

ITPLUS ACADEMY

-----🙞🙜🕮🙞🙜-----

**BÁO CÁO MOCK PROJECT**

Lớp: EP8018E – HĐT

Học viên thực hiện: Nguyễn Hữu Thịnh  
 Nguyễn Quốc Việt

Giảng viên hướng dẫn: Trần Khơ

*Hà Nội, tháng 3 năm 2019*

SMART GARDEN

ĐỒ ÁN

**KẾT THÚC HỌC KỲ CUỐI KHÓA**NGÀNH LẬP TRÌNH NHÚNG

**Lời mở đầu**

Hệ thống nhúng là lĩnh vực nghiên cứu, thiết kế và phát triển các hệ thống phần cứng và phần mềm tích hợp. Hệ thống nhúng được nói tới trong nhiều khía cạnh, từ việc thiết kế mạch điện tử đơn giản đến thiết kế vi xử lý, hệ thống nhúng, hệ thống điều khiển PLC, …

Thế giới hiện tại đang chuyển mình với các thiết bị yêu cầu ngày càng thông minh và nhỏ gọn hơn: xe hơi thông minh, điện thoại thông minh, đồng hồ thông minh, áo thông minh, chìa khóa thông minh… Các hệ thống điều khiển dần chuyển sang các hệ thống điều khiển thông minh cả trong hoạt động công nghiệp và đời sống. Chính vì vậy, ngành Hệ thống nhúng và điều khiển tự động đang có nhu cầu rất rất lớn.

Nguồn nhân lực Hệ thống nhúng và Điều khiển tự động với sự chuyên sâu cả về kỹ thuật truyền thông, thiết kế phần cứng và phát triển phần mềm sẽ là một trong những nhân tố quyết định về sự phát triển kinh tế và khả năng ứng dụng công nghệ thông tin và truyền thông vào mọi mặt quản lý và điều hành các tổ chức và quốc gia.

Hiện nay có rất nhiều công ty, tập đoàn lớn trên thế giới và trong nước tập trung phát triển vào lĩnh vực này kéo theo nhu cầu khổng lồ về nhân lực với mức lương hấp dẫn. Do vậy, nhu cầu đào tạo nhân lực về lĩnh vực Hệ thống, với yêu cầu kỹ năng trong cả phát triển phần mềm và thiết kế phần cứng cho các hệ thống truyền thông, trong đó hệ nhúng là một thành phần được đặt ra trên toàn thế giới.

Theo thống kê của Trung tâm Dự báo nhu cầu nhân lực và Thông tin thị trường lao động TP.HCM, hiện nay nhóm ngành Điện;Điện tử - Viễn thông; Công nghệ thông tin là một trong 3 nhóm ngành có nhu cầu nhân lực cao nhất (chiếm 12% tổng nhu cầu nhân lực). Tổng số nhân lực của nhóm ngành này hiện nay là khoảng 556.000 người và dự báo đến năm 2020 sẽ cần lượng nhân lực khoảng 758.000 người. Nhu cầu nhân sự nhóm ngành nghề này trong giai đoạn 2020-2025 vẫn có nhu cầu tuyển dụng rất lớn, có thể lên đến 16.200 người/năm.Cũng theo đánh giá chung, ngành Kỹ thuật Điện tử, Truyền thông còn là ngành có công việc ổn định nhất trong hầu hết các ngành nghề hiện nay.

IT-Plus là một trong những trung tâm bắt kịp xu thế đưa khóa học lập trình nhúng vào giảng dạy và đào tạo. Chính vì lí do đấy em quyết định tham gia khóa học của trung tâm. Trong quá trình học tập chúng em được bổ sung các kiến thức còn thiếu cũng như những kiến thức không được học ở trường. Ngoài ra chúng em còn được tiếp cận các dự án thực tế.

Em chân thành xin cảm ơn tất cả thành viên trong lớp EP8018E – HĐT và đặc biệt là thầy Trần Khơ đã giúp đã tận tình giúp đỡ em hoàn thành project này.

**MỤC LỤC**

Lời mở đầu..........................................................................................................2

1.Xác định vấn để................................................................................................5

1.1 Giới thiệu dự án.........................................................................................5

1.2 Những hệ thống đã được phát triển...........................................................5

1.3 Tổng quan về hệ thống..............................................................................5

1.4 Giới hạn hệ thống......................................................................................5

1.5 Yêu cầu phần cứng....................................... ............................................5

1.5.1 Yêu cầu tối thiểu..............................................................................5

1.5.2 Phần cứng hệ thống hiện tại.............................................................6

1.5.2.1 Board STM32F103C8T6.......................................................6

1.5.2.2 ESP8266 NodeMCU..............................................................7

2. Thiết kế hệ thống.............................................................................................7

2.1 Cấu hình clock hệ thống. .........................................................................7

2.2 Đo độ ẩm đất.............................................................................................9

