TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH : KỸ THUẬT MÁY TÍNH**

**HỆ : ĐẠI HỌC CHÍNH QUY**

**ĐỀ TÀI : BỘ ĐIỀU KHIỂN**

**BLUETOOTH THÔNG MINH**

**THÁI NGUYÊN – 2021**

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

**KHOA ĐIỆN TỬ**

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH KỸ THUẬT MÁY TÍNH**

**Đề tài : BỘ ĐIỀU KHIỂN BLUETOOTH THÔNG MINH**

**SINH VIÊN : PHẠM THỊ PHƯƠNG THẢO**

**LỚP : K52.KMT.01**

**GIÁO VIÊN HƯỚNG DẪN : TS. NGUYỄN VĂN HUY**

**THÁI NGUYÊN – 2021**

|  |  |
| --- | --- |
| **TRƯỜNG ĐHKTCN** | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA ĐIỆN TỬ** | ***Độc lập - Tự do - Hạnh phúc*** |
|  |  |

**NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP**

Sinh viên: Phạm Thị Phương Thảo.

Lớp: K52.KMT.01 Khoá: K52.

Ngành học: Kỹ thuật máy tính.

Giáo viên hướng dẫn: TS. Nguyễn Văn Huy.

**I.** **Tên đề tài tốt nghiệp**: *Bộ điều khiển Bluetooth thông minh*.

**II. Nội dung các phần thuyết minh:**

1. Phân tích mục tiêu, giải pháp thiết kế.
2. Thiết kế sơ đồ nguyên lý hoạt động của hệ thống.
3. Phân tích, lựa chọn linh kiện.
4. Thiết kế mạch điện tử, phần mềm điều khiển.
5. Lập trình và kiểm thử.

**III. Số lượng các phần mềm**:

1 phần mềm điều khiển mạch trên điện thoại.

**IV. Ngày giao nhiệm vụ:** 21/11/2020.

**V. Ngày hoàn thành nhiệm vụ:** 15/01/2021.

|  |  |  |
| --- | --- | --- |
| **BCN KHOA** | **TRƯỞNG BỘ MÔN** | **GIÁO VIÊN HƯỚNG DẪN** |
| *(Ký và ghi rõ họ tên)* | *(Ký và ghi rõ họ tên)* | *(Ký và ghi rõ họ tên)* |

MỤC LỤC

[CHƯƠNG I: GIỚI THIỆU CHUNG. 1](#_Toc63094684)

[1.1. Đặt vấn đề. 1](#_Toc63094685)

[1.2. Đối tượng nghiên cứu. 1](#_Toc63094686)

[1.2.1. Công nghệ Bluetooth. 2](#_Toc63094687)

[1.2.2. Hệ điều hành Android. 11](#_Toc63094688)

[1.3. Mục tiêu của đề tài. 31](#_Toc63094689)

[CHƯƠNG II: THIẾT KẾ NGUYÊN LÝ. 32](#_Toc63094690)

[2.1. Sơ đồ khối. 32](#_Toc63094691)

[2.2. Nguyên lý hoạt động của hệ thống. 32](#_Toc63094692)

[2.3.1. Arduino. 34](#_Toc63094693)

[2.3.2. Module Bluetooth HC06. 39](#_Toc63094694)

[2.3.3. Relay. 41](#_Toc63094695)

[2.3.4. Điện trở. 41](#_Toc63094696)

[2.3.5. Module thời gian thực RTC DS1307. 42](#_Toc63094697)

[2.3.6. Mạch đo điện AC đa năng. 43](#_Toc63094698)

[CHƯƠNG 3: XÂY DỰNG HỆ THỐNG. 45](#_Toc63094699)

[3.1. Thiết kế phần cứng. 45](#_Toc63094700)

[3.1.1. Sơ đồ nguyên lý. 45](#_Toc63094701)

[3.1.2. Sơ đồ giải thuật của Arduino. 46](#_Toc63094702)

[3.2. Thiết kế phần mềm. 47](#_Toc63094703)

[3.2.1. Thiết kế giao diện. 47](#_Toc63094704)

[3.2.2. Sơ đồ giải thuật chương trình trên Android. 49](#_Toc63094705)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN. 50](#_Toc63094706)

[4.1. Kết luận. 50](#_Toc63094707)

[4.2. Hướng phát triển. 51](#_Toc63094708)

[TÀI LIỆU THAM KHẢO 52](#_Toc63094709)

[PHỤ LỤC 53](#_Toc63094710)

**DANH MỤC HÌNH ẢNH**

[*Hình 1: Biểu tượng Bluetooth.* 2](#_Toc63093713)

[*Hình 2: Kết nối các thiết bị thông qua Bluetooth.* 3](#_Toc63093714)

[*Hình 3: Qúa trình hình thành Piconet.* 7](#_Toc63093715)

[*Hình 4: Mô hình truy vấn trong thực tế.* 9](#_Toc63093716)

[*Hình 5: Ứng dụng của Bluetooth.* 11](#_Toc63093717)

[*Hình 6: Biểu tượng Android.* 12](#_Toc63093718)

[*Hình 7: Giao diện Android 5.0 trên điện thoại.* 13](#_Toc63093719)

[*Hình 8: Cấu trúc stack hệ thống Android.* 15](#_Toc63093720)

[*Hình 9: Các trạng thái của tiến trình Android.* 18](#_Toc63093721)

[*Hình 10: Cách xác định Activity.* 19](#_Toc63093722)

[*Hình 11: Cấu trúc MainActivity.java.* 21](#_Toc63093723)

[*Hình 12: Mô tả thời gian sống của Activity.* 23](#_Toc63093724)

[*Hình 13: Phần mềm Android Studio.* 24](#_Toc63093725)

[*Hình 14: Quá trình thông dịch.* 26](#_Toc63093726)

[*Hình 15: JVM cho các nền tảng khác nhau.* 28](#_Toc63093727)

[*Hình 16:**Các loại ứng dụng được phát triển sử dụng Java.* 29](#_Toc63093728)

[*Hình 17: Các phiên bản của Java.* 29](#_Toc63093729)

[*Hình 18: Các thành phần của Java SE Platform.* 30](#_Toc63093730)

[*Hình 19: Sơ đồ tổng quan của hệ thống.* 32](#_Toc63093731)

[*Hình 20: Sơ đồ chân Arduino Mega 2560.* 35](#_Toc63093732)

[*Hình 21: Thông số kỹ thuật của Arduino Mega 2560.* 38](#_Toc63093733)

[*Hình 22: Phần mềm Visual Studio Code.* 39](#_Toc63093734)

[*Hình 23: Module Bluetooth HC-06.* 39](#_Toc63093735)

[*Hình 24: Relay5V.* 41](#_Toc63093736)

[*Hình 25: Các loại điện trở.* 42](#_Toc63093737)

[*Hình 26: Module thời gian thực DS1307.* 42](#_Toc63093738)

[*Hình 27: Mạch đo điện AC đa năng.* 43](#_Toc63093739)

[*Hình 28: Sơ đồ nguyên lý của hệ thống.* 45](#_Toc63093740)

[*Hình 29: Sơ đồ giải thuật của Arduino.* 46](#_Toc63093741)

[*Hình 30: Giao diện phần mềm lập trình Android.* 47](#_Toc63093742)

[*Hình 31: Thiết kế giao diện cho phần mềm trên điện thoại.* 48](#_Toc63093743)

[*Hình 32: Sơ đồ giải thuật chương trình trên Android.* 49](#_Toc63093744)

[*Hình 33: Kết quả.* 50](#_Toc63093745)

**LỜI NÓI ĐẦU**

Ngày nay, xã hội phát triển mạnh mẽ, kỹ thuật ngày càng hiện đại nên nhu cầu về trao đổi thông tin giải trí, nhu cầu về điều khiển các thiết bị từ xa,…ngày càng cao. Và những hệ thống dây cáp phức tạp lại không thể đáp ứng nhu cầu này, nhất là ở những khu vực chật hẹp, những nơi xa xôi, trên các phương tiện vận chuyển,… Vì vậy công nghệ không dây đã ra đời và phát triển mạnh mẽ, tạo rất nhiều thuận lợi cho con người trong đời sống hằng ngày.

Trong những năm gần đây công nghệ truyền nhận dữ liệu không dây đang có những bước phát triển mạnh mẽ, góp công lớn trong việc phát triển các hệ thống điều khiển, giám sát từ xa, đặc biệt là các hệ thống thông minh. Hiện nay, có khá nhiều công nghệ không truyền nhận dữ liệu không dây như RF, Wifi, Bluetooth, NFC,…Trong đó, Bluetooth là một trong những công nghệ được phát triển từ lâu và luôn được cải tiến để nâng cao tốc độ cũng như khả năng bảo mật.

Trên thị trường Việt Nam hiện nay chưa có nhiều sản phẩm điều khiển thiết bị không dây, đa số những sản phẩm hiện có đều là nhập khẩu từ nước ngoài với giá thành cao. Việc nghiên cứu và thiết kế một bộ sản phẩm điều khiển thiết bị không dây có một ý nghĩa lớn, giúp tăng thêm sự lựa chọn cho người sử dụng, sản phẩm được sản xuất trong nước nên giá thành rẻ và góp phần phát triển các hệ thống điều khiển thông minh.

Do đó, em quyết định thực hiện đề tài: “Bộ điều khiển Bluetooth thông minh”. Đề tài ứng dụng công nghệ Bluetooth phổ biến trên nhiều thiết bị, đặc biệt điểm mới của đề tài so với các sản phẩm hiện có là điều khiển thông qua hệ điều hành Android, giúp tận dụng những thiết bị sử dụng hệ điều hành Android có sẵn của người dùng giúp giảm giá thành sản phẩm, ngoài ra với màn hình hiển thị lớn của điện thoại cho phép hiển thị nhiều thông tin hơn, giúp người dùng sử dụng thuận tiện và đỡ mất thời gian hơn.

**LỜI CẢM ƠN**

Trước hết em xin chân thành cảm ơn các thầy cô khoa Điện Tử, đặc biệt là các thầy cô trong bộ môn Kỹ thuật Máy tính đã tận tình chỉ dạy và trang bị cho em những kiến thức cần thiết trong suốt thời gian ngồi trên giảng đường làm nền tảng cho việc thực hiện đồ án này.

