**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**-----🙣🕮🙡-----**



**BÁO CÁO**

**ĐỒ ÁN THIẾT KẾ HỆ THỐNG NHÚNG**

**Đề tài: Đo nhiệt độ, độ ẩm sử dụng STM32**

**hiển thị lên máy tính thông qua giao tiếp UART**

Giảng viên hướng dẫn : TS.Nguyễn Ngọc Minh

Nhóm : 02

Sinh viên thực hiện : Nguyễn Thị Hương - B15DCDT095

Ngô Thị Là - B15DCDT112

Hoàng Thị Huyền - B15DCDT102

Lê Ô Na - B15DCDT133

**Hà Nội - 2019**

# Lời nói đầu

Trong cuộc sống hiện nay có rất nhiều yếu tố ảnh hưởng đến quá trình sản xuất và con người. Trong đó nhiệt độ - độ ẩm là các yếu tố được đề cập tới rất nhiều, vì thế mạch đo nhiệt độ - độ ẩm ra đời là sự tất yếu. Với sự phát triển của công nghệ hiện nay việc sản xuất mạch đo nhiệt độ với độ chính xác cao là điều khá đơn giản. Việc áp dụng trong thực tế ngày càng phổ biến hơn khi nhiệt độ - độ ẩm đóng vai trò lớn ảnh hưởng đến con người, hàng hóa, máy móc, độ bền,…

Trong các nhà máy, việc đo được nhiệt độ - độ ẩm chính xác giúp sản xuất, theo dõi kho chứa hàng hóa, bảo quản máy móc vận hành. Trong y tế, nó được sử dụng để theo dõi các phòng cách ly, phòng điều trị cho bệnh nhân,…Tùy theo nhu cầu mà chúng ta có thể tùy biến thêm các chức năng để phù hợp với điều kiện ngoại cảnh. Vì vậy nhóm chúng em lựa chọn đề tài: “*Đo nhiệt độ, độ ẩm sử dụng STM32 hiển thị lên máy tính qua giao tiếp UART”* để ứng dụng vào các công việc khác nhau như nông nghiệp thông minh, hệ thống tưới tiêu tự động,…

Mặc dù chúng em đã cố gắng để hoàn thành đề tài nhưng không thể tránh khỏi sai sót. Chúng em mong thầy có những nhận nhận xét góp ý để bài báo cáo được hoàn thiện hơn.

Mục lục

[Lời nói đầu 2](#_Toc8678892)

[1. Khái quát về đề tài 4](#_Toc8678893)

[2. Những công việc đã đạt được trong đề tài 4](#_Toc8678894)

[II. Các linh kiện sử dụng trong mạch. 4](#_Toc8678895)

[1. Kit STM32F0 Discovery 4](#_Toc8678896)

[2. Sensor DHT11 8](#_Toc8678897)

[3. Module giao tiếp UART 9](#_Toc8678898)

[4. LCD 16x2 10](#_Toc8678899)

[III. Nguyên lý hoạt động của mạch 11](#_Toc8678900)

[1. Nguyên lý chung 11](#_Toc8678901)

[2. Cách đọc sensor DHT11 12](#_Toc8678902)

[3. Truyền dữ liệu qua UART 18](#_Toc8678903)

[4. Cách hiển thị dữ liệu lên máy tính và vẽ biểu đồ 22](#_Toc8678904)

[IV. Kết luận và hướng phát triển đề tài 26](#_Toc8678905)

[1. Kết luận 26](#_Toc8678906)

[2. Hướng phát triển của đề tài 26](#_Toc8678907)

[Tài liệu tham khảo 27](#_Toc8678908)

[Phụ Lục 28](#_Toc8678909)

1. **Giới thiệu chung về đề tài.**
2. **Khái quát về đề tài**

Nhiệt độ - độ ẩm là 2 thông số rất quan trọng trong không chỉ ngành công nghiệp, nông nghiệp mà còn trong nhiều lĩnh vực khác. Dựa vào nhiệt độ - độ ẩm, chúng ta có thể theo dõi nơi sản xuất có độ ẩm không khí ra sao, đưa ra các quyết định chăm sóc cây trồng, bảo quản thiết bị,…Với đề tài này chúng em thực hiện đo nhiệt độ - độ ẩm sử dụng kit STM32 discovery, hiển thị giá trị đo được trên giao diện máy tính, quy mô thực hiện nhỏ trong phạm vi một phòng học.

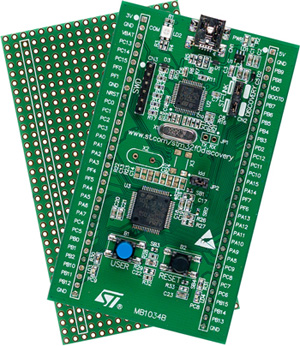
1. **Những công việc đã đạt được trong đề tài**

Với đề tài này chúng em đã thành công trong việc đo đạc nhiệt độ và độ ẩm, giao tiếp giữa stm32 và máy tính ổn định, thành công trong việc xây dựng giao diện từ nền tảng của Python (Tkinter) đồng thời lưu dữ liệu vào database và thực hiện vẽ biểu đồ 2D.

1. **Các linh kiện sử dụng trong mạch.**

## Kit STM32F0 Discovery

STM32F0 Discovery giúp khám phá các tính năng của STM32F051xx Cortex®-M0, dễ dàng phát triển ứng dụng, phù hợp với đối tượng mới bắt đầu và người có kinh nghiệm để phát triển nhanh chóng.

