on GitHub, and the demo videos are available on YouTube.

Architecture



Features

Genie's Lamp currently supports the following features:

- brightness control
  - use PWM to control the brightness
  - RGB Led to control the color
- proximity detection
  - turn on the light only when people pass by
  - Bluetooth signal intensity
- emergency report
  - report the location of the event to the officials
  - change the street lamp's color to red
- remote access
  - control the color and brightness of street lights from the web dashboard
  - receive notification when emergency occurs
- real-time data visualization
  - displays real-time temperature and humidity data

Implementation

Lamp:

Communication via MQTT

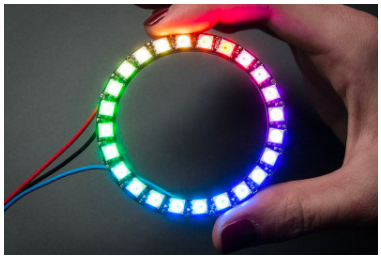
To achieve bidirectional communicate with server of lowest cost, we choose MQTT to be our communication protocol. Different QoS and encryption method allows out network to be more stable and secure.

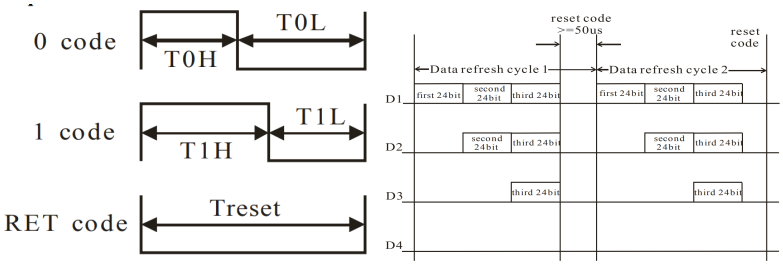
Our lamp can transmit the weather observation data with "publish" function to specific topic of the following format `/{lamp_id}/{sensor_type}` for example : `/lamp1/humidity`. Then "mqtt2db" can read and store sensor data by subscribe to the topic.

As for brightness and color control, we use "subscribe" function to receive control command from `/lamp1/brightness`.

Brightness and Color Control through PWM

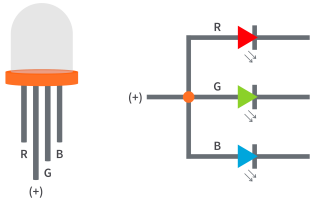
We originally want to use WS2812 LED light strip to control multiple LED with different color and brightness all at once. WS2812 takes only one signal pin for input, using 24bit sequence for one LED. Each bit is represent as PWM-liked signal with duty cycle difference.





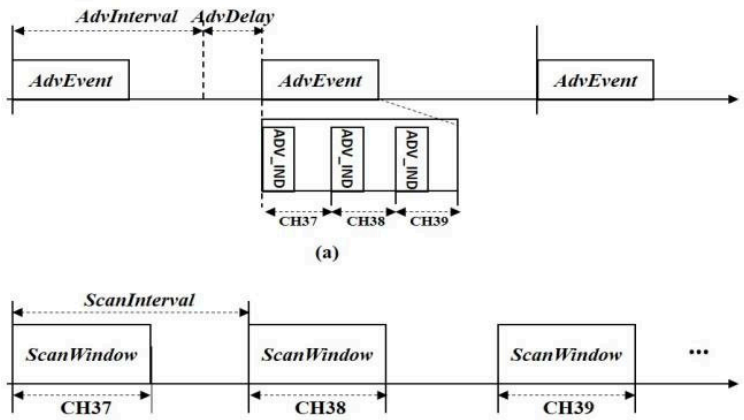
However, Mbed OS implement PWM output using a different timer asynchronously, and the processor can only handle synchronous pin output in about 550ns resolution, which is not enough for the resolution required by WS2812 (350ns). We have try some other approach such as SPI, DMA to generate the signal, but all failed due to the processor clock rate.

In the end, we choose to use a simple RGB LED for our light source. Three diode have common anode so we can control three color individually using PWM with three pin.



Proximity Detection via BLE

We use Mbed OS native BLE api defined in “ble/BLE.h” to scan nearby packets. To be more specific, we use the BLE GAP API to capture packets advertised from nearby services. By filtering device with RSSI and the device name, we can detect whether any pedestrian is waking by or not. If we discover a new device with increasing RSSI, we know that someone is



getting close to the streetlight. Then we can increase the brightness of the light and send a MQTT notification to signify the backend.