2.3 Lấy dữ liêu từ cảm biến nhiệt độ và độ ẩm DHT12. ..............................10

2.4 Xác đinh thời gian hệ thống.....................................................................13

2.5 Gửi và nhận dữ liệu từ Server. ................................................................14

2.6 Server.......................................................................................................16

2.7 Lưu đồ thuật toán.....................................................................................18

3 Thực nghiệm. ..................................................................................................20

3.1 Những linh kiện sử dụng trong dự án......................................................20

3.2 Hình ảnh thực tế.......................................................................................22

4. Kết quả đạt được.............................................................................................22

4.1 Kết quả thực tế.........................................................................................22

4.2 Kết luận ...................................................................................................22

5. Phân công công việc......................................................................................22

6. Danh sách kiểm tra........................................................................................23

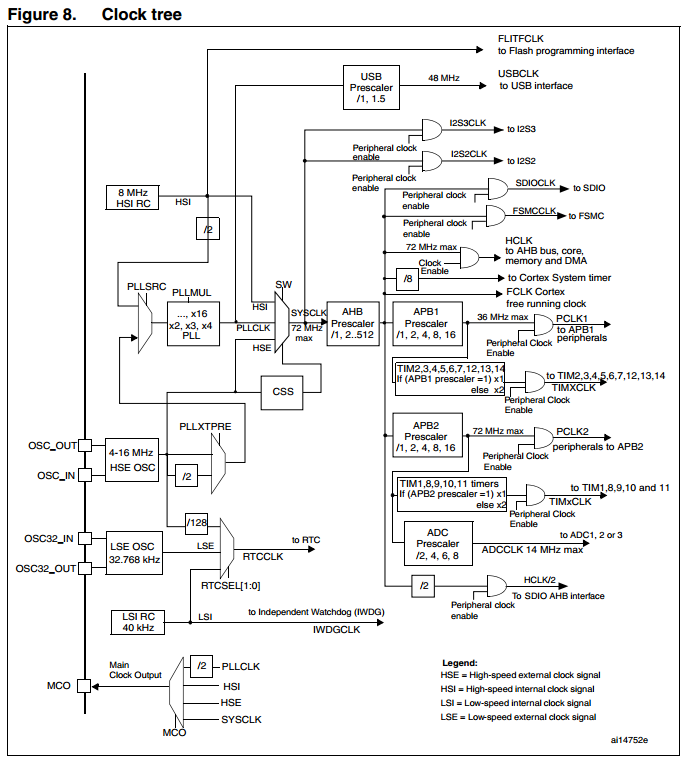
Tài liêu tham khảo.............................................................................................24

Phụ Lục..............................................................................................................25

1. **Xác định vấn đề**
   1. ***Giới thiệu dự án***

* Việt Nam rất nhiều gia đình sống trên thành muốn trồng rau trong nhà để tự cung cấp thực phẩm an toàn hay những người có niềm đam mê với cây cảnh nhưng họ không có thời gian để chăm sóc chúng.
* Nhận thấy điều đó nên em đã bắt tay vào làm một hệ thống để giải quyết những vấn đề đó. Hệ thống đó có tên là : Smart Garden.
* Smart Garden là hệ thống cho biết người dùng nắm bắt được những thông tin cần thiết về khu vườn của họ để đưa ra những giải pháp để giúp cho khu vườn phát triển tốt nhất.
  1. ***Những hệ thống đã được phát triển***
* Dự án của e được lấy ý tưởng từ dự án trồng rau trong nhà kính
  1. ***Tổng quan hệ thống(SMART GARDEN)***
* Sử dụng ADC để lấy dữ liệu độ ẩm đất từ cảm biến độ ẩm đất.
* Sử dụng giao thức I2C để giao tiếp với cảm biến DHT12 để lấy dữ liệu nhiệt độ và độ ẩm không khí.
* Sử dụng UART để giao tiếp với Server.
* Điều khiển các thiết bị thông qua 2 chế độ
* Chế độ điều khiển tự động.
* Chế độ điều khiển bằng tay.
  1. ***Giới hạn của hệ thống***
* Hệ thống chỉ phù hợp với những khu vườn nhỏ, vườn ươm giống hoặc là những chậu cây cảnh nhỏ.
* Những thiết bị có độ chính xác chưa cao.
  1. ***Yêu cầu phần cứng***
* Phụ thuộc nhu cầu người và kích thước khu vườn mà ta lừa chọn những linh kiện phần cứng phù hợp.
  + 1. ***Yêu cầu tối thiểu.***
* 1 vi điểu khiển để có để lấy dữ liệu từ các cảm biến và điều khiển các thiết bị.
* 1 Server để giúp người dùng giao tiếp với hệ thống.
* Những linh kiện khác phụ thuộc vào yêu cầu của người dùng.
  + 1. ***Phần cứng hệ thống hiện tại***
       1. ***Board STM32F103C8T6***
* Vi điều khiển STM32F103C8T6 là họ vi điều khiển 32 bit của hãng TexasInstrument với 64-128 Kb bộ nhớ Flash, USB 2.0 full-speed, CAN, 7 bộ Timer, 2 bộ ADC và 9 giao diện kết nối.
* Lõi : ARM 32 bit Cortex-M3.
* Tần số hoạt động lên tới 72 Mhz.
* Bộ nhớ : 64 Kb Flash, 20 Kb SRAM.
* ADC : 2×12 bit, tần số lấy mẫu 1Mhz.
* DAC : không.
* DMA : Điều khiển 7 kênh DMA
* Timer : 7 bộ, 16 bit
* Giao diện kết nối : 2xI2C, 3xUSART, 2xSPI, CAN, USB 2.0
  + - 1. ***ESP8266 NodeMCU***
* NodeMCU V1.0 được phát triển dựa trên Chip WiFi ESP8266EX bên trong Module ESP-12E dễ dàng kết nối WiFi với một vài thao tác.Board còn tích hợp IC CP2102, giúp dễ dàng giao tiếp với máy tính thông qua Micro USB để thao tác với board.
* Chip: ESP8266EX
* WiFi: 2.4 GHz hỗ trợ chuẩn 802.11 b/g/n
* Điện áp hoạt động: 3.3V
* Điện áp vào: 5V thông qua cổng USB
* Số chân I/O: 11 (tất cả các chân I/O đều có Interrupt/PWM/I2C/One-wire, trừ chân D0)
* Số chân Analog Input: 1 (điện áp vào tối đa 3.3V)
* Bộ nhớ Flash: 4MB
* Giao tiếp: Micro USB.
* Hỗ trợ bảo mật: WPA/WPA2.
* Tích hợp giao thức TCP/IP.

1. **Thiết kế hệ thống**
   1. ***Cấu hình clock hệ thống***

****

*Sơ đồ clock.*

* Hàm UpdateClock72MHz() là hàm chuyển clock hệ thống thành 72MHz (SYSCLK = 72 MHZ).
  + Cho phép khối tạo clock HSI hoạt động và đợi nó sẵn sàng. Set bit HSION lên 1 và đợi bit HSIRDY lên 1.

RCC->CR |= (1 << 0);

while(!(RCC->CR & (1 << 1)));

* + Chọn HSI là clock nguồn của SYSCLK và đợi nó sẵn sàng. Set trường bit SW thành 00 và đợi trường bit SWS thành 00.

RCC->CFGR &= ~(3 << 0);

while((RCC->CFGR & (3 << 2)) != 0);

* + Ngừng hoạt động khối clock PLL để cấu hình nó. Set bit PLLON thành 0.

RCC->CR &= ~(1 << 24);

* + Tăng tần số lên 9 lần của khối PLL clock. Set trường bit PLLMUL thành 7.