****

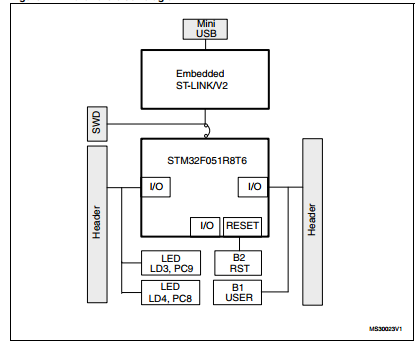
Hình 1: Kit STM32F0 discovery

**Kit STM32F0 Discovery dựa trên STM32F051R8T6 bao gồm tính năng:**

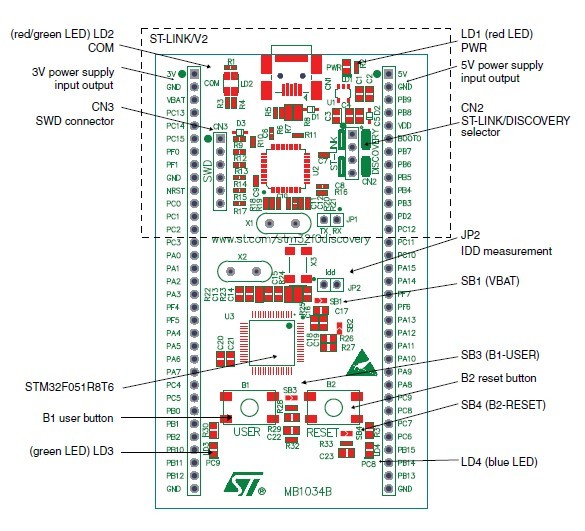
* Bộ nhớ : 64 KB Flash, 8 KB RAM
* Kiểu chân: LQFP64
* Tích hợp mạch nạp và gỡ lỗi ST-LINK / V2.
* Điện áp cung cấp : thông qua cổng USB hoặc từ nguồn điện 5V bên ngoài, nguồn ứng dụng bên ngoài 3 – 5V.

Bốn đèn LED:

* LD1 (đỏ) cho nguồn điện 3,3 V
* LD2 (đỏ / lục) cho giao tiếp USB
* LD3 (xanh lục) cho đầu ra PC9
* LD4 (màu xanh) cho đầu ra PC8
* Hai nút nhấn (reset và lập trình)
* Tất cả các I/O chuẩn 2.54mm
* Các chân header mở rộng cho tất cả các I/O LQFP64 để kết nối với các cảm biến, các thiết bị ngoại vi khác.
* Dưới đây là hình minh họa các kết nối giữa STM32F051R8T6 và các thiết bị ngoại vi của nó (STLINK/V2, nút ấn, đèn LED và header).

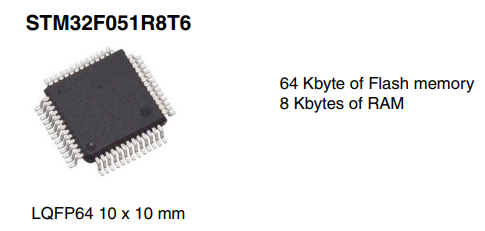


Hình 2: Kết nối STM32F051R8T6 với ngoại vi



Hình 3: Thông số kĩ thuật của STM32F0

STM32F051R8T6 microcontroller



Bộ vi điều khiển STM32F051xx kết hợp ARM® Cortex®-M0 hiệu suất cao Lõi RISC 32 bit hoạt động ở tần số lên đến 48 MHz, bộ nhớ nhúng tốc độ cao (lên đến 64 Kbyte bộ nhớ Flash và 8 Kbyte SRAM). Tất cả các thiết bị cung cấp giao diện truyền thông tiêu chuẩn (tối đa hai I2C, lên đến hai SPI, một I2S, một CEC HDMI và tối đa hai USART), một ADC 12 bit, một 12 bit DAC, sáu bộ định thời 16 bit, một bộ hẹn giờ 32 bit và bộ hẹn giờ điều khiển PWM tiên tiến. Các bộ vi điều khiển STM32F051xx hoạt động ở -40 đến +85 ° C và -40 đến +105 ° C phạm vi nhiệt độ, từ một nguồn cung cấp 2.0 đến 3.6 V. Một bộ các chế độ tiết kiệm toàn diện cho phép thiết kế các ứng dụng năng lượng thấp.

Thiết bị này cung cấp các lợi ích sau.

● Thực thi mã vượt trội để có hiệu suất tốt hơn và hiệu quả mã tuyệt vời cho

giảm sử dụng bộ nhớ nhúng

● Kết nối hiệu suất cao và các thiết bị ngoại vi analog tiên tiến để hỗ trợ rộng phạm vi ứng dụng

● Tùy chọn đồng hồ linh hoạt và chế độ năng lượng thấp với chức năng đánh thức nhanh cho công suất thấp tiêu dùng

Nó có các tính năng chính sau:

● Điều kiện cốt lõi và hoạt động

- ARM® Cortex ™ -M0 0,9 DMIPS / MHz lên đến 48 MHz

- Phạm vi cung cấp 1.8 / 2.0 đến 3.6 V

● Kết nối hiệu suất cao

- USART 6 Mbit / s

- SPI 18 Mbit / s với khung dữ liệu 4 đến 16 bit

- 1 Mbit / s I²C chế độ nhanh cộng

- CEC HDMI

● Kiểm soát nâng cao

- Bộ định thời PWM 5x 16 bit

- Bộ timer cơ bản 1x 16 bit

- Bộ đếm thời gian PWM 32 bit

1. **Sensor DHT11**

DHT11 là cảm biến nhiệt độ và độ ẩm. Nó ra đời sau và được sử dụng thay thế cho dòng SHT1x ở những nơi không cần độ chính xác cao về nhiệt độ và độ ẩm.



Hình 4: Cảm biến DHT11

DHT11 có cấu tạo 3 chân như hình. Nó sử dụng giao tiếp số theo chuẩn 1 dây.

