**Báo cáo tuần 6**

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

Tuần qua em đã sửa lại lỗi khi phát tín hiệu hồng ngoại trên esp-8266 và cài đặt được chức năng lên lịch hẹn giờ cơ bản trên hệ thống: lưu trữ được 10 tín hiệu để lên lịch cùng lúc (có thể mở rộng hơn) và hẹn giờ có lặp lại theo chu kì.

Toàn bộ mã nguồn và báo cáo em cập nhật hàng tuần trên github: <https://github.com/huydinh3010/DoAnHeNhung>

1. **Cập nhật thời gian thực trên Esp-8266**

Sử dụng thư viện “NTPClient.h” để lấy dữ liệu thời gian thực bằng kết nối UDP đến server “pool.ntp.org”, thư viện sẽ tự động đồng bộ thời gian với server trong một khoảng thời gian được định trước. Trong khoảng thời gian đó, thư viện sẽ sử dụng timer trên Esp-8266 để đếm thời gian thông qua 2 hàm *milis()* và *micros()*, hai hàm này sẽ trả về khoảng thời gian trôi qua theo micro giây và mili giây kể từ khi chip bắt đầu chạy.

Thư viện có các hàm/phương thức hữu ích như:

* *Constructor(WiFiUDP wifiUdp, String serverName, long timeOffset, unsigned long updateInterval)*: khởi tạo đối tượng NTPClient gồm các thuộc tính:
  + WifiUDP: khởi tạo kết nối UDP đến server
  + serverName: địa chỉ server
  + timeOffset: múi giờ tính theo giây, ví dụ Việt Nam có múi giờ là UTC+7, nhanh 7 tiếng so với giờ GMT, timeOffset cộng thêm 7 giờ tương ứng là 25200 giây
  + updateInterval: khoảng thời gian tính theo giây sẽ thực hiện đồng bộ thời gian với server, đang được đặt là 1 giờ (3600 giây).
* *update( )*: khi gọi phương thức này, đối tượng NTPClient sẽ kiểm tra đã vượt quá updateInterval hay chưa, nếu vượt quá sẽ tiến hành đồng bộ thời gian với server. Phương thức này sẽ được đặt trong thân hàm loop( ) của chương trình.
* *getFormattedTime( )*: trả về thời gian theo định dạng “hh:mm:ss”
* *getEpochTime( ):* trả về thời gian Epoch – là số giây trôi qua kể từ ngày 1/1/1970, phương thức này sẽ được sử dụng để so sánh độ lệch thời gian, từ đó tính ra số giây cần chờ để thực hiện hẹn giờ trên Esp-8266.

1. **Hẹn giờ trên Esp-8266**

Sử dụng thư viện “Ticker.h” để hẹn giờ, ticker được cài đặt dựa trên software timer của Esp-8266, software timer hỗ trợ sẵn hẹn giờ, thực thi nhiệm vụ thông qua hàm callback. Esp-8266 có 2 hardware timer là timer0 và timer1, nhưng chỉ có timer1 là có thể sử dụng vì timer0 được dùng cho Wifi. Ngược với hardware timer, từ hardware timer có thể khởi tạo được nhiều software timer, đồng nghĩa với việc ta có thể tạo nhiều đối tượng Ticker để hẹn giờ hoặc lập lịch cho nhiều nhiệm vụ được điểu khiển bằng các software timer riêng biệt.

Các hàm/phương thức hữu ích của thư viện:

* *attach\_ms(uint32\_t miliseconds, void\* callback, void \* arg):* phương thức này để thiết lập hẹn giờ với khoảng thời gian miliseconds tính từ lúc bắt đầu gọi hàm, có lặp lại sau mỗi khoảng thời gian đó. Thực thi hàm callback với các tham số arg. Với kiểu uint32\_t ta có thể hẹn giờ lên đến gần 50 ngày.
* *once\_ms( ):* tương tự *attach\_ms* nhưng chỉ thực thi hàm callback 1 lần.
* *detach( ):* hủy timer và callback
* Ngoài ra còn có các hàm nạp chồng lại hoặc có chức năng tương tự hai hàm *attach\_ms( )* và *once\_ms( )*, có thể nhận tham số là giây, hàm callback và tham số cho hàm callback được đóng gói thành một lớp,…

Chức năng hẹn giờ: nhận dữ liệu từ người dùng trên app chứa thông tin về thời gian, chu kì lặp và mã hồng ngoại, chương trình sẽ thực hiện lên lịch để phát tín hiệu hồng ngoại khi đến đúng thời điểm đã đặt và sẽ phát lại theo chu kì lặp nếu có hoặc chỉ phát một lần.