Chúng em cũng xin trân trọng cảm ơn thầy giáo **TS Nguyễn Văn Huy** đã tận tình giúp đỡ và giải quyết các khúc mắc để em có thể hoàn thành đồ án.

Tuy nhiên do hạn chế về kinh nghiệm thực tế và thời gian thực hiện nên việc giải quyết đề tài không thể tránh khỏi những thiếu sót. Dó đó rất mong sự chỉ dẫn thêm của thầy cô cũng như những ý kiến, đóng góp của các bạn sinh viên.

Em xin chân thành cảm ơn.

Sinh viên thực hiện

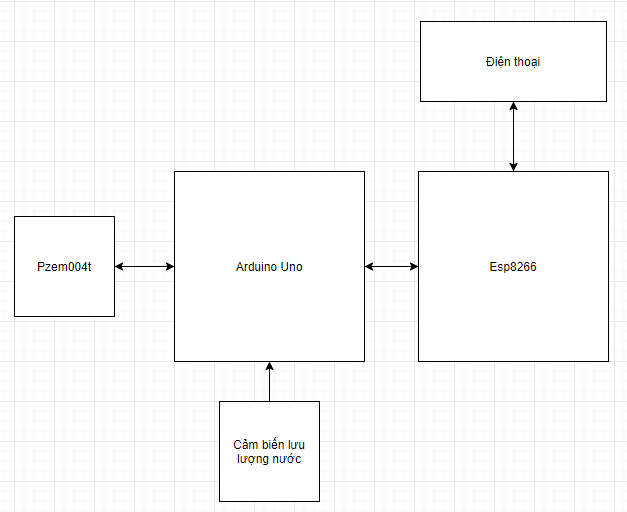
Phạm Thị Phương Thảo.

# CHƯƠNG I: GIỚI THIỆU CHUNG.

# CHƯƠNG II: THIẾT KẾ NGUYÊN LÝ.

## 2.1. Sơ đồ khối.

Hệ thống điều khiển thông minh các thiết bị qua Bluetooth gồm 5 khối chính:



*Hình 19: Sơ đồ tổng quan của hệ thống.*

## 2.2. Nguyên lý hoạt động của hệ thống.

Trong bài sử dụng 2 mạch là esp8266 nodeMCU và Arduino Uno, hệ thống hoạt động theo nguyên lý sau:

* Arduino liên tục đọc cảm biến mỗi giây sau đó gửi dữ liệu sang cho esp8266.ss
* Esp8266 nhận được dữ liệu từ Arduino sẽ tự động ghi vào eeprom.
* Mỗi khi có yêu cầu đọc dữ liệu từ điện thoại, Esp8266 sẽ nhận được lệnh thông qua mqtt và lấy dữ liệu cảm biến đã lưu gửi lên mqtt để phản hồi.
* Khi đó, trên điện thoại sẽ hiển thị số liệu của cảm biến tức là số lượng tiêu thụ điện nước trong tháng.

## 2.3. Lựa chọn linh kiện.

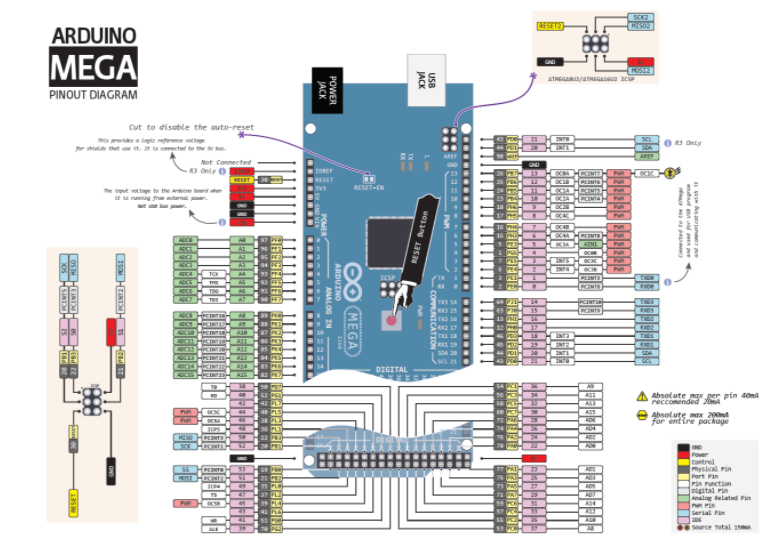
### 2.3.1. Arduino.

Arduino là một board mạch vi xử lý, nhằm xây dựng các ứng dụng tương tác với nhau hoặc với môi trường được thuận lợi hơn. Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM Atmel 32-bit. Những Model hiện tại có thể sử dụng được nhiều ngoại vi, tương thích với nhiều board mở rộng khác nhau

Một mạch Arduino bao gồm một vi điều khiển AVR với nhiều linh kiện bổ sung giúp dễ dàng lập trình và có thể mở rộng với các mạch khác. Một khía cạnh quan trọng của Arduino là các kết nối tiêu chuẩn của nó, cho phép người dùng kết nối với CPU của board với các module thêm vào có thể dễ dàng chuyển đổi, được gọi là shield. Vài shield truyền thông với board Arduino trực tiếp thông qua các chân khác nhau, nhưng nhiều shield được định địa chỉ thông qua serial bus I²C-nhiều shield có thể được xếp chồng và sử dụng dưới dạng song song. Arduino chính thức thường sử dụng các dòng chip megaAVR, đặc biệt là ATmega8, ATmega168, ATmega328, ATmega1280, và ATmega2560. Một vài các bộ vi xử lý khác cũng được sử dụng bởi các mạch Aquino tương thích. Hầu hết các mạch gồm một bộ điều chỉnh tuyến tính 5V và một thạch anh dao động 16 MHz (hoặc bộ cộng hưởng ceramic trong một vài biến thể), mặc dù một vài thiết kế như LilyPad chạy tại 8 MHz và bỏ qua bộ điều chỉnh điện áp onboard do hạn chế về kích cỡ thiết bị. Một vi điều khiển Arduino cũng có thể được lập trình sẵn với một boot loader cho phép đơn giản là upload chương trình vào bộ nhớ flash on-chip, so với các thiết bị khác thường phải cần một bộ nạp bên ngoài. Điều này giúp cho việc sử dụng Arduino được trực tiếp hơn bằng cách cho phép sử dụng 1 máy tính gốc như là một bộ nạp chương trình.

1. **Cấu trúc chi tiết:**

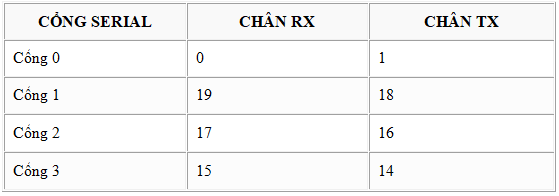
Trong phạm vi đề tài nghiên cứu này, chúng em tập trung vào Aruduino Mega 2560 sử dụng vi điều khiển Aruduino Mega 2560.



*Hình 20: Sơ đồ chân Arduino Mega 2560.*

Arduino Mega 2560 là một vi điều khiển hoạt động dựa trên chip ATmega2560:

* 54 chân digital (trong đó có 15 chân có thể được sủ dụng như những chân PWM là từ chân số 2 → 13 và chân 44 45 46).
* 6 ngắt ngoài: chân 2 (interrupt 0), chân 3 (interrupt 1), chân 18 (interrupt 5), chân 19 (interrupt 4), chân 20 (interrupt 3), and chân 21 (interrupt 2).
* 16 chân vào analog (từ A0 đến A15).
* 4 cổng Serial giao tiếp với phần cứng:

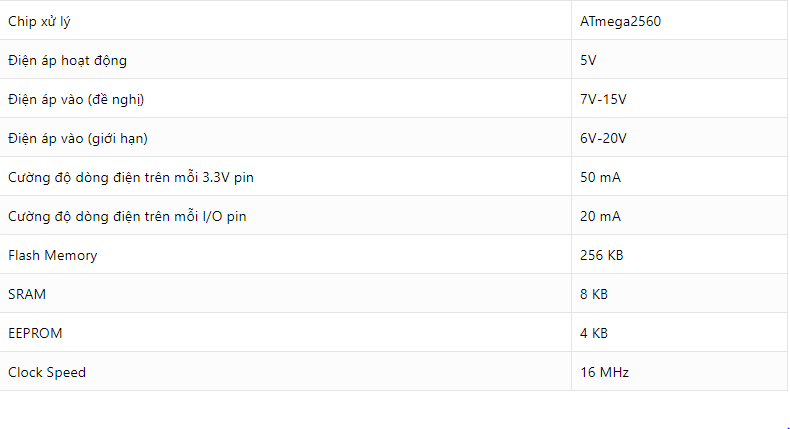


* 1 thạch anh với tần số dao động 16 MHz.
* 1 cổng kết nối USB.
* 1 jack cắm điện.
* 1 đầu ICSP.
* 1 nút reset.