We initially use default settings to scan devices, using an EventQueue to store received events. And we can dispatch them asynchronously in the loop to process proximity informations. However, the program suffered stack overflow over time. We guess that the "scan" method receive too much information to EventQueue and we the dispatch time isn't enough to clear all the queue, and consequently cause stack overflow or hard fault.

Therefore, we try to tuning the parameter for scan. Lowering the scan window and increasing the scan interval can effectively reduce the pressure of the EventQueue, but it also reduce the chance to scan the phones. Unfortunately, phone devices are often designed as central and put less effort at advertising, making them harder to be scanned by our lamp. Tuning the parameter of each scan is very challenging, we need to balance the speed of scanning, quality of scanning, and also the speed of processing.

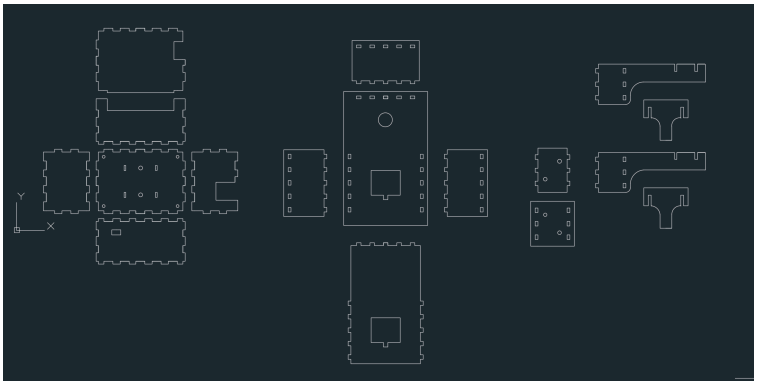
Finally, we found a good configuration working well, but the accuracy and stability depends on the environment and target device a lot. Perhaps we can try to implement proximity detection by classical Bluetooth or WiFi in the future.

Emergency Report Button Service

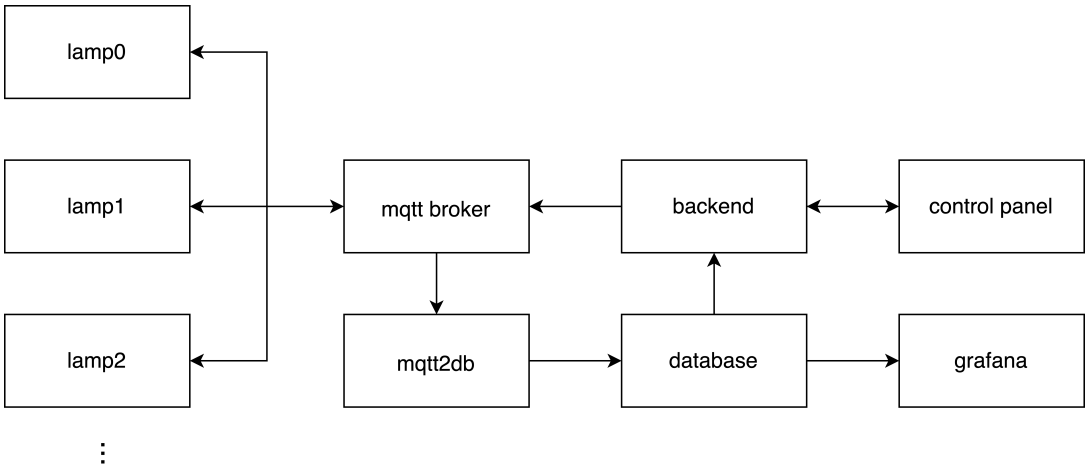
We use a two-stage button for emergency report service. If the button is pressed down, an MQTT packet will be send to MQTT broker and signify the frontend about the emergency report. We implemented two version. One is using interrupt and another is using digital read. The advantage of interrupt is it respond immediately, which is suitable for emergency report, but it also post some risk of race condition or reentrant, and debouncing is essential. The advantage of using simple digital read is we can use in-board pull-down resistor. The circuit design of hardware can be neater. After consider both pros and cons, we choose to use simple digital read since the delay is at most 1 second, which doesn't mater a lot.

Hardware Design

We use 3D laser cutting for the structure. It is fast, cheap, and easy to design. As for the circuit wiring, we align the pin in a row so that solder them together easily.