Triển khai: tạo một lớp chứa các thông tin để hẹn giờ phát tín hiệu hồng ngoại: khoảng thời gian chờ, mã hồng ngoại, chu kỳ lặp lại và một đối lượng Ticker để hẹn giờ, cùng với đó là phương thức phát tín hiệu hồng ngoại. Khởi tạo một mảng 10 đối tượng (cố định, có thể mở rộng) của lớp đó để có thể lên lịch cho 10 mã riêng biệt đồng thời. Mỗi khi có sự cập nhật từ firebase thì chương trình sẽ đọc dữ liệu, kiểm tra trường thời gian sau đó tính toán khoảng thời gian chờ (thời gian sẽ được lưu dạng Epoch, kiểu long), lưu lại trường mã hồng ngoại, trường lặp (là chu kỳ lặp lại trong khoảng từ 0-24 giờ, 0 là không lặp, chỉ thực hiện một lần). Và cuối cùng là lên lịch bằng ticker nếu tất cả dữ liệu là hợp lệ. Dưới đây là mã nguồn của lớp:

class SenderSchedule{

public:

SenderSchedule(){};

~SenderSchedule(){};

int pos;

unsigned int duration;

String data;

int loop;

Ticker ticker;

void set(){

ticker.once(duration, std::bind(&SenderSchedule::doSend, this));

}

void stop(){

ticker.detach();

}

void doSend(){

unsigned long \_t = timeClient.getEpochTime() + loop \* 3600;

sendIR(this->data);

if(loop != 0){

Firebase.setFloat("device/sduration/s" + String(pos), \_t);

} else {

Firebase.setFloat("device/sduration/s" + String(pos), 0);

}

};

} schedule[10];

Giải thích:

* Các thuộc tính:
  + *int pos:* xác định vị trí nằm trong mảng 10 phần tử.
  + *unsigned int duration:* khoảng thời gian đến thời điểm phát tín hiệu.
  + *int loop:* chu kỳ lặp tính theo giờ.
  + *String data:* mã tín hiệu hồng ngoại.
  + *Ticker ticker:* ticker để lên lịch hẹn giờ.
* Các phương thức:
  + *set( ):* bắt đầu hẹn giờ, hẹn giờ thực hiện hàm *doSend( )* một lần.
  + *stop( ):* hủy hẹn giờ.
  + *doSend( ):* được gọi khi đến thời điểm đã đặt. Sau khi thực hiện phát tín hiệu xong, kiểm tra biến lặp, nếu có lặp (khác 0) thì sẽ cập nhật lại giá trị thời điểm mới lên firebase và chờ nhận dữ liệu từ firebase để thiết lập lần lặp kế tiếp.

1. **Giao diện hẹn giờ trên ứng dụng Android**

|  |  |
| --- | --- |
| Tại mỗi vị trí trên listview sẽ có thêm nút để cài đặt chức năng hẹn giờ (hình đồng hồ) cho nút điều khiển đó.  Dialog hiện lên để người dùng nhập ngày giờ và chu kì lặp.  Bấm OK: kiểm tra nếu dữ liệu hợp lệ sẽ cập nhật lên firebase. | A screenshot of a cell phone  Description automatically generated A screenshot of a cell phone  Description automatically generated |
| Giao diện dialog để nhập ngày và giờ. Sử dụng DatePickerDialog và TimePickerDialog. | A screenshot of a cell phone  Description automatically generated A picture containing clock, monitor  Description automatically generated |
| Activity hiện danh sách trạng thái của 10 vị trí hẹn giờ.  Thông tin gồm có: tên nút điều khiển (Empty nếu trống), thời gian, ngày và chu kì lặp. | A screenshot of a cell phone  Description automatically generated |

# **Dự kiến tuần tới**

* Tinh chỉnh giao diện ứng dụng, kịch bản giao tiếp giữa các thành phần của hệ thống và kiểm thử
* Hàn module thành mạch nếu đã hoạt động ổn định