Chi tiết về các chân trên board mạch Arduino Mega2560:

* Chân 5V & 3.3V : Chân này được sử dụng để cung cấp điện áp đầu ra khoảng 5V.
* Chân GND : Có 5 chân nối mass có sẵn trên board Arduino Mega, giúp dễ dàng kết nối nếu thực hiện dự án với nhiều kết nối thiết bị ngoại vi
* Chân reset : Được sử dụng để thiết lập lại board mạch về lại ban đầu. Mức tích cực LOW được thiết lập sẽ reset lại board mạch.
* Chân Vin : Là chân điện áp đầu vào cung cấp cho mạch Arduino Mega, điện áp từ 7V đến 20V. Mặt khác điện áp được cấp bởi jack nguồn DC có thể được lấy thông qua chân này. Tuy nhiên, điện áp đầu ra thông qua chân này đến mạch Arduino sẽ được tự động thiết lập là 5V.
* Chân truyền thông nối tiếp ( Serial Communication ) : RXD và TXD là các chân nối tiếp được sử dụng để truyền và nhận dữ liệu nối tiếp, chân Rx đại diện cho việc truyền dữ liệu còn Tx được sử dụng để nhận dữ liệu. Có tất cả 4 kết hợp các chân nối tiếp này được sử dụng trong đó Serial 0 là chân RX(0) và TX(1), Serial 1là chân TX(18) và RX(19), Serial 2 là chân TX(16) và RX(17), và Serial 3 là chân TX(14) và RX(15).
* Chân Ngắt ngoài ( External Interrupts) : 6 chân được sử dụng để tạo các ngắt ngoài đó là ngắt 0 (chân 0), ngắt 1 (chân 3), ngắt 2 (chân 21), ngắt 3 (chân 20), ngắt 4 (chân 19), ngắt 5 (chân 18). Các chân này tạo ra các ngắt bằng một số cách tức là cung cấp giá trị LOW, tăng hoặc giảm hoặc thay đổi giá trị cho các chân ngắt.
* Đèn LED : Arduino Mega 2560 tích hợp đèn LED trên board mạch kết nối với chân 13. Giá trị HIGH đèn LED được bật và LOW đèn LED tắt. Giúp người lập trình quan sát trực quan khi test, kiểm tra chương trình trên board Arduino
* Chân AREF : Chân tạo điện áp tham chiếu cho đầu vào analogs
* Các chân tương tự ( Analogs) :  Có 16 chân analog được tích hợp trên board Arduino có ký hiệu là A0 đến A15. Điều quan trọng cần lưu ý là tất cả các chân analog này có thể được sử dụng làm chân I / O Digital. Mỗi chân analog đi kèm với độ phân giải 10 bit. Các chân này có thể có điện áp thay đổi tử 0V đến 5V. Tuy nhiên, giá trị trên có thể được thay đổi bằng cách sử dụng hàm ISF và analogReference ().
* Giao tiếp I2C : Hai chân 20 và 21 hỗ trợ giao tiếp I2C trong đó 20 đại diện cho SDA (Dòng dữ liệu nối tiếp chủ yếu được sử dụng để giữ dữ liệu) và 21 đại diện cho SCL (Dòng đồng hồ nối tiếp chủ yếu được sử dụng để cung cấp đồng bộ hóa dữ liệu giữa các thiết bị)
* Truyền thông SPI : Được sử dụng để truyền dữ liệu giữa Arduino và các thiết bị ngoại vi khác. Chân 50 (MISO), Chân51 (MOSI), Chân 52 (SCK), Chân 53 (SS) được sử dụng để liên lạc SPI.

1. **Thông số kỹ thuật:**

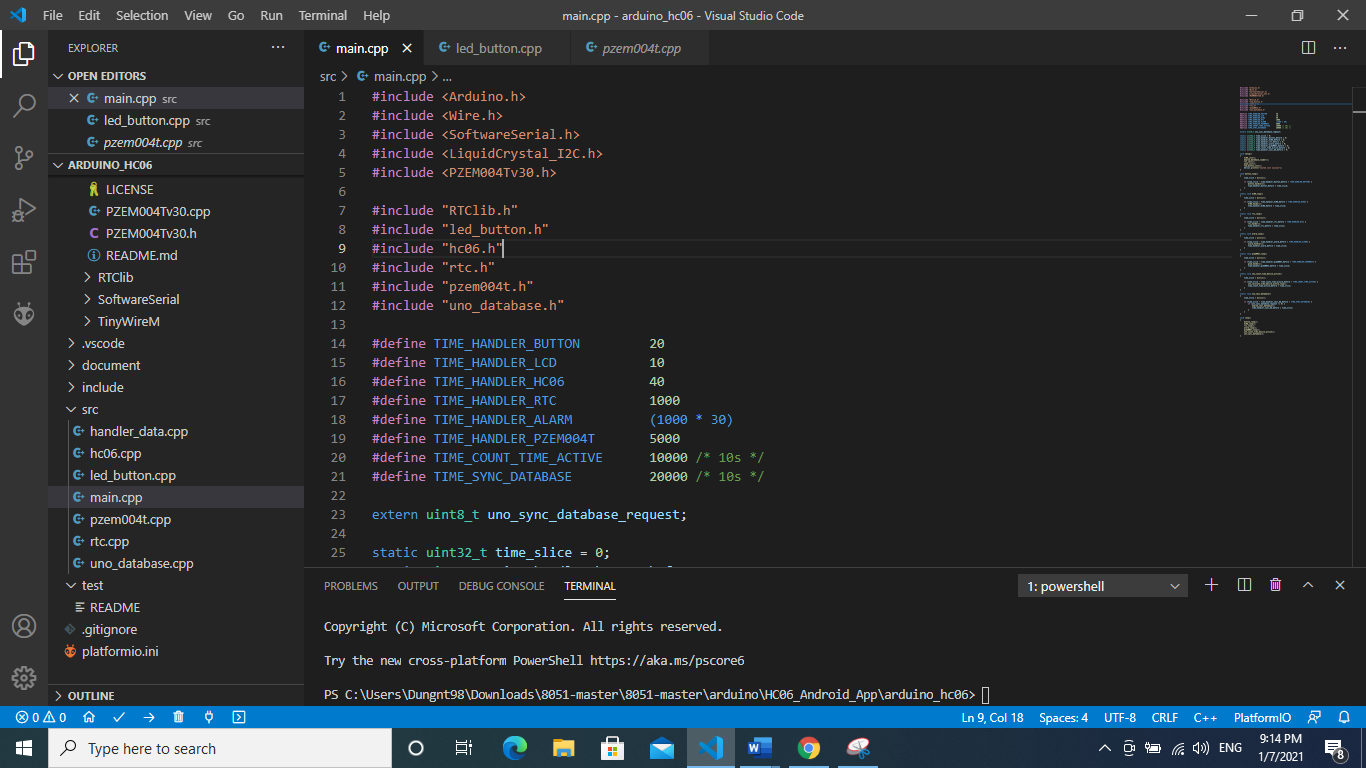


*Hình 21: Thông số kỹ thuật của Arduino Mega 2560.*

1. **Lập trình cho Arduino:**

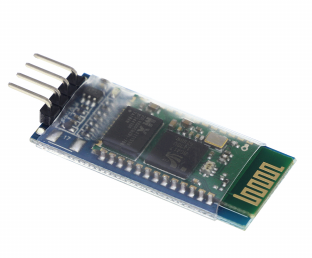
Các thiết bị dựa trên nền tảng Arduino được lập trình bằng ngôn riêng. Ngôn ngữ này dựa trên ngôn ngữ Wiring được viết cho phần cứng nói chung. Và Wiring lại là một biến thể của C/C++. Một số người gọi nó là Wiring, một số khác thì gọi là C hay C/C++. Riêng đội ngũ phát triển Arduino gọi là “*ngôn ngữ Arduino*”. Ngôn ngữ Arduino bắt nguồn từ C/C++ phổ biến hiện nay do đó rất dễ học, dễ hiểu.

Để lập trình cũng như gửi lệnh và nhận tín hiệu từ mạch Arduino, nhóm phát triển dự án này đã cũng cấp đến cho người dùng một môi trường lập trình Arduino được gọi là Visual Studio Code như hình dưới đây.



*Hình 22: Phần mềm Visual Studio Code.*

### 2.3.2. Esp8266 Node MCU



*Hình 23: Module Bluetooth HC-06.*

1. Lý do chọn Module HC-06:

Với thiết kế nhỏ gọn, tiện lợi, giao tiếp với vi điều khiển chỉ bằng 2 chân (Tx và Rx), module bluetooth HC06 sẽ giúp người sử dụng thực hiện các dự án truyền dẫn và điều khiển từ xa một cách dễ dàng.

Điểm khác biệt so với HC05 đó là HC06 chỉ có thể chạy được 1 chế độ Slave (khác với HC05 có thể hoạt động với chế độ Mater hoặc Slave). Điều này có nghĩa là người sử dụng không thể chủ động kết nối từ vi điều khiển đến các thiết bị ngoại vi. Mà cách kết nối là phải sử dụng thiết bị ngoại vi (điện thoại thông minh, máy tính laptop) để dò tín hiệu kêt nối Buletooth mà HC06 phát ra. Sau khi pair thành công người sử dụng có thể gửi tín hiệu từ vi điều khiển đến các thiết bị ngoại vi này, và ngược lại.

1. Thông số kỹ thuật:

* Điện áp hoạt động: 3V3-5V DC.
* Dòng điện tiêu thụ: 20-30mA.
* Nhiệt độ hoạt động: -20~75°C.
* Sử dụng chip: CSR Bluetooth V2.0.
* Cấu hình **Slave** mặc định, không thay đổi được.
* Hỗ trợ tốc độ baud: 200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
* Kích thước: 28x15x2,35mm.
* Giao tiếp: UART (TX,RX).
* Tốc độ:

+ Bất đồng bộ: 2.1Mbps(Max)/160kbps.

+ Đồng bộ: 1Mbps/1Mbps.

* Bảo mật: mã hóa và chứng thực.
* Cấu hình mặc định:

+ Tốc độ baud 9600, N, 8, 1.

+ Mật khẩu: 1234.

1. Nguyên lý hoạt động:

Module này gồm 4 chân GND, VCC, TX, RX Khi kết nối chỉ cần nối chân TX với chân 0(RX) và chân RX nối với chân 1 (chân TX) trên Arduino uno r3 sau đó có thể lập trình gửi và nhận dữ liệu như 1 cổng Serial thông thường.

### 2.3.6. Mạch đo điện AC đa năng.



*Hình 27: Mạch đo điện AC đa năng.*

Mạch đo điện AC đa năng 100A giao tiếp UART PZEM-004T được sử dụng để đo và theo dõi gần như hoàn toàn các thông số về điện năng AC của mạch điện như điện áp hoạt động, dòng tiêu thụ, công suất và năng lượng tiêu thụ, mạch sử dụng giao tiếp UART dễ dàng kết nối truyền dữ liệu tới Vi điều kiển hoặc máy tính, thích hợp cho các ứng dụng theo dõi năng lượng, IoT…

Mạch đo điện AC đa năng 100A giao tiếp UART PZEM-004T nhỏ gọn, dễ lắp đặt, sử dụng cách đo dòng cách ly an toàn và khả năng đo dòng lên đến 100A, mạch có chất lượng gia công và linh kiện tốt, độ bền cao.