Server and Frontend Services:



MQTT Broker

An Eclipse Mosquitto MQTT broker that connects all our network. The lamps can subscribe and publish data from it.

MQTT2DB

A service that subscribes to sensor data published by the lamps and saves the data into the database.

Database

A PostgreSQL database that stores the data, which integrates well with Grafana.

Backend

An HTTP server that facilitates the control panel. It receives HTTP requests from the front end and publishes the corresponding control signal to the lamps.

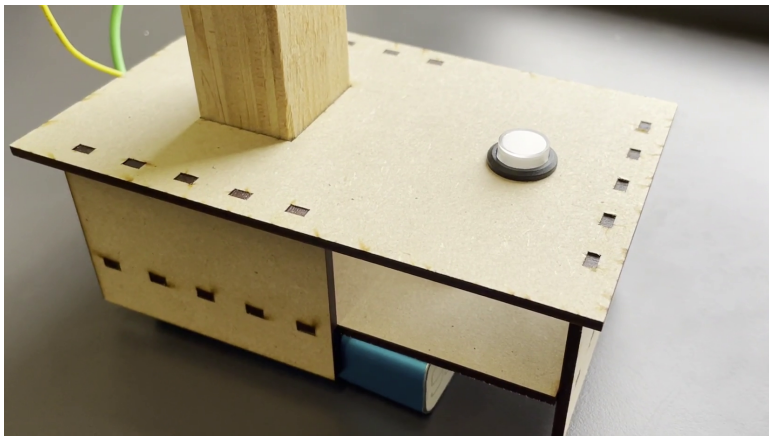
Control Panel

A web app built with React that can control the color and brightness of all lamps. The service also receives notifications when the streetlight network reports an emergency.

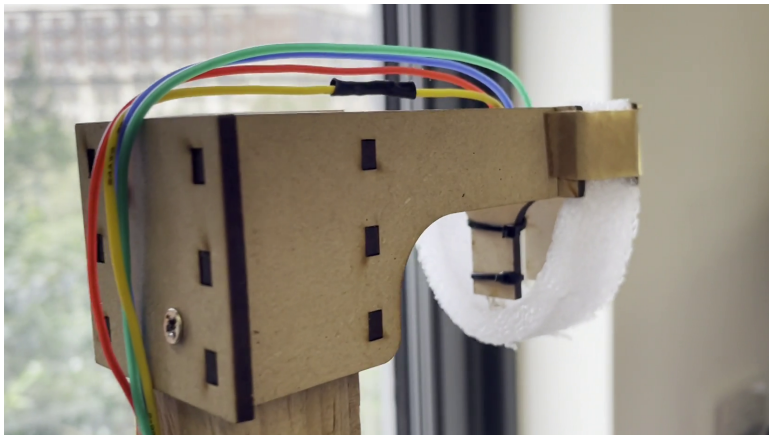
Grafana Dashboard

An open-source data-visualizing web service. We use it to visualize the weather observation data from the lamp.