RCC->CFGR &= ~(15 << 18);

RCC->CFGR |= (7 << 18);

* + Không chia HSE clock khi PLL chọn HSE làm nguồn nuôi. Set bit PLLXTPRE thành 0.

RCC->CFGR &= ~(1 << 17);

* + Khối PLL clock chọn khối HSE clock làm nguồn nuôi. Set bit PLLSRC thành 1.

RCC->CFGR |= (1 << 16);

* + Cho phép khối clock nguồn HSE hoạt động và đợi nó sẵn sàng. Set bit HSEON lên 1 và đợi bit HSERDY lên 1

RCC->CR |= (1 << 16);

while(!(RCC->CR & (1 << 17)));

* + Cho phép khối tạo clock PLL hoạt động và đợi nó sẵn sàng. Set bit PLLON lên 1 và đợi bit PLLRDY lên 1

RCC->CR |= (1 << 24);

while(!(RCC->CR & (1 << 25)));

* + Chọn khối PLL clock là clock nguồn của SYSCLK và đợi nó sẵn sàng. Set trường bit SW thành 10 và đợi trường bit SWS thành 10.

RCC->CFGR |= (2 << 0);  
while((RCC->CFGR & (3 << 2)) != (2 << 2));

* 1. ***Đo độ ẩm đất****.*

Để đo được độ ẩm đất từ cảm biến độ ẩm đất ta sử dụng bộ ADC của STM32F103C8T6 được cấu hình trong hàm ADC1\_Config().

* + Thay đổi độ ưu tiên của bộ ADC1 trong NVIC là 8 và cho phép ngắt của ADC1 hoạt động.

NVIC->IP[4] |= (8 << 20);  
NVIC->ISER[0] |= (1 << 18);

* + Mở cổng clock của GPIOA và ADC1.

RCC->APB2ENR |= (1 << 2) | (1 << 9);

* + Cấu hình chân PTA1 để chuyển nó hoạt động ở chế độ ADC1.

GPIOA->CRL &= ~(3 << 4);

GPIOA->CRL &= ~(3 << 6);

* + ADC1 sử dụng clock là 9MHz.

RCC->CFGR |= (3 << 14);

* + Cho phép ngắt hoạt động khi chuyển đổi xong.

ADC1->CR1 |= (1 << 5);

* + Chuyển đổi khi bit SWSTART = 1.

ADC1->CR2 |= (1 << 20);

ADC1->CR2 |= (7 << 17);

* + Dữ liêu được đọc từ phải sang trái.

ADC1->CR2 &= ~(1 << 11);

* + ADC1 hoạt động ở chế độ chuyển đổi đơn.

ADC1->CR2 &= ~(1 << 1);

* + Chân nhận tín hiệu là chân PTA1.

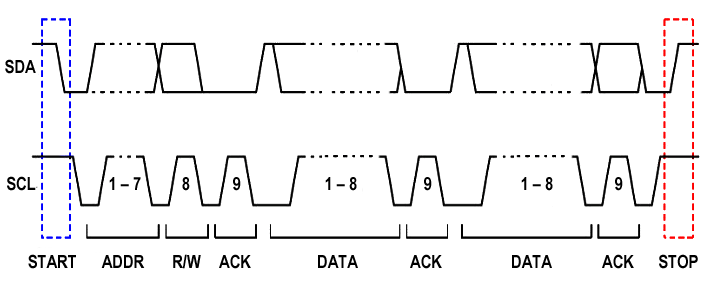
ADC1->SQR3 &= ~(31 << 0);

ADC1->SQR3 |= (1 << 0);

* + Cho phép bộ ADC1 hoạt động.

ADC1->CR2 |= (1 << 0);

* 1. ***Lấy dữ liêu từ cảm biến nhiệt độ và độ ẩm DHT12.***

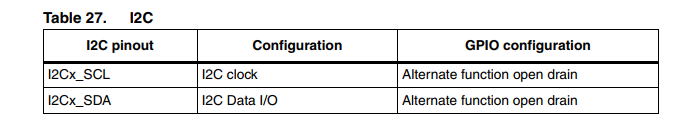
Ta sử dụng I2C module để giao tiếp với DHT12 được cấu hình ở I2C\_Config().

*Giao thức I2C*

* + Mở cổng clock của I2C1 và GPIOB. Set bit I2C1EN và GPIOBEN thành 1.

RCC->APB2ENR |= (1 << 3);

RCC->APB1ENR |= (1 << 21);

* + Cấu hình các chân PTB6 và PTB7 để chuyển từ chế độ I/O sang I2C. Set trường bit CNF và MODE của PTB6 và PTB7 thành 0b11

GPIOB->CRL |= (3 << 24);

GPIOB->CRL |= (3 << 26);

GPIOB->CRL |= (3 << 28);

GPIOB->CRL |= (3 << 30);

* + Khai báo clock cấp cho I2C là 18MHz. Set trường bit FREQ thành 0x12.

I2C1->CR2 &= ~(63 << 0);

I2C1->CR2 |= (18 << 0);

* + Module I2C hoạt động ở chế độ I2C tiêu chuẩn. Set bit F/S thành 0.

I2C1->CCR &= ~(1 << 15);

* + Tốc độ truyển và nhận dữ liệu là 100KHz. Set trường bit CCR thành 0x5A.

I2C1->CCR &= ~(4095 << 0);

I2C1->CCR |= (90 << 0);

* + Cho phép I2C module hoạt động. Set bit PE thành 1.

I2C1->CR1 |= (1 << 0);

Hàm Read\_DHT12 là hàm truyền cmd và nhận dữ liệu từ cảm biến nhiệt độ, độ ẩm không khí DHT12.

* Tạo ra tín hiệu bắt đầu(Start). Set bit START thành 1.

I2C1->CR1 |= (1 << 8);

* Đợi I2C1 module tạo ra tín hiệu bắt đầu(Start). Đợi bit SB thành 1.

while(!(I2C1->SR1 & (1 << 0)));

* Xóa bit SB.

I2C1->SR1;

* Gửi địa chỉ của DHT12 là 0X5C và bit thông báo gửi dữ liệu. Set trường bit DR thành 0xB8.