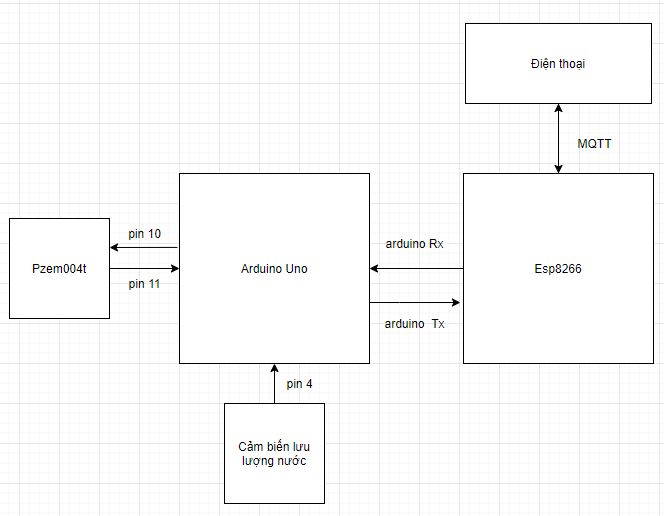
**Thông số kỹ thuật:**

* Điện áp đo và hoạt động: 80 ~ 260VAC / 50 - 60Hz, sai số 0.01
* Dòng điện đo và hoạt động: 0 ~ 100A, sai số 0.01
* Công suất đo và hoạt động: 0 ~ 26000W
* Năng lượng đo và hoạt động: 0~9999kWh.
* Giao tiếp UART mức logic TTL 5VDC baudrate mặc định 9600, 8, 1.
* Có opto cách ly an toàn giữa mạch đo và mạch nhận tín hiệu UART.
* Lưu giữ thông số năng lượng tiêu thụ trong bộ nhớ.
* Kích thước: 30 x 75 mm.

# CHƯƠNG 3: XÂY DỰNG HỆ THỐNG.

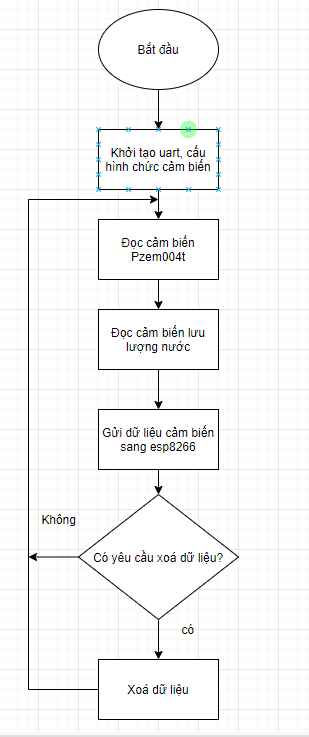
## 3.1. Thiết kế phần cứng.

### 3.1.1. Sơ đồ nguyên lý.



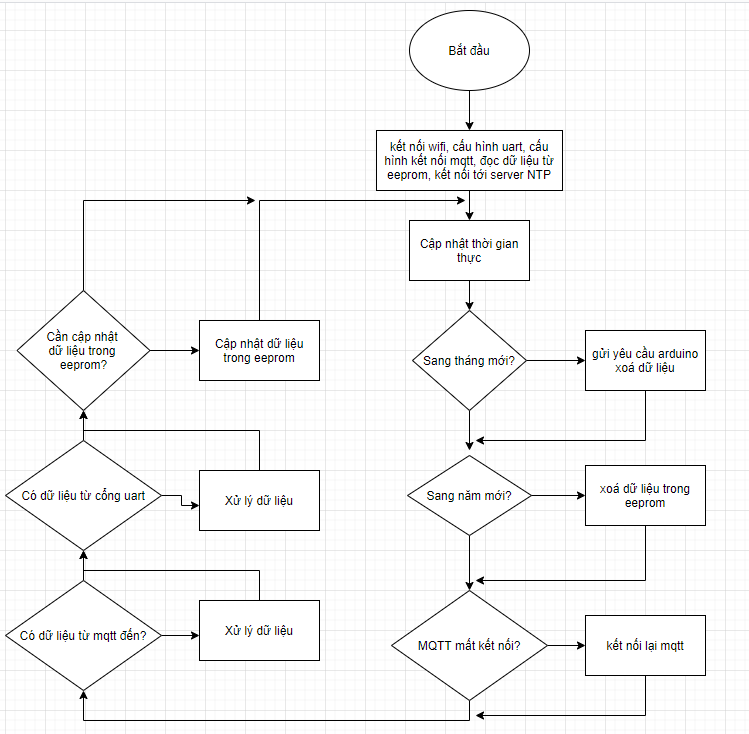
*Hình 28: Sơ đồ nguyên lý của hệ thống.*

### 3.1.2. Sơ đồ giải thuật của Arduino.



*Hình 29: Sơ đồ giải thuật của Arduino.*

### 3.1.2. Sơ đồ giải thuật của Esp8266.

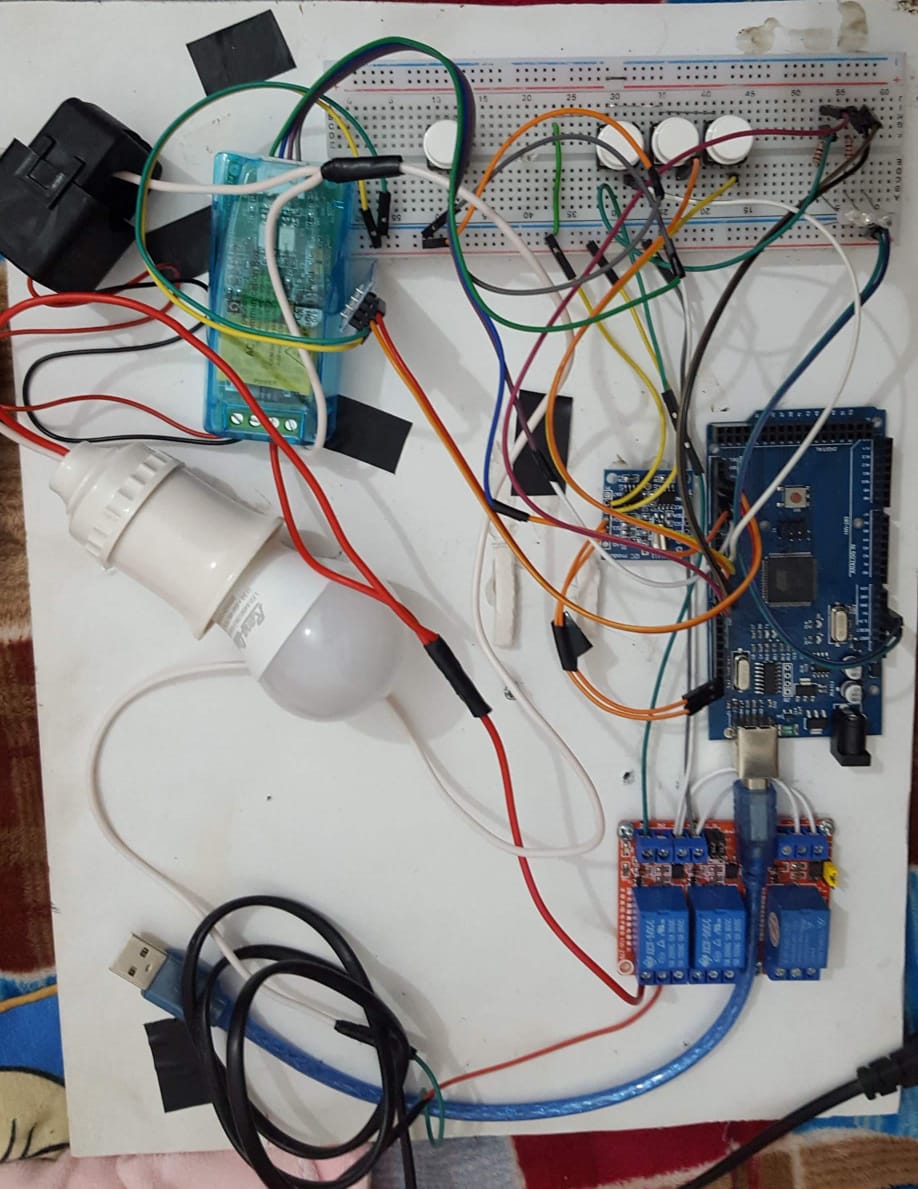


*Hình 29: Sơ đồ giải thuật của Esp8266.*

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.

## 4.1. Kết luận.

Sau 2 tháng nỗ lực em đã hoàn thành đề tài “Xây dựng hệ thống điều khiển thiết bị thông minh qua Bluetooth” đúng thời gian quy định. Về mặt cơ bản đã giải quyết được các vấn đề đặt ra. Đây là một đề tài mang tính tổng hợp bao gồm cả thiết kế thi công mạch và kĩ thuật lập trình.



*Hình 33: Kết quả.*

Ưu điểm:

* Thiết bị hoạt động tốt, các module nhỏ gọn.
* Giá thành các thiết bị rẻ, phù hợp túi tiền.
* Phần cứng nhỏ gọn, phần mềm đơn giản dễ hiểu, giao diện lập trình quen thuộc.
* Giao diện Android dễ dùng.

Nhược điểm:

* Chưa trải qua lắp đặt ngoài thực tế.
* Giao diện Android chưa phong phú.
* Cự ly truyền dữ liệu giữa điện thoại đến module Bluetooth còn hạn chế. Do chất lượng linh kiện không đảm bảo đúng thông số trong Datasheet nên có 1 số lỗi nhỏ như thời gian chưa hoạt động chính xác cũng như công suất không đảm bảo.

## 4.2. Hướng phát triển.

- Phát triển app kiểm soát số lần đóng mở thiết bị.

- Kết hợp nhiều mô hình lại với nhau để mở rộng phạm vi và thiết bị điều khiển.