Thông số kĩ thuật:

* Đo độ ẩm: 20%-95%
* Nhiệt độ: 0-50ºC
* Sai số độ ẩm ±5%
* Sai số nhiệt độ: ±2ºC

## Module giao tiếp UART

Module giao tiếp UART CP2102 sử dụng chip CP2102 của hãng SILICON LABS được dùng để chuyển giao tiếp từ USB sang UART TTL và ngược lại.

Module giao tiếp này phù hợp với nhiều hệ điều hành như Windows, Mac, Linux, Android,…rất dễ sử dụng và giao tiếp.

Trên module có 6 cổng đầu ra: 3.3V, 5V, Tx, Rx, GND, DTR/RST. Trong đó, chân DTR được sử dụng để reset vi điều khiển trong quá trình nạp (tương thích với Arduino Promini). Led nguồn sáng khi gắn vào máy tính và led báo hiệu Tx/Rx sẽ sáng khi module nhận/gửi dữ liệu.



Hình 5: Mạch chuyển USB UART CP2102

Mô tả chân:

* TXD: chân truyền dữ liệu UART, dùng kết nối với chân Rx của các module khác.
* RXD: chân nhận dữ liệu UART, dùng kết nối đến chân Tx của các module khác.
* GND: chân nối đất.
* 5V: nguồn điện áp dương (tối đa 500mA).
* 3.3V: nguồn điện áp dương 3.3V
* DTR: chân reset để nạp cho vi điều khiển.

1. **LCD 16x2**



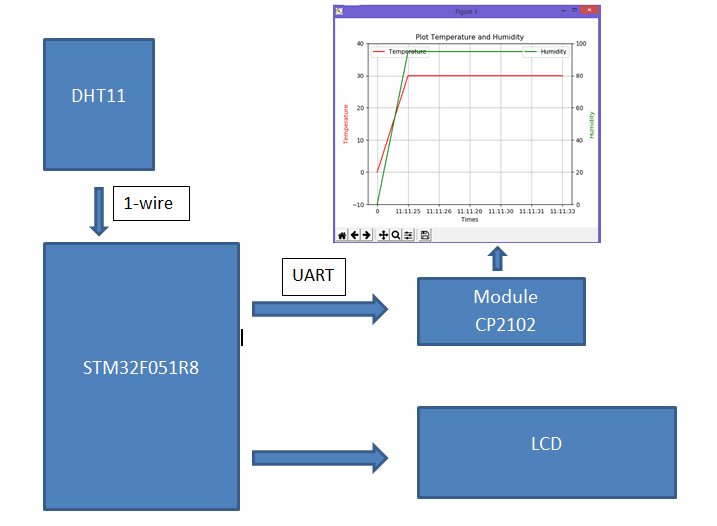
*Hình 6: Hình ảnh LCD 16x2*

|  |  |  |
| --- | --- | --- |
| **Pin** | **Chức năng** | **Tên** |
| 1 | Chân nối đất (GND) cho LCD | VSS |
| 2 | Cấp nguồn cho LCD, điện áp 5V | Vcc |
| 3 | Điều chỉnh độ tương phản thông qua một điện trở thay đổi | VEE |
| 4 | + Logic “0”: bus DB0-DB7 nối với thanh ghi lệnh IR của LCD (ở chế độ ghi) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ đọc)  + Logic “1” : bus DB0-DB7 sẽ nối vớit hanh ghi dữ liệu DB bên trong LCD | RS |
| 5 | Chân chọn chế độ đọc ghi. Nối chân R/W với logic “0” để LCD hoạt động ở chế độ ghi hoặc nối với logic “1” để LCD ở chế độ đọc | RW |
| 6 | Gửi dữ liệu đến các chân dữ liệu khi có xung cao đến thấp | E |
| 7 | Tám đường bus dữ liệu dùng để trao đổi thông tin với MCU, có 2 chế độ:  + Chế độ 8 bit: dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7  + Chế độ 4 bit: dữ liệu được truyền trên 4 đường từ DB4 – DB7, với bit MSB là bit DB7 | DB0 |
| 8 | DB1 |
| 9 | DB2 |
| 10 | DB3 |
| 11 | DB4 |
| 12 | DB5 |
| 13 | DB6 |
| 14 | DB7 |
| 15 | Đèn nền VCC (5V) | A |
| 16 | Đèn nền (0V) | K |

# Nguyên lý hoạt động của mạch

## Nguyên lý chung

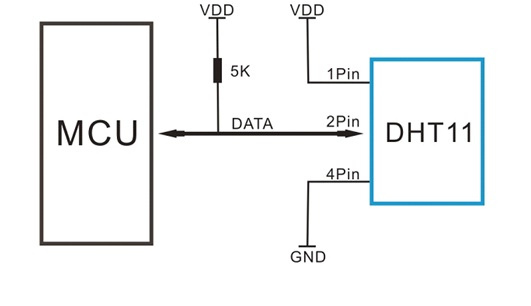
Sử dụng kit stm32f0 discovery thu thập dữ liệu nhiệt độ - độ ẩm từ cảm biến DHT11, giá trị thu thập được được hiển thị lên màn hình LCD 16x2. Thông qua module giao tiếp UART, giá trị đo được được lưu vào database và hiển thị thông qua giao diện trên máy tính. Trên app có phát triển thêm tính năng vẽ biểu đồ thời gian thực giúp người dùng dễ quan sát. Dữ liệu trong database được lưu vào bảng theo từng ngày, giúp cho việc truy xuất và quản lý dễ dàng.