I2C1->DR = 0xB8;

* Đợi gửi địa chỉ xong. Đợi bit ADDR thành 1.

while(!(I2C1->SR1 & (1 << 1)));

* Xóa bit trạng thái ADDR.

I2C1->SR1;

I2C1->SR2;

* Đợi bit trạng thái TxE = 1 để có thể gửi dữ liệu. Đợi bit TxE thành 1.

while(!(I2C1->SR1 & (1 << 7)));

* Gửi lệnh. Set trường bit DR thành cmd.

I2C1->DR = cmd;

* Đợi bit trạng thái TxE và BTF để xác nhận là đã gửi xong lệnh và có thể gửi tiếp. Đợi bit TxE và BTF thành 1.

while((!(I2C1->SR1 & (1 << 7))) && (!(I2C1->SR1 & (1 << 2))));

* Tạo ra tín hiệu bắt đầu(Start). Set bit START thành 1.

I2C1->CR1 |= (1 << 8);

* Đợi I2C1 module tạo ra tín hiệu bắt đầu(Start). Đợi bit SB thành 1.

while(!(I2C1->SR1 & (1 << 0)));

* Xóa bit SB.

I2C1->SR1;

* Gửi địa chỉ của DHT12 là 0X5C và bit thông báo nhận dữ liệu. Set trường bit DR thành 0xB9.

I2C1->DR = 0xB9;

* Đợi gửi địa chỉ xong. Đợi bit ADDR thành 1.

while(!(I2C1->SR1 & (1 << 1)));

* Xóa bit trạng thái ADDR.

I2C1->SR1;

I2C1->SR2;

* Đợi bit trạng thái RxE = 1 để có thể nhận dữ liệu. Đợi bit RxE thành 1.

while(!(I2C1->SR1 & (1 << 6)));

* Lưu lại dữ liêu.

Data = I2C1->DR;

* Tạo ra tín hiệu kết thúc(End). Set bit END thành 1.

I2C1->CR1 |= (1 << 9);

* Trả về dữ liệu vừa nhận được.

return Data;

* 1. ***Xác đinh thời gian hệ thống.***

Sử dụng RTC module để xác định thời gian hoạt động của các thiết bị được cấu hình ở hàm RTC\_Config()

* + Mở cổng clock PWR và BKP để có thể thay đổi các giá trị tring thanh ghi của vùng sao lưu. Set bit PWREN và BKPEN thành 1.

RCC->APB1ENR |= (1 << 27) | (1 << 28);

* + Cho phép truy cập vào các thanh ghi vùng sao lưu. Set bit DPB thành 1.

PWR->CR |= (1 << 8);

* + Khởi tạo lại các thanh ghi của vùng nhớ sao lưu. Set bit BDRST thành 1 rồi xóa.

RCC->BDCR |= (1 << 16);

RCC->BDCR &= ~(1 << 16);

* + Chọn clock nuôi RTC module là khối clock HSE / 128. Set trường bit RTCSEL thành 0b11.

RCC->BDCR |= (3 << 8);

* + Cho phép module RTC clock hoạt động. Set bit RTCEN thành 1.

RCC->BDCR |= (1 << 15);

* + Đợi cho các thanh ghi của RTC đồng bộ. Đợi bit RSF thành 1.

while(!(RTC->CRL & (1 << 3)));

* + Đợi cho các thao tác lên thanh ghi RTC kết thúc. Đợi bit RTOFF thành 1.

while(!(RTC->CRL & (1 << 5)))

* + Chuyển sang chế độ có thể cấu hình RTC. Set bit CNF thành 1.

RTC->CRL |= (1 << 4);

* + Chia clock xuống còn 1Hz. Set trường bit thành 0xF424.

RTC->PRLL = 62500;

* + Thoát khỏi chế độ cấu hình. Set bit CNF thành 0.

RTC->CRL &= ~(1 << 4);

* + Đợi cho các thao tác lên thanh ghi RTC kết thúc. Đợi bit RTOFF thành 0.

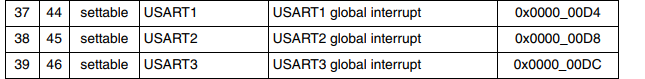
while(!(RTC->CRL & (1 << 5)));

* 1. ***Gửi và nhận dữ liệu từ Server.***

Sử dụng UART module để giao tiếp với Server được cấu hình ở hàm UART1\_Config()



*Giao thức UART.*

* + Thay đổi độ ưu tiên của ngắt UART1 là 1 và cho phép ngắt UART1 hoạt động.

*Bảng vecto ngắt UART*

NVIC->IP[9] |= (1 << 12);

NVIC->ISER[1] |= (1 << 5);

* + Mở cổng clock của GPIOA và UART1. Set bit GPIOAEN thành UART1 thành 1.

RCC->APB2ENR |= (1 << 2) | (1 << 14);

* + Chuyển đổi chân PTA9 và PTA10 từ I/O mode sang UART1 mode. Set bit trường bit MODE của PTA9 thành 0b11 và trường bit CNF PTA9 thành 0b10, set bit trường bit MODE của PTA10 thành 0b00 và trường bit CNF PTA9 thành 0b01.

/\*\*\*\* Config PTA9 - USART-TX. \*\*\*\*/

GPIOA->CRH |= (3 << 4);

GPIOA->CRH &= ~(3 << 6);

GPIOA->CRH |= (2 << 6);

/\*\*\*\*Config PTA10 - USART - RX.\*\*\*\*/

GPIOA->CRH &= ~(3 << 8);

GPIOA->CRH &= ~(3 << 10);

GPIOA->CRH |= (1 << 10);

* + Baud rate = 9600. Set trường bit DIV\_Mantissa thành 0x1D4 và trường bit DIV\_Fraction thành 0xC;

//Baud rate : 9600 -> BRR = 72MHZ / (9600 \* 16) = 468.75.

USART1->BRR = (468 << 4) | (12 << 0);

* + 1 bit start, 1 bit stop, 8 bit dữ liệu. Set bit M của thanh ghi CR1 thành 0 và trường bit STOP của thanh ghi CR2 thành 0b00.

USART1->CR1 &= ~(1 << 12);

USART1->CR2 &= ~(3 << 12);

* + Không cho phép bit chẵn lẻ hoạt động. Set bit PCE thành 1.

USART1->CR1 &= ~(1 << 10);

* + Cho phép UART1 hoạt động. Set bit UE thành 1.