- Có thể xây dựng hệ thống mạng để quản lý thiết bị và hoạt động của hệ thống.

- Do cự ly giữa điện thoại và Bluetooth còn hạn chế nên có thể chuyển nghiên cứu sang module Wifi hoặc chạy trên server…

# TÀI LIỆU THAM KHẢO

* Bài giảng nhúng, bộ môn Kỹ thuật máy tính- Trường Đại học Kỹ thuật Công nghiệp – Đại học Thái Nguyên.
* Website : Wikipedia tiếng Việt.

<https://vi.wikipedia.org/wiki>

* <http://dammedientu.vn/hoc-avr-bai-7-giao-tiep-ic-thoi-gian-thuc-ds1307/>
* <http://arduino.vn/bai-viet/953-huong-dan-thiet-lap-chi-tiet-cho-module-bluetooth>

# PHỤ LỤC

//*main*

#include <Arduino.h>

#include <Wire.h>

#include <SoftwareSerial.h>

#include <LiquidCrystal\_I2C.h>

#include <PZEM004Tv30.h>

#include "RTClib.h"

#include "led\_button.h"

#include "hc06.h"

#include "rtc.h"

#include "pzem004t.h"

#include "uno\_database.h"

#define TIME\_HANDLER\_BUTTON         20

#define TIME\_HANDLER\_LCD            10

#define TIME\_HANDLER\_HC06           40

#define TIME\_HANDLER\_RTC            1000

#define TIME\_HANDLER\_ALARM          (1000 \* 30)

#define TIME\_HANDLER\_PZEM004T       5000

#define TIME\_COUNT\_TIME\_ACTIVE      10000 /\* 10s \*/

#define TIME\_SYNC\_DATABASE          20000 /\* 10s \*/

extern uint8\_t uno\_sync\_database\_request;

static uint32\_t time\_slice = 0;

static uint32\_t time\_handler\_button\_before = 0;

static uint32\_t time\_handler\_hc06\_before = 0;

static uint32\_t time\_handler\_rtc\_before = 0;

static uint32\_t time\_handler\_alarm\_before = 0;

static uint32\_t time\_handler\_pzem004t\_before = 0;

static uint32\_t time\_count\_time\_active\_before = 0;

static uint32\_t time\_handler\_sync\_db\_before = 0;

void setup()

{

    hc06\_init();

    eeprom\_database\_loader();

    rtc\_init();

    pzem\_init();

    led\_button\_init();

    Serial.println("system init success");

}

void button\_loop()

{

    time\_slice = millis();

    if (time\_slice - time\_handler\_button\_before > TIME\_HANDLER\_BUTTON) {

        button\_handler();

        time\_handler\_button\_before = time\_slice;

    }

}

static void hc06\_loop()

{

    time\_slice = millis();

    if (time\_slice - time\_handler\_hc06\_before > TIME\_HANDLER\_HC06) {

        hc06\_hander();

        time\_handler\_hc06\_before = time\_slice;

    }

}

static void rtc\_loop()

{

    time\_slice = millis();

    if (time\_slice - time\_handler\_rtc\_before > TIME\_HANDLER\_RTC) {

        rtc\_hander();

        time\_handler\_rtc\_before = time\_slice;

    }

}

static void alarm\_loop()

{

    time\_slice = millis();

    if (time\_slice - time\_handler\_alarm\_before > TIME\_HANDLER\_ALARM) {

        alarm\_handler();

        time\_handler\_alarm\_before = time\_slice;

    }

}

static void pzem004t\_loop()

{

    time\_slice = millis();

    if (time\_slice - time\_handler\_pzem004t\_before > TIME\_HANDLER\_PZEM004T) {

        pzem\_hander();

        time\_handler\_pzem004t\_before = time\_slice;

    }

}

static void uno\_count\_time\_device\_active()

{

    time\_slice = millis();

    if (time\_slice - time\_count\_time\_active\_before > TIME\_COUNT\_TIME\_ACTIVE) {

        uno\_caculate\_time\_device\_active\_loop();

        time\_count\_time\_active\_before = time\_slice;

    }

}

static void uno\_sync\_database()

{

    time\_slice = millis();

    if (time\_slice - time\_handler\_sync\_db\_before > TIME\_SYNC\_DATABASE) {

        if (uno\_sync\_database\_request != 0) {

            eeprom\_sync\_database();

            time\_handler\_sync\_db\_before = time\_slice;

        }

    }

}

void loop()

{

    button\_loop();

    hc06\_loop();

    rtc\_loop();

    alarm\_loop();

    pzem004t\_loop();

    uno\_count\_time\_device\_active();

    uno\_sync\_database();

}

//handler data

#include "handler\_data.h"

#include <stdlib.h>

#include "m\_typedef.h"

#include "led\_button.h"

#include "uno\_database.h"

#include "rtc.h"

extern device\_info\_t m\_device;

extern uint8\_t uno\_sync\_database\_request;

/\* {"cmd\_type":0; "cmd":y} \*/

static void uno\_handler\_control\_io(JsonDocument &\_doc)

{

    uint8\_t m\_cmd = \_doc["cmd"];

    control\_device(m\_cmd);

}

/\* {cmd\_type:1, "cmd":1, "hour":10, "minutes":10, "second":0} \*/

static void uno\_handler\_set\_alarm(JsonDocument &\_doc)

{

    char respond[256];

    uint8\_t cmd = \_doc["cmd"];

    if (m\_device.alarm\_is\_set >= MAX\_CMD\_ALARM) {

        memset(respond, 0, sizeof(respond));

        sprintf(respond, "{\"cmd\_type\":%d, \"res\":Fail}\n", SET\_ALARM);

        uno\_respond\_app(respond);

        return;

    }

    memset(respond, 0, sizeof(respond));

    sprintf(respond, "cmd: %d\n", cmd);

    Serial.println(respond);

    /\* Do set time alarm here \*/

    m\_device.m\_time\_alarm[m\_device.alarm\_is\_set].m\_time.tm\_hour = \_doc["hour"];

    m\_device.m\_time\_alarm[m\_device.alarm\_is\_set].m\_time.tm\_min = \_doc["minutes"];

    m\_device.m\_time\_alarm[m\_device.alarm\_is\_set].m\_time.tm\_sec = \_doc["second"];

    m\_device.m\_time\_alarm[m\_device.alarm\_is\_set].m\_cmd = cmd;

    uno\_sync\_database\_request = 1;

    Serial.print("alarm\_is\_set!");

    Serial.print("cmd: ");

    Serial.println(m\_device.m\_time\_alarm[m\_device.alarm\_is\_set].m\_cmd);

    m\_device.alarm\_is\_set ++;

    memset(respond, 0, sizeof(respond));

    sprintf(respond, "{\"cmd\_type\":%d, \"res\":OK}\n", SET\_ALARM);

    uno\_respond\_app(respond);

}

/\* {cmd\_type:2, "cmd":x, "day":11, "month":11, "year":2020, "hour":10, "minutes":10, "second":0} \*/

void uno\_handler\_remove\_alarm(JsonDocument &\_doc)

{

    char respond[256];

    /\* clear alarm \*/

    m\_device.alarm\_is\_set = 0;

    memset(m\_device.m\_time\_alarm, 0, sizeof(m\_device.m\_time\_alarm));

    uno\_sync\_database\_request = 1;

    memset(respond, 0, sizeof(respond));

    sprintf(respond, "{\"cmd\_type\":%d, \"res\":OK}\n", RESET\_ALARM);

    uno\_respond\_app(respond);

}

/\* {"cmd\_type":3} \*/

void uno\_handler\_query\_info(void)

{

    report\_current\_state(3);

}

/\* {"cmd\_type":4} \*/

static void uno\_handler\_query\_time\_active\_one\_day()

{

    uno\_get\_time\_active\_on\_day();

}

/\* {"cmd\_type":5} \*/

static void uno\_handler\_query\_time\_active\_one\_week()

{

    uno\_get\_time\_active\_in\_week();

}

/\* {"cmd\_type":6} \*/

static void uno\_handler\_query\_time\_active\_one\_month()

{

    uno\_get\_time\_active\_in\_month();

}

/\* {"cmd\_type":7} \*/

static void uno\_handler\_get\_time\_alarm()

{

    uno\_get\_time\_alarm\_was\_set();

}

/\* {"cmd\_type":x; "cmd":y} \*/

void handler\_data(char\* command)

{

    if (command == NULL) {

        return;

    }

    DynamicJsonDocument doc(180);

    DeserializationError error = deserializeJson(doc, command);

    if (error) {

        Serial.print(F("DeserializeJson() failed\n"));

        Serial.println(error.c\_str());

        return;

    }

    Serial.println(command);

    uint8\_t cmd\_type = (uint8\_t)doc["cmd\_type"];

    Serial.print("cmd\_type: ");

    Serial.println(cmd\_type);

    switch (cmd\_type) {

    case CONTROL\_IO: {

        uno\_handler\_control\_io(doc);

    } break;

    case SET\_ALARM: {

        uno\_handler\_set\_alarm(doc);

    } break;

    case RESET\_ALARM: {

        uno\_handler\_remove\_alarm(doc);

    } break;

    case QUERY\_INFOM: {

        uno\_handler\_query\_info();

    } break;

    case QUERY\_TIME\_DAY: {

        uno\_handler\_query\_time\_active\_one\_day();

    } break;

    case QUERY\_TIME\_WEEK: {

        uno\_handler\_query\_time\_active\_one\_week();

    } break;

    case QUERY\_TIME\_MONTH: {

        uno\_handler\_query\_time\_active\_one\_month();

    } break;

    case GET\_TIME\_ALARM: {

        uno\_handler\_get\_time\_alarm();

    } break;

    default:

        Serial.println("Cmd\_type was not match!");

        break;

    }

}

// điều khiển HC06

#include "hc06.h"

#include "m\_typedef.h"

#include "handler\_data.h"

#define HC06    Serial1

char hc06\_rx\_queue[300];

uint16\_t p\_hc06\_rx\_data = 0;

// SoftwareSerial mySerial(12, 13); // RX, TX

void hc06\_init(void)