Hình 7: Sơ đồ khối hệ thống

1. **Cách đọc sensor DHT11**

Sơ đồ nguyên lý kết nối với vi xử lý:



Hình 8: Sơ đồ kết nối với MCU

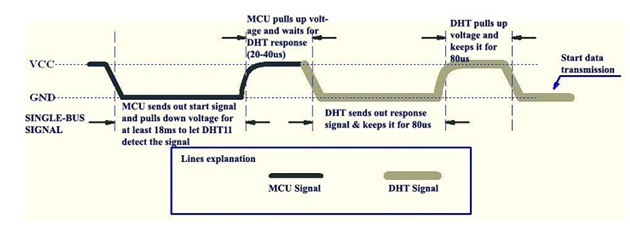
**Nguyên lý hoạt động**:

Để có thể giao tiếp với DHT11 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước:

Gửi tín hiệu muốn đo (Start) tới DHT11, sau đó DHT11 xác nhận lại.

Khi đã giao tiếp được với DHT11, Cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.

Bước 1: gửi dữ liệu START



Hình 9: Gửi dữ liệu Start

MCU thiết lập chân DATA là Output, kéo chân DATA xuống 0 trong khoảng thời gian >18ms. Trong Code mình để 25ms. Khi đó DHT11 sẽ hiểu MCU muốn đo giá trị nhiệt độ và độ ẩm.

MCU đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.

Sau khoảng 20-40us, DHT11 sẽ kéo chân DATA xuống thấp. Nếu >40us mà chân DATA ko được kéo xuống thấp nghĩa là ko giao tiếp được với DHT11.

Chân DATA sẽ ở mức thấp 80us sau đó nó được DHT11 kéo nên cao trong 80us. Bằng việc giám sát chân DATA, MCU có thể biết được có giao tiếp được với DHT11 ko. Nếu tín hiệu đo được DHT11 lên cao, khi đó hoàn thiện quá trình giao tiếp của MCU với DHT.

Bước 2: đọc giá trị lên DHT11

DHT11 sẽ trả giá trị nhiệt độ và độ ẩm về dưới dạng 5 byte. Trong đó:

Byte 1: giá trị phần nguyên của độ ẩm (RH%)

Byte 2: giá trị phần thập phân của độ ẩm (RH%)

Byte 3: giá trị phần nguyên của nhiệt độ (TC)

Byte 4 : giá trị phần thập phân của nhiệt độ (TC)

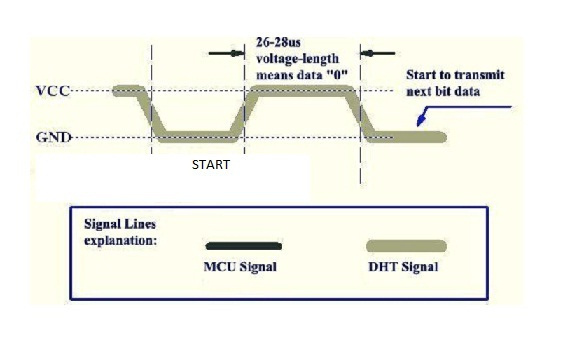
Byte 5 : kiểm tra tổng

Nếu Byte 5 = (8 bit) (Byte1 +Byte2 +Byte3 + Byte4) thì giá trị độ ẩm và nhiệt độ là chính xác, nếu sai thì kết quả đo không có nghĩa.

Đọc dữ liệu:

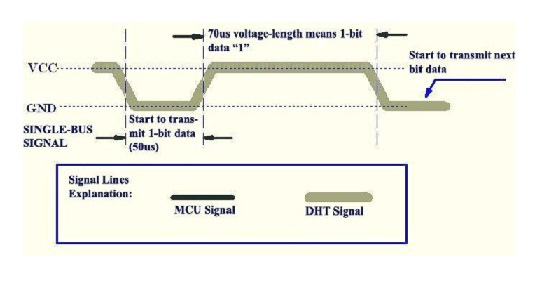
Sau khi giao tiếp được với DHT11, DHT11 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về MCU, tương ứng chia thành 5 byte kết quả của Nhiệt độ và độ ẩm.

Bit 0:



Hình 10: Gửi bít 0

Bit 1:



Hình 11: Gửi bit 1

Sau khi tín hiệu được đưa về 0, ta đợi chân DATA của MCU được DHT11 kéo lên 1. Nếu chân DATA là 1 trong khoảng 26-28 us thì là 0, còn nếu tồn tại 70us là 1. Do đó trong lập trình ta bắt sườn lên của chân DATA, sau đó delay 50us. Nếu giá trị đo được là 0 thì ta đọc được bit 0, nếu giá trị đo được là 1 thì giá trị đo được là 1. Cứ như thế ta đọc các bit tiếp theo.

Kết nối với STM32 như sau:

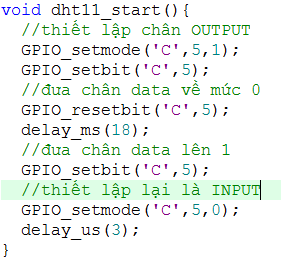
PC5 ——————-DATA(DHT11)

VCC——————-VCC (5V)

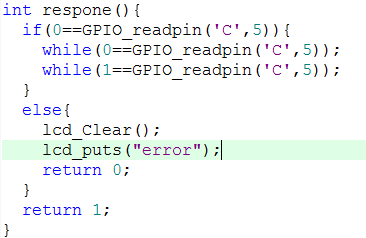
GND——————GND(DHT11)

Phần mềm sử dụng để lập trình: KEIL C V5

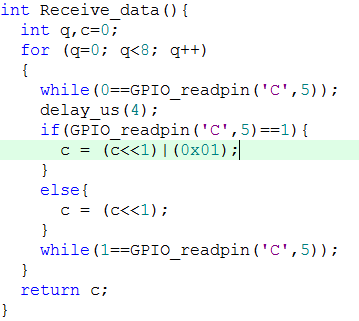
Đầu tiên, tại file main.c ta khởi tạo hàm dht11\_start() tương ứng với bước 1 phía trên.