USART1->CR1 |= (1 << 13);

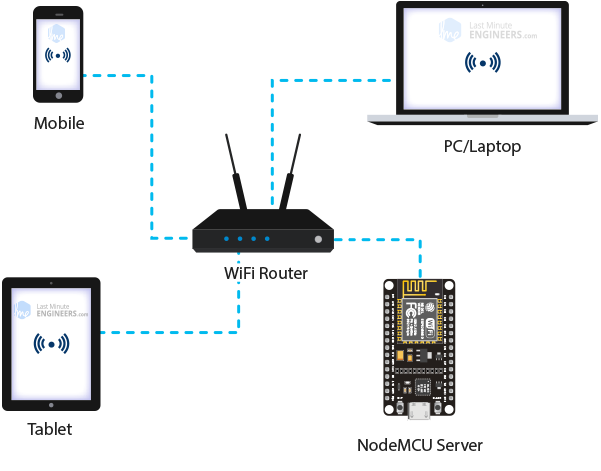
* + Ngắt khi có tín hiệu nhận. Set bit RXNEIE thành 1.

USART1->CR1 |= (1 << 5);

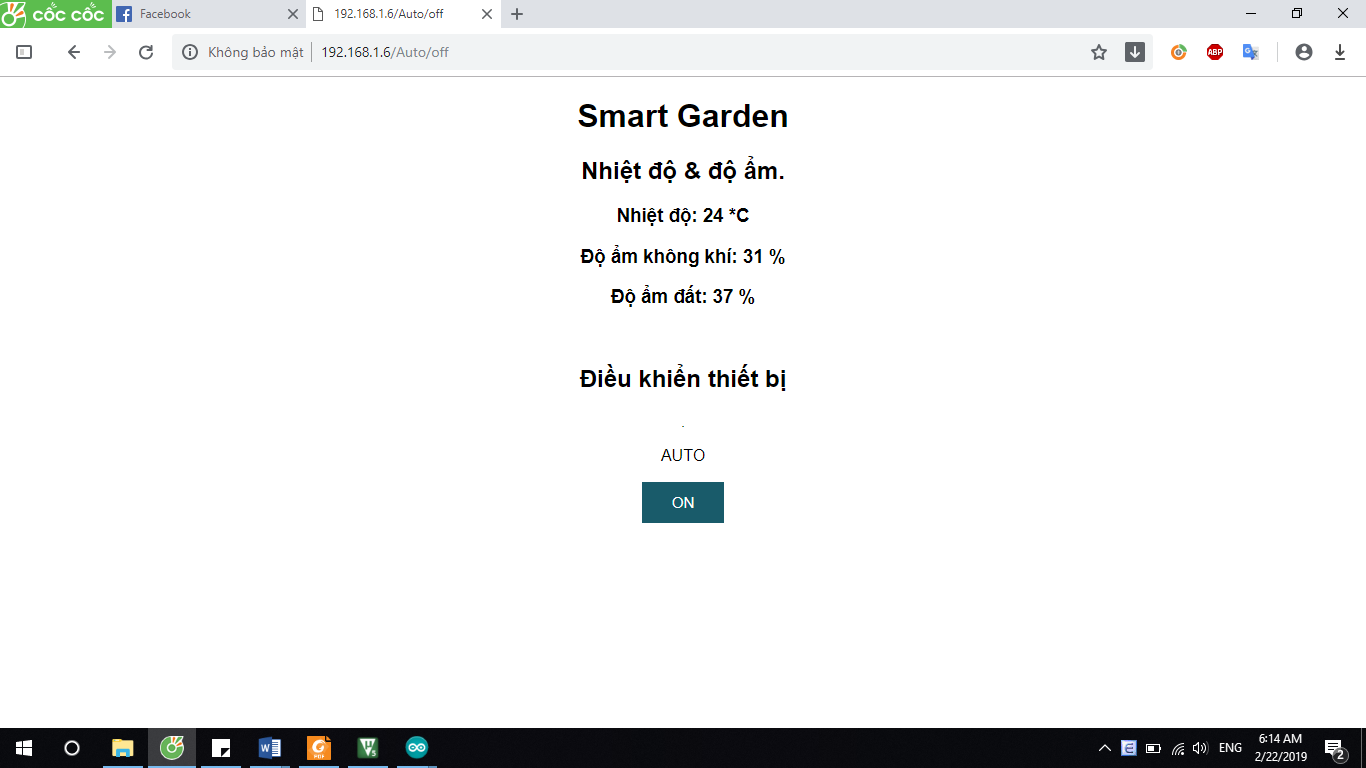
* + Cho phép truyền và nhận dữ liệu. Set bit RE và TE thành 1.

USART1->CR1 |= (1 << 3);  
USART1->CR1 |= (1 << 2);

* 1. **Server**

******

*Mô hình người dùng giao tiếp với Server*

* + Server sẽ truy cập vào mạng Wifi và từ đó những những thiết bị công nghệ cao kết nối với cùng Wifi của server có thể truy vào server để đọc dữ liệu cũng như là gửi dữ liệu điều khiển về cho server.

*Giao diện người dùng ở chế độ tự động*

* + Ở chế độ tự động thì người dùng sẽ nhìn thấy nhiệt độ, độ ẩm không khí và độ ẩm đất cùng với nút điều khiển chuyển chế độ.
  + Ở chế độ điều khiển bằng tay thì người dùng sẽ nhìn thấy nhiệt độ, độ ẩm không khí và độ ẩm đất, nút điều khiển chuyển chế độ cùng với nút bật tắt đèn và nút bật tắt máy bơm.
  1. ***Lưu đồ thuật toán***