{

    Serial.begin(9600);

    HC06.begin(9600);

    // mySerial.begin(9600);

}

void hc06\_hander(void)

{

    char ch;

    p\_hc06\_rx\_data = 0;

    memset(hc06\_rx\_queue, 0, sizeof(hc06\_rx\_queue));

    if (HC06.available() > 0) {

        while (HC06.available() > 0) {

            ch = HC06.read();

            hc06\_rx\_queue[p\_hc06\_rx\_data] = ch;

            p\_hc06\_rx\_data ++;

            delay(1); /\* Delay for wait data ready on serial port \*/

        }

        Serial.print("recieved: ");

        // Serial.println(hc06\_rx\_queue);

        /\* Handler data recieved \*/

        handler\_data(hc06\_rx\_queue);

        p\_hc06\_rx\_data = 0;

        memset(hc06\_rx\_queue, 0, sizeof(hc06\_rx\_queue));

    }

}

// led\_ button

#include "led\_button.h"

#include "m\_typedef.h"

#include "rtc.h"

#include "uno\_database.h"

#include "m\_typedef.h"

extern device\_info\_t m\_device;

extern uint8\_t uno\_sync\_database\_request;

static uint16\_t time\_button\_press[NUMBER\_BUTTON] = {0, 0, 0};

static uint8\_t m\_buttons[NUMBER\_BUTTON] = {BUTTON1\_PIN, BUTTON2\_PIN, BUTTON3\_PIN};

static uint8\_t m\_leds[NUMBER\_LED] = {LED1\_PIN, LED2\_PIN, LED3\_PIN};

void gpio\_on(uint8\_t pin)

{

    digitalWrite(pin, HIGH);

}

void gpio\_off(uint8\_t pin)

{

    digitalWrite(pin, LOW);

}

void gpio\_toggle(uint8\_t pin)

{

    /\* toggle value pin \*/

    digitalWrite(pin, !digitalRead(pin));

    /\* Update state and report \*/

    report\_current\_state(0);

    uno\_sync\_database\_request = 1;

}

static void blink\_led\_status()

{

    uint8\_t time\_delay = 100;

    uint8\_t pin = LED\_STATUS;

    digitalWrite(LED\_STATUS, HIGH);

    delay(time\_delay);

    digitalWrite(pin, LOW);

    delay(time\_delay);

    digitalWrite(pin, HIGH);

    delay(time\_delay);

    digitalWrite(pin, LOW);

    delay(time\_delay);

    digitalWrite(pin, HIGH);

    delay(time\_delay);

    digitalWrite(pin, LOW);

    delay(time\_delay);

    digitalWrite(pin, HIGH);

    delay(time\_delay);

    digitalWrite(pin, LOW);

    delay(time\_delay);

}

void control\_device(uint8\_t cmd)

{

    switch (cmd)

    {

    case CH1\_ON: {

        gpio\_on(LED1\_PIN);

    } break;

    case CH1\_OFF: {

        gpio\_off(LED1\_PIN);

    } break;

    case CH2\_ON: {

        gpio\_on(LED2\_PIN);

    } break;

    case CH2\_OFF: {

        gpio\_off(LED2\_PIN);

    } break;

    case CH3\_ON: {

        gpio\_on(LED3\_PIN);

    } break;

    case CH3\_OFF: {

        gpio\_off(LED3\_PIN);

    } break;

    default: {

        Serial.print("Control invalid pin\n");

    } break;

    }

    /\* Update and Report information of this device \*/

    report\_current\_state(0);

    uno\_sync\_database\_request = 1;

}

hc06\_ctrl\_t uno\_get\_cmd\_for\_pin(uint8\_t index, uint8\_t is\_turn\_on)

{

    if (index == 0 && is\_turn\_on) {

        return CH1\_ON;

    } else if (index == 0 && !is\_turn\_on) {

        return CH1\_OFF;

    } else if (index == 1 && is\_turn\_on) {

        return CH2\_ON;

    } else if (index == 1 && !is\_turn\_on) {

        return CH2\_OFF;

    } else if (index == 2 && is\_turn\_on) {

        return CH3\_ON;

    } else if (index == 2 && !is\_turn\_on) {

        return CH3\_OFF;

    }

    /\* error \*/

    return CH3\_OFF;

}

extern uint8\_t alarm\_is\_set;

extern m\_alarm\_t m\_time\_alarm[MAX\_CMD\_ALARM];

extern uint8\_t uno\_sync\_database\_request;

static void scan\_button\_handler(uint8\_t button\_index)

{

   if (!digitalRead(m\_buttons[button\_index])) /\* press \*/

   {

        time\_button\_press[button\_index] += TIME\_SLICE\_TO\_READ\_BUTTON;

        if (time\_button\_press[button\_index] > OS\_BTN\_IS\_PRESS\_TIME\_MAX) {

            blink\_led\_status();

        }

   } else { /\* realse \*/

        if (time\_button\_press[button\_index] >= OS\_BTN\_IS\_PRESS\_TIME\_MIN

            && time\_button\_press[button\_index] <= OS\_BTN\_IS\_PRESS\_TIME\_MAX)

        {

            if (button\_index < 3 ) /\* Just have 3 leds \*/

            {

                Serial.println("Button was press");

                /\* Control output \*/

                /\* Send message to bluetooth \*/

                gpio\_toggle(m\_leds[button\_index]);

            }

        } else if (time\_button\_press[button\_index] > OS\_BTN\_IS\_PRESS\_TIME\_MAX)

        {

            /\* handler button remove alarm \*/

            Serial.println("Button was hold");

            /\* clear alarm \*/

            // m\_device.alarm\_is\_set = 0;

            // memset(m\_device.m\_time\_alarm, 0, sizeof(m\_device.m\_time\_alarm));

            memset(&m\_device, 0, sizeof(m\_device));

            Serial.println("clear alarm!");

        }

        time\_button\_press[button\_index] = 0;

   }

}

static void button\_init(void)

{

    int i;

    for(i=0; i < NUMBER\_BUTTON; i++)

    {

        pinMode(m\_buttons[i], INPUT\_PULLUP);

    }

}

static void led\_init(void)

{

    int i;

    for (i=0; i < NUMBER\_LED; i++)

    {

        pinMode(m\_leds[i], OUTPUT);

    }

    pinMode(LED\_STATUS, OUTPUT);

    /\* Turn off all \*/

    for (i=0; i < NUMBER\_LED; i++)

    {

        digitalWrite(m\_leds[i], LOW);

    }

    digitalWrite(LED\_STATUS, LOW);

}

void led\_button\_init()

{

    button\_init();

    led\_init();

}

void button\_handler(void)

{

    scan\_button\_handler(BUTTON1);

    scan\_button\_handler(BUTTON2);

    scan\_button\_handler(BUTTON3);

}

// điều khiển pzem004

#include "pzem004t.h"

#include "led\_button.h"

#include "m\_typedef.h"

PZEM004Tv30 pzem(&Serial2); /\* Cam Cheo Day \*/

float voltage;

float current;

float power;

float energy;

float frequency;

float pf;

void pzem\_init(void)

{

    pzem\_reader();

    pzem\_show();

}

void pzem\_reader(void)

{

    if ((digitalRead(LED1\_PIN) | digitalRead(LED2\_PIN) | digitalRead(LED3\_PIN))) {

        voltage = pzem.voltage();

        current = pzem.current();

        power = pzem.power();

        energy = pzem.energy();

        frequency = pzem.frequency();

        pf = pzem.pf();

    }

}

void pzem\_show(void)

{

    if (voltage != NAN) {

        Serial.print("Voltage: ");

        Serial.print(voltage);

        Serial.println("V");

    } else {

        Serial.println("Error reading voltage");

    }

    if(current != NAN){

        Serial.print("Current: ");

        Serial.print(current);

        Serial.println("A");

    } else {

        Serial.println("Error reading current");

    }

    if(current != NAN){

        Serial.print("Power: ");

        Serial.print(power);

        Serial.println("W");

    } else {

        Serial.println("Error reading power");

    }

    if(current != NAN){

        Serial.print("Energy: ");

        Serial.print(energy,3);

        Serial.println("kWh");

    } else {

        Serial.println("Error reading energy");

    }

    if(current != NAN){

        Serial.print("Frequency: ");

        Serial.print(frequency, 1);

        Serial.println("Hz");

    } else {

        Serial.println("Error reading frequency");

    }

    if(current != NAN){

        Serial.print("PF: ");

        Serial.println(pf);

    } else {

        Serial.println("Error reading power factor");

    }

    Serial.println();

}

void pzem\_hander(void)

{

    pzem\_reader();

    // pzem\_show();

}

// điều khiển RTC

#include "rtc.h"

#include <time.h>

#include "m\_typedef.h"

#include "led\_button.h"

#include "uno\_database.h"

extern device\_info\_t m\_device;

extern m\_user\_habit\_trace\_t user\_habit[NUMBER\_CHANNEL];

RTC\_DS1307 rtc;

DateTime now;

struct tm m\_time\_local;

uint8\_t alarm\_is\_set = 0;

struct tm temp\_time\_start[NUMBER\_CHANNEL];

uint16\_t temp\_time\_start\_total\_time\_active[NUMBER\_CHANNEL]; /\* minutes \*/

void rtc\_init(void)

{

    if (Serial) {

        if (! rtc.begin()) {

            Serial.print("Couldn't find RTC");

            while (1);

        }

        if (!rtc.isrunning()) {

            Serial.print("RTC is NOT running!");

            Serial.println();

        }

        rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

    }

}

static void time\_alarm() {

    uint8\_t i;

    for (i = 0; i < MAX\_CMD\_ALARM; i++) {

        if (now.hour() == m\_device.m\_time\_alarm[i].m\_time.tm\_hour

            && now.minute() == m\_device.m\_time\_alarm[i].m\_time.tm\_min)

        {

            Serial.print("Do Alarm Hanlder with cmd: ");

            Serial.println(m\_device.m\_time\_alarm[i].m\_cmd);

            control\_device(m\_device.m\_time\_alarm[i].m\_cmd);

        } else {

            Serial.println(m\_device.m\_time\_alarm[i].m\_time.tm\_hour);