****

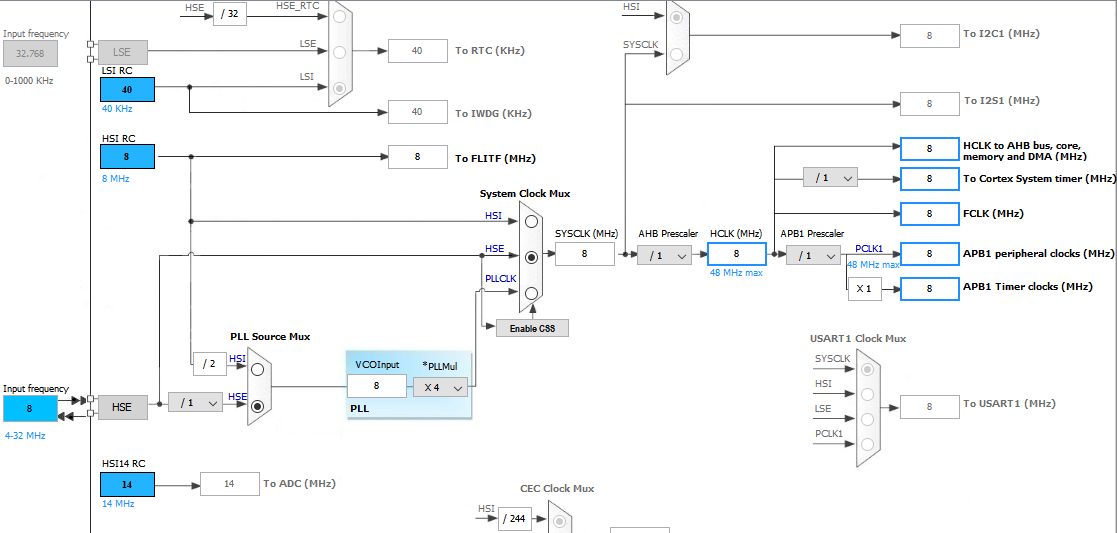
Tiếp theo xây dựng hàm phản hồi (response): đợi chân data lên mức 1 xong lại đợi chân data về mức 0 (đợi đến khi nào chân data không còn bằng 0 nữa thì thoát vòng while trên và nhảy vào vòng while dưới).



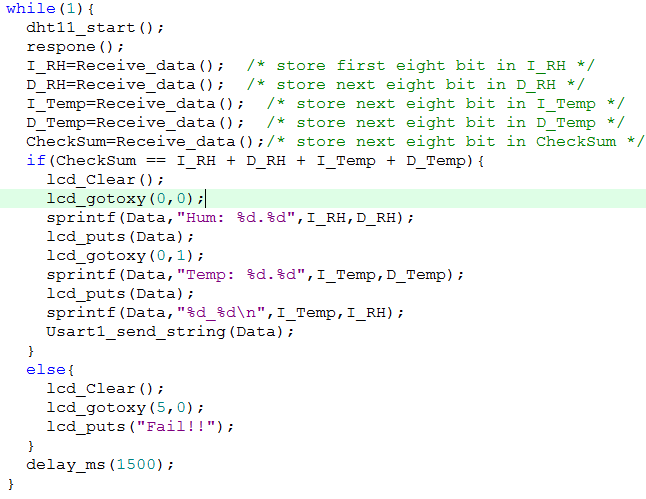
Hàm nhận dữ liệu: DHT11 sẽ trả giá trị nhiệt độ và độ ẩm về dưới dạng 5 byte, mỗi byte 8 bit dữ liệu. Dùng vòng lặp for quét 8 bit dữ liệu, kiểm tra nếu chân data bằng “1” thì dịch bit 1 vào bit đầu tiên của byte c, ngược lại sẽ dịch bit “0” vào. Cứ như vậy, sau khi quét 8 bit, hàm sẽ trả về byte c.



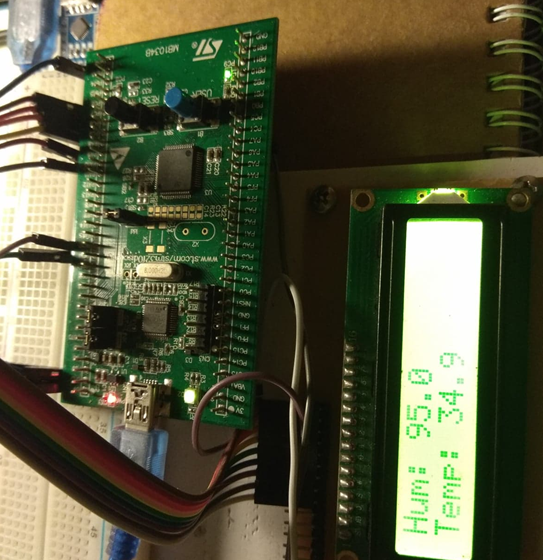
Trong chương trình chính, khai báo các hàm khởi tạo gồm: LCD, RCC\_HSE(sử dụng thạch anh ngoại 8MHz), RCC\_UART, systick, …



Trong vòng lặp chương trình chính, khởi tạo dht11 bằng hàm dht11\_start(), sau đó đợi phản hồi từ dht, dht sẽ gửi 5 byte bao gồm: byte 1( phần nguyên độ ẩm – I\_RH ), byte 2(phần thập phân độ ẩm – D\_RH), byte 3(phần nguyên nhiệt độ - I\_Temp), byte 4(phần thập phân nhiệt độ - D\_Temp) và cuối cùng là byte CheckSum. Kiểm tra xem nếu byte CheckSum bằng tổng 4 byte trước thì đồng nghĩa với việc dht nhận được dữ liệu đúng, lưu giá trị nhiệt độ - độ ẩm vào chuỗi Data, hiển thị chuỗi Data lên LCD đồng thời gửi lên máy tính thông qua UART.