*Lưu đồ thuật toán*

Bắt đầu

Mode

Đủ

Không đủ

Bật

Tắt

Bật

Tắt

> 60%

< 30%0%

Tự động

Bằng tay

Ánh sáng

Kết thúc

Đèn

Máy bơm

Kiểm tra mode

Bật đèn

Tắt đèn

Điều khiển

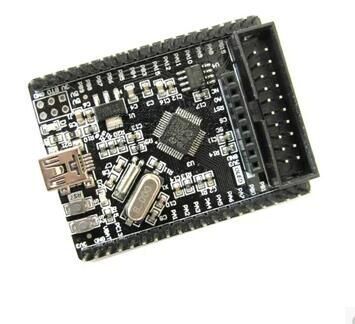
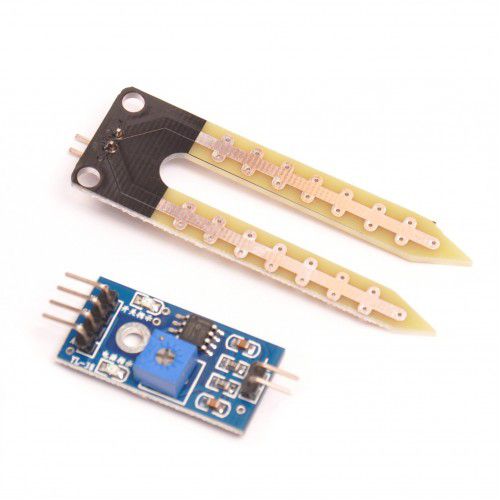
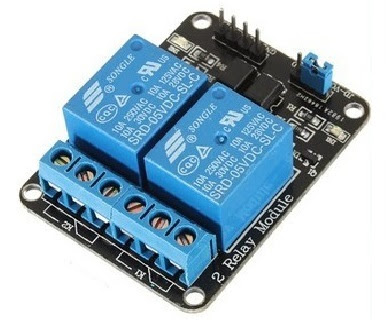
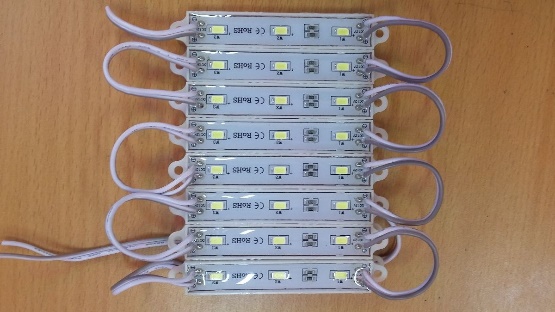
Độ ẩm đất

Tắt máy bơm

Bật máy bơm

1. **Thực nghiệm**
   1. ***Những linh kiện sử dụng trong dự án.***

******

******

*Máy bơm ngầm mini*

*Đèn hắt sáng 12V*

*Mạch nạp ST - Link V2*

*STM32F103C8T6 MINI V2*

*ESP8266 NODE-MCU*

*Cảm biến ánh sáng*

*Relay 2 kênh*

*Cảm biến độ ẩm đất*

*Cảm biến nhiệt độ, độ ẩm DHT12*

*ADAPTER 12V – 2A*

* 1. ***Hình ảnh thực tế***

*Hình ảnh thực tế*

**

1. **Kết quả đạt được**
   1. ***Kết quả thực tế***

* Đã đo được nhiệt độ, độ ẩm không khí, độ ẩm đất.
* Đã gửi được dữ liệu lên server(ESP8266).
* Hoạt động ở chế độ tự động ổn định ở chế độ điều khiển bằng tay rất chính xác.
  1. **Kết luận**
* Dự án đã hoàn thành 100% mục tiêu đề ra. Tuy nhiên còn nhiều chỗ cần khắc phục như cần phải nhanh và nhạy hơn, cảm biến nhiệt độ ẩm cần được đọc chuẩn và chính xác hơn. Code cần phải viết dưới dạng thư viện để người khác đọc dễ hiểu hơn.

1. **Phân công công việc**

|  |  |  |
| --- | --- | --- |
|  | Nguyễn Hữu Thinh | Nguyễn Quốc Việt |
| Lập trình Server |  | X |
| Lập trình MCU kết với ngoại vi và điều khiển các thết bị | X |  |
| Word | X |  |

1. **Danh sách kiểm tra**

|  |  |  |
| --- | --- | --- |
|  | Đạt | Chưa đạt |
| Nhiệt độ và độ ẩm không khí. | X |  |
| Độ ẩm đất | X |  |
| Điều khiển đèn, máy bơm | X |  |
| Chế độ tự động | X |  |
| Chế độ điều khiển bằng tay | X |  |

**Tài liêu tham khảo**

1. STM32F10x Refenence Manual.
2. STM32\_Smart\_Schematic(Sơ đồ chân của STM32F103C8T6 V2).
3. Tài liệu Arduino ESP8266.

**Phụ Lục**

1. ***Code STM32F103C8T6.***

#include "stm32f10x.h"

#include "Value.h"

void UpdateClock72MHz()

{

RCC->CR |= (1 << 0);

while(!(RCC->CR & (1 << 1)));

RCC->CFGR &= ~(3 << 0);

while((RCC->CFGR & (3 << 2)) != 0);

RCC->CR &= ~(1 << 24);

RCC->CFGR &= ~(15 << 18);

RCC->CFGR |= (7 << 18);

RCC->CFGR &= ~(1 << 17);

RCC->CFGR |= (1 << 16);

RCC->CFGR &= ~(15 << 4);

RCC->CFGR &= ~(7 << 8);

RCC->CFGR |= (5 << 8);

RCC->CR |= (1 << 16);

while(!(RCC->CR & (1 << 17)));

RCC->CR |= (1 << 24);

while(!(RCC->CR & (1 << 25)));

RCC->CFGR |= (2 << 0);

while((RCC->CFGR & (3 << 2)) != (2 << 2));

}

void ADC1\_Config()

{

NVIC->IP[4] |= (8 << 20);

NVIC->ISER[0] |= (1 << 18);

RCC->APB2ENR |= (1 << 2) | (1 << 9);

GPIOA->CRL &= ~(3 << 4);

GPIOA->CRL &= ~(3 << 6);

RCC->CFGR |= (3 << 14);

ADC1->CR1 |= (1 << 5);

ADC1->CR2 |= (1 << 20);

ADC1->CR2 |= (7 << 17);

ADC1->CR2 &= ~(1 << 11);

ADC1->CR2 &= ~(1 << 1);

ADC1->SQR1 &= ~(15 << 20);

ADC1->SQR3 &= ~(31 << 0);

ADC1->SQR3 |= (1 << 0);

ADC1->CR2 |= (1 << 0);

}

void I2C\_Config()

{

RCC->APB2ENR |= (1 << 3);

RCC->APB1ENR |= (1 << 21);