            Serial.println(m\_device.m\_time\_alarm[i].m\_time.tm\_min);

        }

    }

}

static void control\_with\_users\_habit() {

    /\* If device active time was not pass time to get user's habit \*/

    if  (m\_device.number\_days\_dev\_active < USER\_HABIT\_TRACE\_DAYS) {

        return;

    }

    uint8\_t i;

    for (i = 0; i < NUMBER\_CHANNEL; i++) {

        if (m\_device.users\_habit\_is\_set[i]) {

            if (now.hour() == m\_device.users\_habit\_alarm[i].m\_time.tm\_hour

                && now.minute() == m\_device.users\_habit\_alarm[i].m\_time.tm\_min)

            {

                control\_device(m\_device.users\_habit\_alarm[i].m\_cmd); /\* turn on channel \*/

            }

        }

    }

}

extern uint8\_t uno\_sync\_database\_request;

/\* run each 10 second \*/

void uno\_caculate\_time\_device\_active\_loop(void)

{

    uint8\_t dev\_index;

    /\* CHANNEL1 \*/

    dev\_index = CHANNEL1;

    if (digitalRead(LED1\_PIN)) {

        /\* Count total time active on day \*/

        m\_device.time\_active\_on\_day[dev\_index].tm\_sec += 10; /\* 10s \*/

        if (m\_device.time\_active\_on\_day[dev\_index].tm\_sec >= 60) { /\* minutes \*/

            m\_device.time\_active\_on\_day[dev\_index].tm\_min ++;

            m\_device.time\_active\_in\_week\_tm\_min[dev\_index] ++;

            m\_device.time\_active\_in\_month\_tm\_min[dev\_index] ++;

            m\_device.time\_active\_on\_day[dev\_index].tm\_sec = 0;

            if (m\_device.time\_active\_on\_day[dev\_index].tm\_min >= 60) { /\* hours \*/

                m\_device.time\_active\_on\_day[dev\_index].tm\_min = 0;

                /\* Count time active on day \*/

                m\_device.time\_active\_on\_day[dev\_index].tm\_hour ++;

                /\* Count time active on week \*/

                m\_device.time\_active\_in\_week\_tm\_hour[dev\_index] ++;

                /\* Count time active on month \*/

                m\_device.time\_active\_in\_month\_tm\_hour[dev\_index] ++;

            }

            /\* Count time for tracking user's habit \*/

            if (user\_habit[dev\_index].is\_tracking\_time[dev\_index]) {

                temp\_time\_start\_total\_time\_active[dev\_index] ++; /\* 1 minutes \*/

            }

        }

        /\* Tracking user's habit \*/

        if (0 == user\_habit[dev\_index].is\_tracking\_time[dev\_index]) {

            user\_habit[dev\_index].is\_tracking\_time[dev\_index] = 1; /\* start tracking \*/

            /\* Get time device turn on \*/

            temp\_time\_start[dev\_index].tm\_hour = now.hour();

            temp\_time\_start[dev\_index].tm\_min = now.minute();

        }

        /\* sync database \*/

        uno\_sync\_database\_request = 1;

    }

    else {

        user\_habit[dev\_index].is\_tracking\_time[dev\_index] = 0; /\* stop tracking \*/

        /\* Check time device was used to get user's habit \*/

        if (temp\_time\_start\_total\_time\_active[dev\_index] > user\_habit[dev\_index].total\_time\_active\_previous[m\_device.number\_days\_dev\_active]) {

            /\* Store user's using device time \*/

            user\_habit[dev\_index].total\_time\_active\_previous[m\_device.number\_days\_dev\_active] = temp\_time\_start\_total\_time\_active[dev\_index];

            user\_habit[dev\_index].time\_start\_previous[m\_device.number\_days\_dev\_active].tm\_hour = temp\_time\_start[dev\_index].tm\_hour;

            user\_habit[dev\_index].time\_start\_previous[m\_device.number\_days\_dev\_active].tm\_min = temp\_time\_start[dev\_index].tm\_min;

        }

    }

    /\* CHANNEL2 \*/

    dev\_index = CHANNEL2;

    if (digitalRead(LED2\_PIN)) {

        /\* Count total time active on day \*/

        m\_device.time\_active\_on\_day[dev\_index].tm\_sec += 10; /\* 10s \*/

        if (m\_device.time\_active\_on\_day[dev\_index].tm\_sec >= 60) { /\* minutes \*/

            m\_device.time\_active\_on\_day[dev\_index].tm\_min ++;

            m\_device.time\_active\_in\_week\_tm\_min[dev\_index] ++;

            m\_device.time\_active\_in\_month\_tm\_min[dev\_index] ++;

            m\_device.time\_active\_on\_day[dev\_index].tm\_sec = 0;

            if (m\_device.time\_active\_on\_day[dev\_index].tm\_min >= 60) { /\* hours \*/

                m\_device.time\_active\_on\_day[dev\_index].tm\_min = 0;

                /\* Count time active on day \*/

                m\_device.time\_active\_on\_day[dev\_index].tm\_hour ++;

                /\* Count time active on week \*/

                m\_device.time\_active\_in\_week\_tm\_hour[dev\_index] ++;

                /\* Count time active on month \*/

                m\_device.time\_active\_in\_month\_tm\_hour[dev\_index] ++;

            }

            /\* Count time for tracking user's habit \*/

            if (user\_habit[dev\_index].is\_tracking\_time[dev\_index]) {

                temp\_time\_start\_total\_time\_active[dev\_index] ++; /\* 1 minutes \*/

            }

        }

        /\* Tracking user's habit \*/

        if (0 == user\_habit[dev\_index].is\_tracking\_time[dev\_index]) {

            user\_habit[dev\_index].is\_tracking\_time[dev\_index] = 1; /\* start tracking \*/

            /\* Get time device turn on \*/

            temp\_time\_start[dev\_index].tm\_hour = now.hour();

            temp\_time\_start[dev\_index].tm\_min = now.minute();

        }

        /\* sync database \*/

        uno\_sync\_database\_request = 1;

    }

    else {

        user\_habit[dev\_index].is\_tracking\_time[dev\_index] = 0; /\* stop tracking \*/

        /\* Check time device was used to get user's habit \*/

        if (temp\_time\_start\_total\_time\_active[dev\_index] > user\_habit[dev\_index].total\_time\_active\_previous[m\_device.number\_days\_dev\_active]) {

            /\* Store user's using device time \*/

            user\_habit[dev\_index].total\_time\_active\_previous[m\_device.number\_days\_dev\_active] = temp\_time\_start\_total\_time\_active[dev\_index];

            user\_habit[dev\_index].time\_start\_previous[m\_device.number\_days\_dev\_active].tm\_hour = temp\_time\_start[dev\_index].tm\_hour;

            user\_habit[dev\_index].time\_start\_previous[m\_device.number\_days\_dev\_active].tm\_min = temp\_time\_start[dev\_index].tm\_min;

        }

    }

    /\* CHANNEL3 \*/

    dev\_index = CHANNEL3;

    if (digitalRead(LED3\_PIN)) {

        /\* Count total time active on day \*/

        m\_device.time\_active\_on\_day[dev\_index].tm\_sec += 10; /\* 10s \*/

        if (m\_device.time\_active\_on\_day[dev\_index].tm\_sec >= 60) { /\* minutes \*/

            m\_device.time\_active\_on\_day[dev\_index].tm\_min ++;

            m\_device.time\_active\_in\_week\_tm\_min[dev\_index] ++;

            m\_device.time\_active\_in\_month\_tm\_min[dev\_index] ++;

            m\_device.time\_active\_on\_day[dev\_index].tm\_sec = 0;

            if (m\_device.time\_active\_on\_day[dev\_index].tm\_min >= 60) { /\* hours \*/

                m\_device.time\_active\_on\_day[dev\_index].tm\_min = 0;

                /\* Count time active on day \*/

                m\_device.time\_active\_on\_day[dev\_index].tm\_hour ++;

                /\* Count time active on week \*/

                m\_device.time\_active\_in\_week\_tm\_hour[dev\_index] ++;

                /\* Count time active on month \*/

                m\_device.time\_active\_in\_month\_tm\_hour[dev\_index] ++;

            }

            /\* Count time for tracking user's habit \*/

            if (user\_habit[dev\_index].is\_tracking\_time[dev\_index]) {

                temp\_time\_start\_total\_time\_active[dev\_index] ++; /\* 1 minutes \*/

            }

        }

        /\* Tracking user's habit \*/

        if (0 == user\_habit[dev\_index].is\_tracking\_time[dev\_index]) {

            user\_habit[dev\_index].is\_tracking\_time[dev\_index] = 1; /\* start tracking \*/

            /\* Get time device turn on \*/

            temp\_time\_start[dev\_index].tm\_hour = now.hour();

            temp\_time\_start[dev\_index].tm\_min = now.minute();

        }

        /\* sync database \*/

        uno\_sync\_database\_request = 1;

    }

    else {

        user\_habit[dev\_index].is\_tracking\_time[dev\_index] = 0; /\* stop tracking \*/

        /\* Check time device was used to get user's habit \*/

        if (temp\_time\_start\_total\_time\_active[dev\_index] > user\_habit[dev\_index].total\_time\_active\_previous[m\_device.number\_days\_dev\_active]) {

            /\* Store user's using device time \*/

            user\_habit[dev\_index].total\_time\_active\_previous[m\_device.number\_days\_dev\_active] = temp\_time\_start\_total\_time\_active[dev\_index];

            user\_habit[dev\_index].time\_start\_previous[m\_device.number\_days\_dev\_active].tm\_hour = temp\_time\_start[dev\_index].tm\_hour;

            user\_habit[dev\_index].time\_start\_previous[m\_device.number\_days\_dev\_active].tm\_min = temp\_time\_start[dev\_index].tm\_min;

        }

    }

}

static void caculate\_user\_habit(uint8\_t dev\_index)

{

    /\* If device was set time active depend on user's habit before \*/

    if (m\_device.users\_habit\_is\_set[dev\_index] != 0) {

        return;