Dưới đây là hình ảnh hiển thị nhiệt độ - độ ẩm lên LCD:



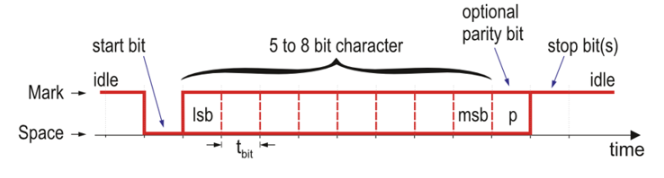
1. **Truyền dữ liệu qua UART**

Khác với kênh truyền song song, trong kênh truyền nối tiếp ta chỉ cần một đường truyền để truyền tuần tự n bit của gói dữ liệu.

*Khái quát chung về UART:*

Tất cả các kênh truyền nối tiếp đều yêu cầu tín hiệu clock nhằm thiết lập tốc độ tín hiệu truyền nhận dữ liệu giữa các bên cho khớp với nhau. Tùy thuộc vào vào cách sử dụng tín hiệu clock như thế nào mà chúng ta có định nghĩa về kênh truyền đồng bộ hay kênh truyền không đồng bộ.

  Kênh truyền không đồng bộ sử dụng bộ tạo tín hiệu clock độc lập ở mỗi thiết bị, dữ liệu được truyền theo một khung truyền và tốc độ baud được thống nhất bởi bên truyền và bên nhận.  Còn đối với kênh truyền đồng bộ sẽ có những tín hiệu clock được truyền đồng thời với dữ liệu trên một đường truyền nối giữa hai thiết bị.



Hình 12: Truyền dữ liệu nối tiếp

Start là bit đầu tiên được truyền trong một frame truyền, bit này có chức năng báo cho thiết bị nhận biết rằng có một gói dữ liệu sắp được truyền tới. Start là bit bắt buộcphải có trong khung truyền, và nó là một bit thấp (0).

    Data hay dữ liệu cần truyền là thông tin chính mà chúng ta cần gởi và nhận. Số lượng bit data tùy thuộc vào các loại vi điều khiển khác nhau, thường thì data có 8bit. Trong truyền thông nối tiếp UART, bit có trọng số nhỏ nhất (LSB – Least Significant Bit, bit bên phải) của data sẽ được truyền trước và cuối cùng là bit có trọng số lớn nhất (MSB – Most Significant Bit, bit bên trái). Tuy nhiên thứ tự truyền này có thể được cài đặt bởi người dùng.

   Parity là bit dùng để kiểm tra dữ liệu truyền có đúng không (một cách tương đối). Có 2 loại parity là parity chẵn (even parity) và parity lẻ (odd parity). Parity chẵn nghĩa là số lượng số “1” trong dữ liệu bao gồm bit parity luôn là số chẵn. Ngược lại tổng số lượng các số “1” trong parity lẻ luôn là số lẻ (Parity bit không phải là bit bắt buộc và vì thế chúng ta có thể loại bit này khỏi khung truyền).

  Stop bits là 01 hoặc nhiều bit báo cho thiết bị nhận rằng một gói dữ liệu đã được gởi xong. Sau khi nhận được stop bits, thiết bị nhận sẽ tiến hành kiểm tra khung truyền để đảm bảo tính chính xác của dữ liệu. Stop bits là các bit bắt buộc xuất hiện trong khung truyền.

Bất kỳ giao tiếp hai chiều USART nào cũng yêu cầu tối thiểu hai chân: Nhận dữ liệu (RX) và truyền dữ liệu ra (TX):

• RX: Nhận dữ liệu đầu vào.

• TX: Truyền dữ liệu đầu ra.

Dữ liệu nối tiếp được truyền và nhận thông qua các chân của STM32 này trong chế độ USART bình thường, bao gồm:

• Một start bit

• Chiều dài dữ liệu (7, 8 hoặc 9 bit)

• Các bit dừng 0,5, 1, 1,5, 2 chỉ ra rằng khung đã hoàn thành

• Giao diện USART sử dụng trình tạo tốc độ baud

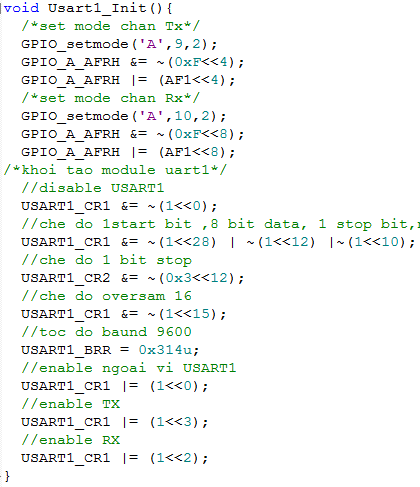
• Một thanh ghi trạng thái (USART\_ISR)

• Nhận và truyền các thanh ghi dữ liệu (USART\_RDR, USART\_TDR)

• Thanh ghi tốc độ baud (USART\_BRR)

• Thanh ghi thời gian bảo vệ (USART\_GTPR) trong trường hợp chế độ Smartcard mode.

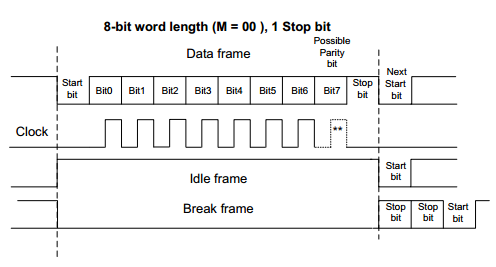
Dưới đây là hàm khởi tạo USART: 1 bit start, 1 bit stop, 8 bit data, tốc độ baund 9600, cho chép chân truyền và nhận dữ liệu.



