**Báo cáo tuần 5**

# **Công việc đã làm trong tuần qua**

Tuần qua em đã test lại quá trình gửi nhận dữ liệu điều khiển giữa Arduino Uno, Esp-8266 với Firebase và phát hiện ra lỗi:

* Lỗi 1: xảy ra trong quá trình gửi dữ liệu String quá dài từ Esp8266 lên firebase, firebase không nhận được dữ liệu cập nhật. Dữ liệu về mã hồng ngoại được lưu dạng string như sau: *“9008 4480 581 573,…”*, (các số thể hiện cho khoảng thời gian bật/tắt), cách này không tối ưu, tốn dung lượng lưu trữ trên firebase do với đoạn mã thông thường số lượng các giá trị số ở khoảng từ 70-100 và có thể lên tới 400 giá trị đối với điều hòa. Cách khắc phục: nén mã hồng ngoại thành dạng chuỗi byte, chuỗi kí tự in được.
* Lỗi 2: SRAM trên Uno chỉ có 2kB, rất dễ tràn bộ nhớ vì bộ đệm sử dụng khi thu tín hiệu hồng ngoại đã chiếm 1kB đồng thời việc truyền UART cũng cần bộ đệm khi mà dữ liệu là dạng chuỗi byte dài. Giải pháp sử dụng: chỉ sử dụng Esp8266, có dung lượng SRAM lên đến 80kB.

1. **Nén dữ liệu hồng ngoại**

Dữ liệu hồng ngoại thu được từ module thu hồng ngoại là một chuỗi các giá trị kiểu int, thể hiện cho khoảng thời gian bật/tắt của đèn phát hồng ngoại theo đơn vị micro giây. Ví dụ một chuỗi dữ liệu 71 giá trị thu được có dạng như sau:

*“9008 4480 581 573 582 573 582 572 581 573 582 573 581 573 581 573 581 573 581 1673 556 1674 556 1673 556 1674 556 1674 556 1674 556 1673 556 1672 558 573 581 1674 556 1673 556 574 581 1673 557 572 582 573 581 573 582 1673 557 572 581 574 581 1673 557 573 581 1674 557 1673 556 1673 557 39342 9011 2250 557”*

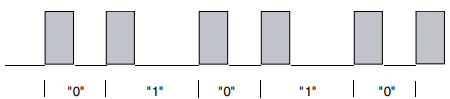
Ta có thể làm tròn thành bội của 50 mà độ chính xác vẫn cao. Chuỗi dữ liệu sau khi làm tròn thành bội của 50:

*“9000 4500 600 550 600 550 600 550 600 550 600 550 600 550 600 550 600 550 600 1650 550 1650 550 1650 550 1650 550 1650 550 1650 550 1650 550 1650 550 550 600 1650 550 1650 550 550 600 1650 550 550 600 550 600 550 600 1650 550 550 600 550 600 1650 550 550 600 1650 550 1650 550 1650 550 39350 9000 2250 550”*

Mục đích của việc làm tròn là có thể giảm được dải biểu diễn của dữ liệu từ int sang byte, chuỗi sẽ giảm dung lượng đi còn 1 nửa. Chia toàn bộ giá trị cho 50 sẽ thu được một chuỗi mới có giá trị nhỏ hơn và đa số có thể biểu diễn bằng 1 byte.

*“180 90 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 33 11 33 11 33 11 33 11 33 11 33 11 33 11 33 11 11 12 33 11 33 11 11 12 33 11 11 12 11 12 11 12 33 11 11 12 11 12 33 11 11 12 33 11 33 11 33 11 787 180 45 11”*

Tiếp đến, đối với dữ liệu của tín hiệu hồng ngoại, thông thường sẽ theo chuẩn mã hóa khoảng cách (pulse distance). Cụ thể như hình bên dưới:



Các khoảng thời gian tín hiệu ở mức cao là như nhau, do vậy có thể sử dụng kiểu nén chuỗi: các kí tự liền nhau, giá trị giống nhau sẽ được thay bằng giá trị và số lượng. Cụ thể ở đây sẽ là các giá trị xen kẽ nhau 1 vị trí mà giống nhau liên tiếp sẽ được thay bằng giá trị và số lượng:

* Duyệt các giá trị ở vị trí chẵn, nếu các giá trị liên tiếp giống nhau sẽ được thay bằng giá trị và số lượng, đồng thời chèn thêm 1 kí tự có mã 33 ở trước để báo hiệu.
* Tương tự với các kí tự ở vị trí lẻ

Xét đoạn đầu của chuỗi ví dụ bên trên:

“*180 90 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 33,…”*

Các giá trị có số lượng 1 như 180, 90 sẽ được giữ nguyên. Ở vị trí chẵn, ta thấy có 9 giá trị 12 liên tiếp, vậy 9 giá trị này sẽ được thay bằng 33, 9 và 12. Ở vị trí lẻ ta có 8 giá trị 11 liên tiếp, vậy 8 giá trị này sẽ được thay bằng 33, 8 và 11.

Tuy nhiên firebase chỉ chấp nhận chuỗi là các giá trị ACSII in được, vậy nên phải chuyển các giá trị byte trên (ngoại trừ kí tự báo hiệu 33) thành kí tự ASCII in được, quy ước mã hóa đối với từng giá trị:

* 1 – 63: giá trị được cộng thêm 60 và ép kiểu thành dạng char (1 kí tự in được có mã trong khoảng từ 61 - 123)
* 64 – 4096: chuyển thành 2 kí tự in được. ví dụ 787 = 12 \* 64 + 19, sẽ chuyển thành 2 giá trị 12 và 19, cộng thêm 60 để chuyển thành kí tự in được như trên. Chèn thêm 1 kí tự có mã 34 ở đầu để phân biệt với các byte lẻ. Tổng cộng ta cần 3 byte cho giá trị 787 là: 34, 72 và 79. Thông thường các giá trị sẽ tập trung ở khoảng 1 – 63, chỉ có một vài giá trị nằm trong khoảng này.

Chuỗi thu được sau toàn bộ quá trình nén:

*=C">p!1H!0GH!\*GHG!+HG!\*HGH!+G">pG#"=V!0G!0]G!\*]G]!+G]!\*G]G!+]"HOi*

Hai kí tự đầu tiên là độ dài chuỗi dữ liệu phía sau (2byte). Từ kí tự tiếp theo đến kí tự ‘*#’* là những giá trị ở vị trí chẵn, phần còn lại là ở vị trí lẻ. Thuật toán cho kết quả: có thể giảm còn ¼ so với độ dài chuỗi giá trị ban đầu. Tỷ lệ giữa độ dài chuỗi nén so với số giá trị ban đầu có thể chỉ là ½. Hoạt động hiệu quả với số lượng giá trị nhiều, tín hiệu thu được càng chuẩn sẽ hoạt động càng hiệu quả. Như chuỗi ví dụ trên, từ một chuỗi 71 giá trị kiểu int, đã được nén thành chuỗi kí tự có độ dài 65bytes.

1. **Thu phát hồng ngoại trên Esp-8266**

Không sử dụng mạch uno để thu phát mà chuyển sang Esp-8266, tất cả hoạt động sẽ được thực hiện trên Esp-8266: thu hồng ngoại, phát hồng ngoại, kết nối và giao tiếp với Firebase. Thay vì trước đó là sử dụng Uno để thu và phát hồng ngoại, Esp-8266 điều khiển Uno qua chuẩn nối tiếp UART.

*A circuit board

Description automatically generated*A desk with a computer mouse and keyboard

Description automatically generated

*Trước Sau*

**Note:** mã nguồn em cập nhật trên github: <https://github.com/huydinh3010/DoAnHeNhung>

# **Dự kiến tuần tới**

* Thêm chức năng hẹn giờ
* Chỉnh sửa giao diện ứng dụng
* Hàn module thành mạch nếu đã hoạt động ổn định