GPIOB->CRL |= (3 << 24);

GPIOB->CRL |= (3 << 26);

GPIOB->CRL |= (3 << 28);

GPIOB->CRL |= (3 << 30);

I2C1->CR2 &= ~(63 << 0);

I2C1->CR2 |= (18 << 0);

I2C1->CCR &= ~(1 << 15);

I2C1->CCR &= ~(4095 << 0);

I2C1->CCR |= (90 << 0);

I2C1->TRISE &= ~(63 << 0);

I2C1->TRISE |= (91 << 0);

I2C1->CR1 |= (1 << 0);

}

char Read\_DHT12(char cmd)

{

char Data;

I2C1->CR1 |= (1 << 8);

while(!(I2C1->SR1 & (1 << 0)));

I2C1->SR1;

I2C1->DR = 0xB8;

while(!(I2C1->SR1 & (1 << 1)));

I2C1->SR1;

I2C1->SR2;

while(!(I2C1->SR1 & (1 << 7)));

I2C1->DR = cmd;

while((!(I2C1->SR1 & (1 << 7))) && (!(I2C1->SR1 & (1 << 2))));

I2C1->CR1 |= (1 << 8);

while(!(I2C1->SR1 & (1 << 0)));

I2C1->SR1;

I2C1->DR = 0xB9;

while(!(I2C1->SR1 & (1 << 1)));

I2C1->SR1;

I2C1->SR2;

while(!(I2C1->SR1 & (1 << 6)));

Data = I2C1->DR;

I2C1->CR1 |= (1 << 9);

return Data;

}

void UART1\_Config()

{

NVIC->IP[9] |= (1 << 12);

NVIC->ISER[1] |= (1 << 5);

RCC->APB2ENR |= (1 << 2) | (1 << 14);

GPIOA->CRH |= (3 << 4);

GPIOA->CRH &= ~(3 << 6);

GPIOA->CRH |= (2 << 6);

GPIOA->CRH &= ~(3 << 8);

GPIOA->CRH &= ~(3 << 10);

GPIOA->CRH |= (1 << 10);

USART1->BRR = (468 << 4) | (12 << 0);

USART1->CR1 &= ~(1 << 12);

USART1->CR2 &= ~(3 << 12);

USART1->CR1 &= ~(1 << 10);

USART1->CR1 |= (1 << 13);

USART1->CR1 |= (1 << 5);

USART1->CR1 |= (1 << 3);

USART1->CR1 |= (1 << 2);

}

void Send\_for\_ESP8266()

{

USART1->DR = '#';

while((!(USART1->SR & (1 << 6))) && (!(USART1->SR & (1 << 7))));

USART1->SR &= ~(1 << 6);

USART1->SR;

USART1->DR = Value.DoAmDat;

while((!(USART1->SR & (1 << 6))) && (!(USART1->SR & (1 << 7))));

USART1->SR &= ~(1 << 6);

USART1->SR;

USART1->DR = Value.DoAmKK;

while((!(USART1->SR & (1 << 6))) && (!(USART1->SR & (1 << 7))));

USART1->SR &= ~(1 << 6);

USART1->SR;

USART1->DR = Value.NhietDoKK;

while((!(USART1->SR & (1 << 6))) && (!(USART1->SR & (1 << 7))));

USART1->SR &= ~(1 << 6);

USART1->SR;

USART1->DR = '@';

while((!(USART1->SR & (1 << 6))) && (!(USART1->SR & (1 << 7))));

USART1->SR &= ~(1 << 6);

USART1->SR;

}

int Check\_String(char \*KT)

{

uint8\_t i = 0;

while(i < RX\_leng)

{

if(Data\_Receiver[i] != \*(KT + i))

{

return FALSE;

}

i++;

}

return TRUE;

}

void RTC\_Config()

{

RCC->APB1ENR |= (1 << 27) | (1 << 28);

PWR->CR |= (1 << 8);

RCC->BDCR |= (1 << 16);

RCC->BDCR &= ~(1 << 16);

RCC->BDCR |= (3 << 8);

RCC->BDCR |= (1 << 15);

while(!(RTC->CRL & (1 << 3)));

RTC->CRL &= ~(1 << 3);

while(!(RTC->CRL & (1 << 5)));

RTC->CRL |= (1 << 4);

RTC->PRLL = 62500;

RTC->CNTL = 0;

RTC->CRL &= ~(1 << 4);

while(!(RTC->CRL & (1 << 5)));

}

void Delay\_ms(uint32\_t time)

{

Count = time;

while(Count != 0);

}

void SysTick\_Configuration()

{

SysTick->CTRL &= ~(1 << 0);

SysTick->CTRL |= (1 << 1);

SysTick->CTRL &= ~(1 << 2);

SysTick->VAL =0;

SysTick->LOAD = 9000;

SysTick->CTRL |= (1 << 0);

}

void Check\_Light()

{

uint32\_t Temp = 0;

Temp |= (RTC->CNTH << 16);

Temp |= RTC->CNTL;

Temp = Temp % 86400;

if(((Temp >= 21600 ) && (Temp < 64800)) && (AUTO == TRUE))

{

if(GPIOA->IDR & (1 << 2))

{

GPIOA->ODR |= (1 << 6);

}

else

{

GPIOA->ODR &= ~(1 << 6);

}

}

}

void Config\_Check\_Light()

{

RCC->APB2ENR |= (1 << 2);

GPIOA->CRL &= ~(3 << 8);

GPIOA->CRL &= ~(3 << 10);

GPIOA->CRL |= (2 << 10);

GPIOA->ODR |= (1 << 2);

}

void GPIO\_Config()

{

RCC->APB2ENR |= (1 << 2);

GPIOA->CRL |= (3 << 24);

GPIOA->CRL &= ~(3 << 26);

GPIOA->CRL |= (3 << 28);

GPIOA->CRL &= ~(3 << 30);

}

int main()

{

UpdateClock72MHz();

RTC\_Config();

UART1\_Config();

ADC1\_Config();

I2C\_Config();

GPIO\_Config();

Config\_Check\_Light();

SysTick\_Configuration();

while(1)

{

Delay\_ms(4000);

Value.DoAmKK = Read\_DHT12(0x00);

Delay\_ms(4000);

Value.NhietDoKK = Read\_DHT12(0x02);

}

}

void SysTick\_Handler()

{

Count--;

if(RT % 1000 == 0)

{

Check\_Light();

//Conversion ADC1.