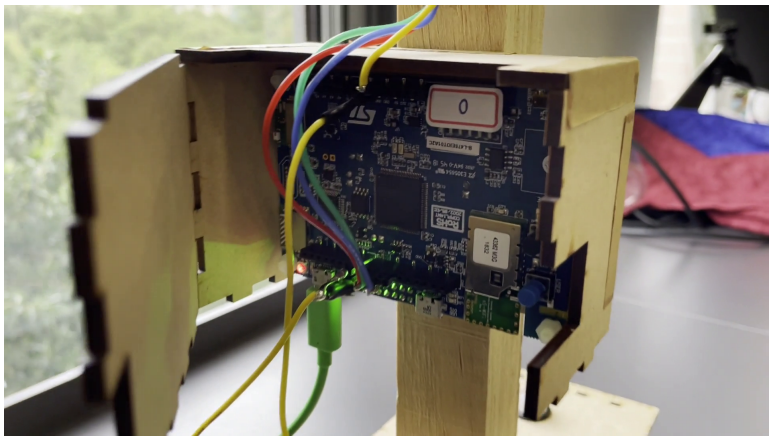
Demonstration



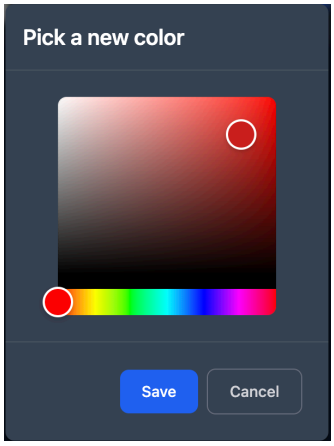
Emergency Button



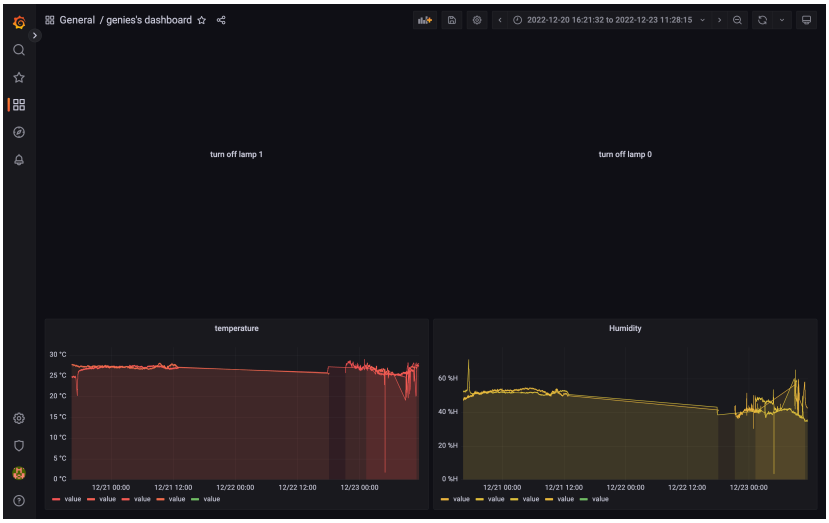
LED Light



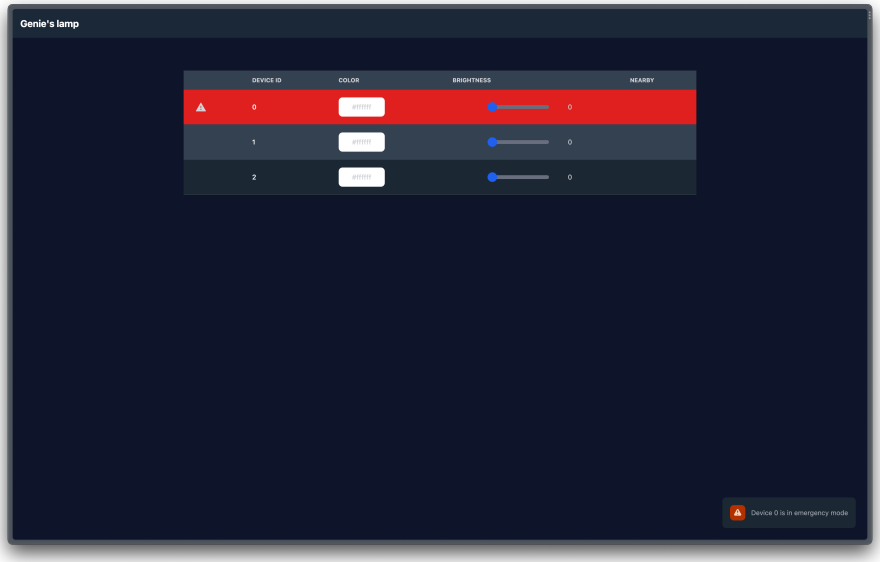
Control Box



Color Control



Grafana Dashboard for sensor data



Control Panel & Emergency Report

Conclusion

We propose Genie's Lamp, an IoT Intelligent Streetlight Network. It is capable of energy conservation, weather observation, and emergency report. With all the knowledge we learned from the class - multithreading, BLE, PWM, socket programming, and some skills we had, we managed to implement this project. It is a complete project and makes us improve a lot.

References

1. Mbed OS: https://os.mbed.com/mbed-os/
2. BLE GAP API: https://os.mbed.com/docs/mbed-os/v6.15/apis/gap.html
3. PwmOut API: https://os.mbed.com/docs/mbed-os/v6.15/apis/pwmout.html
4. MQTT: https://mqtt.org/
5. HelloMQTT Library: https://os.mbed.com/teams/mqtt/code/HelloMQTT/file/d8a31b66a85d/main.cpp/