Để đạt được tốc độ Baund 9600, với tần số CLK fCK = 8MHz, trong trường hợp oversampling bằng 16, ta sử dụng công thức sau:

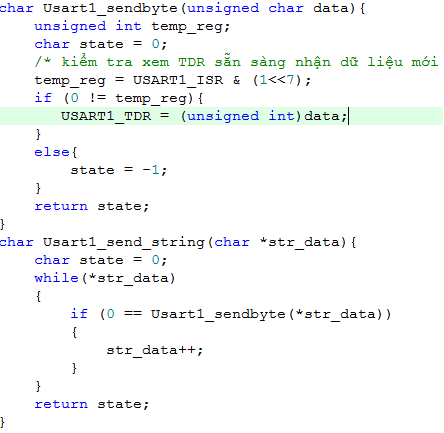


Khi đó, USARTDIV = 8 000 000/9600, BRR = USARTDIV = 833d = 0314h.



Hình 13: Truyền dữ liệu nối tiếp 8 bit của STM32

UART gửi các byte và chuỗi tới máy chủ thông qua 2 hàm sau:



Trong hàm Usart\_send\_string(), khao báo 1 con trỏ mang tên \*str\_data, trỏ đến byte đầu tiên của chuỗi, sau đó kiểm tra việc truyền từng byte có được thực hiện hay không? Nếu đúng thì cứ tăng con trỏ cho tới khi gửi hết từng byte của chuỗi.

1. **Cách hiển thị dữ liệu lên máy tính và vẽ biểu đồ**

Dữ liệu sau khi được truyền qua UART lên máy tính, sẽ được lưu vào database. Việc hiển thị dữ liệu lên app được lấy từ trong bảng của database và vẽ biểu đồ thời gian thực.

Xây dựng app trên máy tính sử dụng Python (Tkinter):

*Giới thiệu chung về Python*:

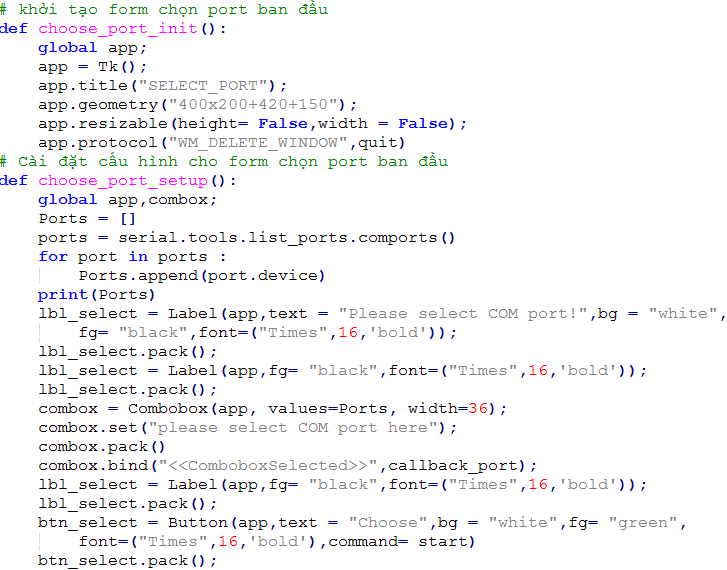
Python là một ngôn ngữ lập trình hướng đối tượng, python hoàn toàn tạo kiểu động, cấp phát bộ nhớ động. Mục đích ra đời của Python là cung cấp một ngôn ngữ lập trình có cấu trúc rõ ràng, sáng sủa, thuận tiện cho người mới học lập trình. Python được phát triển bởi Guido và Rossum. Phiên bản đầu tiên được phát hành vào năm 1991. Python được lấy cảm hứng từ ABC, Haskell, Java, Lisp, Icon và Perl. Python là một ngôn ngữ thông dịch, đa nền tảng. Một trong những đặc điểm độc nhất của Python là ngôn ngữ này không dùng đến dấu chấm phẩy, dấu mở - đóng ngoặc {} để kết thúc câu lệnh hay khối lệnh, mà cách duy nhất để nó nhận biết một lệnh là dấu thụt đầu dòng.

Hiện tại Python có hai dòng phiên bản là dòng 2.x và 3.x. Phiên bản Python mà chúng em sử dụng trong đề tài này là phiên bản **3.7.2.** Ngày nay Python được phát triển bởi một cộng đồng tình nguyện trên khắp thế giới, và lẽ dĩ nhiên nó là một phần mềm mã nguồn mở.

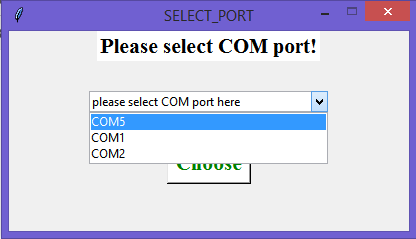
*Giới thiệu chung về Tkinter*:

Tkinter là một gói trong Python có chứa module Tk hỗ trợ cho việc lập trình GUI. Tk ban đầu được viết cho ngôn ngữ **Tcl**. Sau đó Tkinter được viết ra để sử dụng Tk bằng trình thông dịch Tcl trên nền Python. Ngoài Tkinter ra còn có một số công cụ khác giúp tạo một ứng dụng GUI viết bằng Python như wxPython, PyQt, và PyGTK.