ADC1->CR2 |= (1 << 22);

Count\_Temp\_DoAmDat++;

}

if(RT >= 10000)

{

Value.DoAmDat = 0;

for(i = 0; i < 10; i++)

{

Value.DoAmDat += Temp\_DoAmDat[i];

}

Value.DoAmDat /= Value.DoAmDat / 10;

Send\_for\_ESP8266();

RT = 0;

}

else

{

RT++;

}

}

void USART1\_IRQHandler()

{

if(USART1->DR == '#')

{

if(Check\_String("AUTO") == TRUE)

{

AUTO = TRUE;

GPIOA->ODR &= ~(1 << 7);

}

else if(Check\_String("NOT\_AUTO") == TRUE) {

AUTO = FALSE;

GPIOA->ODR &= ~(1 << 6);

GPIOA->ODR &= ~(1 << 7);

}

else if((Check\_String("ON\_LIGHT") == TRUE) && (AUTO == FALSE))

{

GPIOA->ODR |= (1 << 6);

}

else if((Check\_String("ON\_PUMP") == TRUE) && (AUTO == FALSE))

{

GPIOA->ODR |= (1 << 7);

}

else if((Check\_String("OFF\_LIGHT") == TRUE) && (AUTO == FALSE))

{

GPIOA->ODR &= ~(1 << 6);

}

else if((Check\_String("OFF\_PUMP") == TRUE) && (AUTO == FALSE))

{

GPIOA->ODR &= ~(1 << 7);

}

USART1->DR;

RX\_leng = 0;

}

else if(USART1->DR == '\*')

{

RX\_leng = 0;

USART1->DR;

}

else

{

Data\_Receiver[RX\_leng] = USART1->DR;

RX\_leng++;

}

USART1->SR &= ~(1 << 5);

}

void ADC1\_2\_IRQHandler()

{

if((ADC1->DR > 2867) && (AUTO == TRUE))

{

GPIOA->ODR |= (1 << 7);

}

else if((ADC1->DR <= 1638) && (AUTO == TRUE))

{

GPIOA->ODR &= ~(1 << 7);

}

Temp\_DoAmDat[Count\_Temp\_DoAmDat] = (char)(100 - (((float)(ADC1->DR) / 4095) \* 100));

ADC1->SR &= ~(1 << 1);

}

1. ***Code ESP8266.***

#include <ESP8266WiFi.h>

const char\* ssid = "WiFI";

const char\* password = "PassWord";

WiFiServer server(80);

String header;

String LIGHT = "off";

String PUMP = "off";

String AUTO = "off";

char Temp;

int i = 0;

char Receive[4];

int NhietDoKK = 0;

int DoAmKK;

int DoAmDat;

void setup() {

Serial.begin(9600);

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

if (Serial.available() > 0) {

Temp = Serial.read();

if(Temp == '#')

{

i = 0;

}

else if(Temp == '@')

{

DoAmDat = Receive[0];

DoAmKK = Receive[1];

NhietDoKK = Receive[2];

}

else if(i <= 3)

{

Receive[i] = Temp;

i++;

}

}

WiFiClient client = server.available();

if (client) {

String currentLine = "";

while (client.connected()) {

if (client.available()) {

char c = client.read();

//Serial.write(c);

header += c;

if (c == '\n') {

if (currentLine.length() == 0) {

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

// turns the GPIOs on and off

if (header.indexOf("GET /Light/on") >= 0) {

Serial.print("\*ON\_LIGHT#");

LIGHT = "on";

} else if (header.indexOf("GET /Light/off") >= 0) {

Serial.print("\*OFF\_LIGHT#");

LIGHT = "off";

} else if (header.indexOf("GET /Pump/on") >= 0) {

Serial.print("\*ON\_PUMP#");

PUMP = "on";

} else if (header.indexOf("GET /Pump/off") >= 0) {

Serial.print("\*OFF\_PUMP#");

PUMP = "off";

} else if (header.indexOf("GET /Auto/on") >= 0) {

Serial.print("\*NOT\_AUTO#");

AUTO = "on";

} else if (header.indexOf("GET /Auto/off") >= 0) {

Serial.print("\*AUTO#");

LIGHT = "off";

PUMP = "off";

AUTO = "off";

}

client.println("<!DOCTYPE html><html>");

client.println("<head><meta charset='utf-8'><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

your preferences

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #195B6A; border: none; color: white; padding: 12px 30px;");

client.println("text-decoration: none; font-size: 15px; margin: 1px; cursor: pointer;}");

client.println(".button2 {background-color: #77878A;}</style></head>");

client.println("<body><h1>Smart Garden</h1>");

client.println("<h2>Nhiệt độ & độ ẩm.</h2> ");

client.println("<h3>Nhiệt độ: ");

client.println(NhietDoKK);

client.println("\*C</h3><h3>Độ ẩm không khí: ");

client.println(DoAmKK);

client.println("%</h3><h3>Độ ẩm đất: ");

client.println(DoAmDat);

client.println("%</h3>");

client.println("<br>");

client.println("<h2>Điều khiển thiết bị</h2>. ");

client.println("<p>AUTO</p>");

if (AUTO=="off") {

client.println("<p><a href=\"/Auto/on\"><button class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/Auto/off\"><button class=\"button button2\">OFF</button></a></p>");

}

if(AUTO == "on")

{

client.println("<p>LIGHT</p>");

if (LIGHT=="off") {

client.println("<p><a href=\"/Light/on\"><button class=\"button button2\">OFF</button></a></p>");

} else {

client.println("<p><a href=\"/Light/off\"><button class=\"button \">ON</button></a></p>");

}

client.println("<p>PUMP</p>");

if (PUMP=="off") {

client.println("<p><a href=\"/Pump/on\"><button class=\"button button2\">OFF</button></a></p>");

} else {

client.println("<p><a href=\"/Pump/off\"><button class=\"button\">ON</button></a></p>");

}

}

client.println("</body></html>");

client.println();

break;

} else {

currentLine = "";

}

} else if (c != '\r') {

currentLine += c;

}

}

}

header = "";

client.stop();

}

}