    }

    uint8\_t i = 0;

    uint8\_t habit\_day\_was\_select = 0;

    uint16\_t time\_active\_longest = 0;

    time\_active\_longest = user\_habit[dev\_index].total\_time\_active\_previous[habit\_day\_was\_select];

    /\* Caculating longest time active in 5 days \*/

    for (i = 1; i < USER\_HABIT\_TRACE\_DAYS; i++) {

        if (user\_habit[dev\_index].total\_time\_active\_previous[i] > time\_active\_longest)

        {

            time\_active\_longest = user\_habit[dev\_index].total\_time\_active\_previous[i];

            habit\_day\_was\_select = i;

        }

    }

    /\* Set this device had alarm by user's habit \*/

    m\_device.users\_habit\_is\_set[dev\_index] = 1;

    /\* Set time ALARM for User's Habit \*/

    m\_device.users\_habit\_alarm[dev\_index].m\_cmd = uno\_get\_cmd\_for\_pin(dev\_index, 1); /\* turn on \*/

    m\_device.users\_habit\_alarm[dev\_index].m\_time.tm\_hour = user\_habit[dev\_index].time\_start\_previous[habit\_day\_was\_select].tm\_hour;

    m\_device.users\_habit\_alarm[dev\_index].m\_time.tm\_min = user\_habit[dev\_index].time\_start\_previous[habit\_day\_was\_select].tm\_min;

    m\_device.users\_habit\_alarm[dev\_index+1].m\_cmd = uno\_get\_cmd\_for\_pin(dev\_index, 0); /\* turn off \*/

    m\_device.users\_habit\_alarm[dev\_index+1].m\_time.tm\_hour = m\_device.users\_habit\_alarm[dev\_index].m\_time.tm\_hour + (time\_active\_longest / 60);

    m\_device.users\_habit\_alarm[dev\_index+1].m\_time.tm\_min = m\_device.users\_habit\_alarm[dev\_index].m\_time.tm\_min + (time\_active\_longest % 60);

}

void uno\_set\_user\_habit(void)

{

    caculate\_user\_habit(CHANNEL1);

    caculate\_user\_habit(CHANNEL2);

    caculate\_user\_habit(CHANNEL3);

}

void alarm\_handler()

{

    if (m\_device.alarm\_is\_set) {

        time\_alarm();

    }

}

void rtc\_hander(void)

{

    now = rtc.now();

    /\* Check End of day - 0h, update time active every day \*/

    if ((now.hour() == 0) && (now.minute() == 0) && (now.second() == 0))

    {

        /\* Count number of day device was active, just count first five days \*/

        if (m\_device.number\_days\_dev\_active < USER\_HABIT\_TRACE\_DAYS) {

            m\_device.number\_days\_dev\_active ++;

            if (m\_device.number\_days\_dev\_active == USER\_HABIT\_TRACE\_DAYS) {

                uno\_set\_user\_habit();

            }

        }

        if (now.dayOfTheWeek() == 0) { /\* End of the week \*/

            /\* Reset data of week to use for new week \*/

            memset(m\_device.time\_active\_in\_week\_tm\_hour, 0, sizeof(m\_device.time\_active\_in\_week\_tm\_hour));

            memset(m\_device.time\_active\_in\_week\_tm\_min, 0, sizeof(m\_device.time\_active\_in\_week\_tm\_min));

            /\* Reset data to use for new month \*/

            if (now.day() == 1) {

                memset(m\_device.time\_active\_in\_month\_tm\_hour, 0, sizeof(m\_device.time\_active\_in\_week\_tm\_hour));

                memset(m\_device.time\_active\_in\_month\_tm\_min, 0, sizeof(m\_device.time\_active\_in\_week\_tm\_min));

            }

        }

        /\* clear daily data \*/

        memset(&m\_device.time\_active\_on\_day, 0, sizeof(m\_device.time\_active\_on\_day));

    }

    /\* Caculate user's habit \*/

    control\_with\_users\_habit();

}

// uno\_database

#include "uno\_database.h"

#include <EEPROM.h>

#include "led\_button.h"

#include <stdlib.h>

#include "pzem004t.h"

device\_info\_t m\_device;

m\_user\_habit\_trace\_t user\_habit[NUMBER\_CHANNEL];

uint8\_t uno\_sync\_database\_request = 0;

static void eeprom\_clear()

{

    for (uint8\_t i = 0 ; i < (sizeof(device\_info\_t)+1); i++) {

        EEPROM.write(i, 0);

    }

}

void eeprom\_database\_loader(void)

{

    memset(&user\_habit, 0, sizeof(user\_habit));

    memset(&m\_device, 0, sizeof(device\_info\_t));

    EEPROM.get(EEPROM\_DB\_ADDR, m\_device);

    Serial.println("Uno load database:");

    // m\_device.time\_active\_in\_week[0] = 0;

    // m\_device.time\_active\_in\_week[1] = 0;

    // m\_device.time\_active\_in\_week[2] = 0;

    // m\_device.time\_active\_in\_month[0] = 0;

    // m\_device.time\_active\_in\_month[1] = 0;

    // m\_device.time\_active\_in\_month[2] = 0;

}

void eeprom\_sync\_database(void)

{

    Serial.println("Uno sync database");

    eeprom\_clear();

    EEPROM.put(EEPROM\_DB\_ADDR, m\_device);

    uno\_sync\_database\_request = 0;

}

void uno\_update\_current\_state\_switch(void)

{

    if (digitalRead(LED1\_PIN)) {

        m\_device.m\_state\_out[0] = CH1\_ON;

    } else {

        m\_device.m\_state\_out[0] = CH1\_OFF;

    }

    if (digitalRead(LED2\_PIN)) {

        m\_device.m\_state\_out[1] = CH2\_ON;

    } else {

        m\_device.m\_state\_out[1] = CH2\_OFF;

    }

    if (digitalRead(LED3\_PIN)) {

        m\_device.m\_state\_out[2] = CH3\_ON;

    } else {

        m\_device.m\_state\_out[2] = CH3\_OFF;

    }

}

extern float voltage;

extern float current;

extern float power;

extern float energy;

extern float frequency;

extern float pf;

void report\_current\_state(uint8\_t cmd)

{

    char respond[200];

    memset(respond, 0, sizeof(respond));

    char str\_vol[6], str\_current[5], str\_power[5];

    uno\_update\_current\_state\_switch();

    dtostrf(voltage, 4, 2, str\_vol); str\_vol[5] = '\0';

    dtostrf(current, 4, 2, str\_current);

    dtostrf(power, 4, 2, str\_power);

    sprintf(respond, "{\"cmd\_type\":%d, \"state\":[%d, %d, %d], \"Voltage\": %5s, \"Current\": %s, \"Power\": %s, \"res\":\"OK\"}\n",

                            cmd,

                            m\_device.m\_state\_out[0],

                            m\_device.m\_state\_out[1],

                            m\_device.m\_state\_out[2],

                            str\_vol, str\_current, str\_power);

    uno\_respond\_app(respond);

    Serial.println(respond);

}

/\* {cmd\_type:7} \*/

void uno\_get\_time\_alarm\_was\_set(void)

{

    char respond[300];

    char alarms\_str[200];

    char format\_alarm[20];

    uint8\_t index;

    memset(alarms\_str, 0, sizeof(alarms\_str));

    for (index = 0; index < m\_device.alarm\_is\_set; index++) {

        /\* Get Alarm \*/

        memset(format\_alarm, 0, sizeof(format\_alarm));

        sprintf(format\_alarm, "[%d, %d, %d]", \

                    m\_device.m\_time\_alarm[index].m\_cmd, \

                    m\_device.m\_time\_alarm[index].m\_time.tm\_hour, \

                    m\_device.m\_time\_alarm[index].m\_time.tm\_min);

        strcat(alarms\_str, format\_alarm);

        /\* Create Format to make Json \*/

        if (index != m\_device.alarm\_is\_set - 1) {

            strcat(alarms\_str, ", ");

        }

    }

    memset(respond, 0, sizeof(respond));

    sprintf(respond, "{\"cmd\_type\":7, \"alarms\":[%s], \"res\":\"OK\"}\n", alarms\_str);

    uno\_respond\_app(respond);

    Serial.println(respond);

}

/\* {cmd\_type:4, "dev":0} \*/

void uno\_get\_time\_active\_on\_day(void)

{

    char respond[128];

    uint8\_t index;

    for (index = 0; index < NUMBER\_CHANNEL; index++) {

        memset(respond, 0, 128);

        sprintf(respond, "{\"cmd\_type\":4, \"dev\":%d, \"time\_active\_day\":[%d, %d], \"res\":\"OK\"}\n", \

                                index, \

                                m\_device.time\_active\_on\_day[index].tm\_hour, \

                                m\_device.time\_active\_on\_day[index].tm\_min);

        uno\_respond\_app(respond);

        Serial.println(respond);

    }

}

/\* {"cmd\_type":5} \*/

void uno\_get\_time\_active\_in\_week(void)

{

    char respond[128];

    uint8\_t index;

    for (index = 0; index < NUMBER\_CHANNEL; index++) {

        memset(respond, 0, 128);

        sprintf(respond, "{\"cmd\_type\":5, \"dev\":%d, \"time\_active\_week\":[%d, %d], \"res\":\"OK\"}\n", \

                                index, \

                                m\_device.time\_active\_in\_week\_tm\_hour[index], \

                                m\_device.time\_active\_in\_week\_tm\_min[index]);

        uno\_respond\_app(respond);

        Serial.println(respond);

    }

}

/\* {"cmd\_type":6} \*/

void uno\_get\_time\_active\_in\_month(void)

{

    char respond[128];

    uint8\_t index;

    for (index = 0; index < NUMBER\_CHANNEL; index++) {

        memset(respond, 0, 128);

        sprintf(respond, "{\"cmd\_type\":6, \"dev\":%d, \"time\_active\_month\":[%d, %d], \"res\":\"OK\"}\n", \

                                index, \

                                m\_device.time\_active\_in\_month\_tm\_hour[index], \

                                m\_device.time\_active\_in\_month\_tm\_min[index]);

        uno\_respond\_app(respond);

        Serial.println(respond);

    }

}