Để sử dụng Tkinter, tất cả những gì chúng ta cần làm là khai báo module Tkinter bằng cách “from tkinter import\*”, ngoài ra nếu muốn sử dụng các công cụ như vẽ đồ thị, thời gian thực, sql, … ta khai báo các module

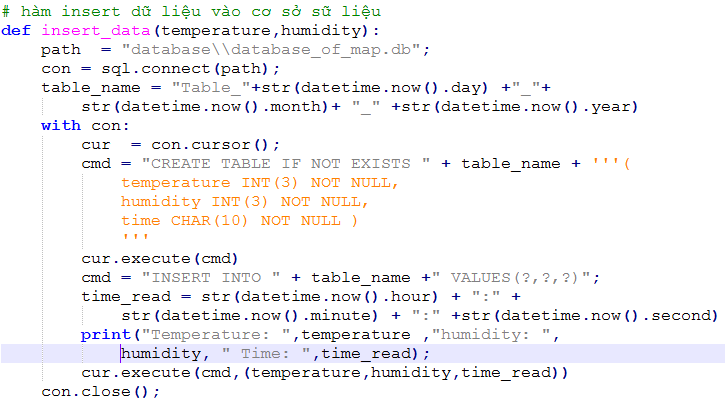


Đầu tiên, xây dựng form chọn port, bằng cách quét hết các cổm COM đang kết nối ta có thể chọn port phù hợp khi giao tiếp với UART, sử dụng câu lệnh “serial.tools.list\_ports.comports()” để quét các cổng COM, danh sách các cổng COM sẽ được lưu vào một mảng có tên Ports. Dưới đây là hình ảnh sau khi build:

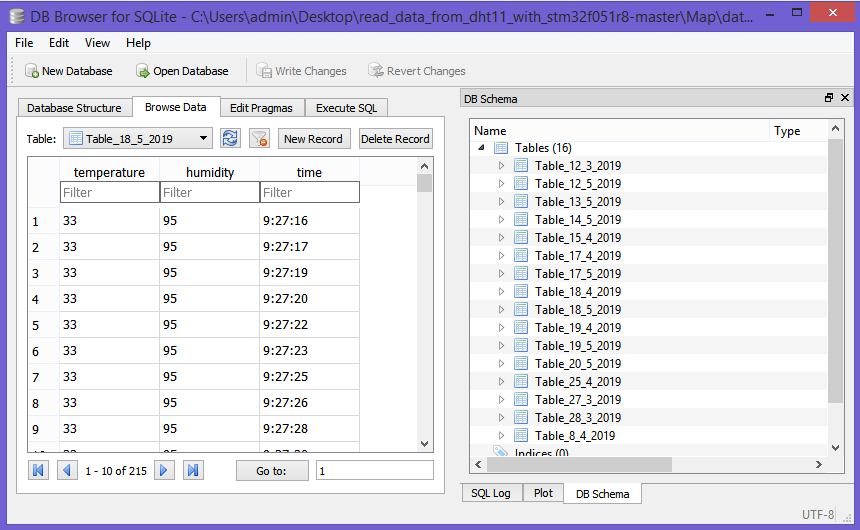


Hình 14: Giao diện hiển thị khi chọn port

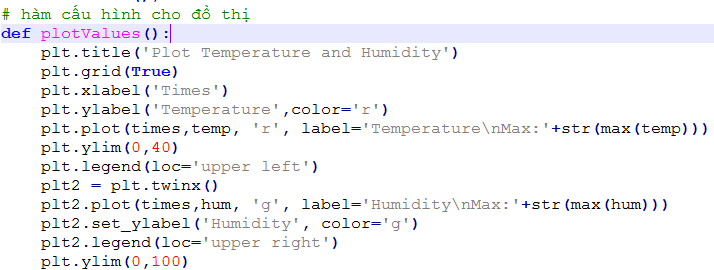
Hàm tạo bảng, insert giá trị nhiệt độ - độ ẩm vào bảng trong DB:



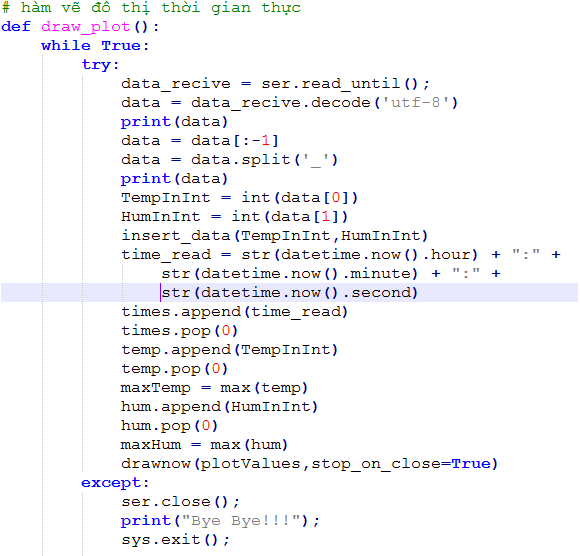
Việc truy xuất dữ liệu nhiệt độ - độ ẩm được thực hiện dễ dàng khi giá trị của 2 tham số này được lưu vào DB, mở thư mục database, giá trị nhiệt độ - độ ẩm được lưu vào bảng theo từng ngày như sau:



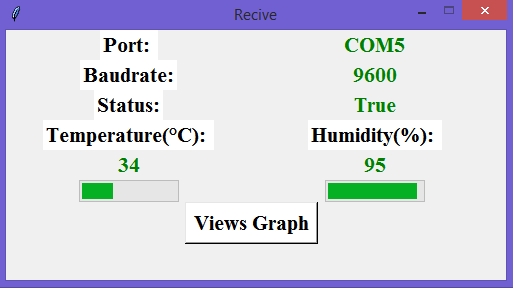
Hàm cấu hình cho đồ thị: vẽ đồ thị thời gian thực 2D, với trục đứng bên phải là trục Humidity, trục đứng bên trái là trục Temperature, trục nằm ngang chung là trục Times, đồng thời hiển thị giá trị cao nhất của temp và hum.



Hàm lấy dữ liệu từ database và vẽ đồ thị thời gian thực:

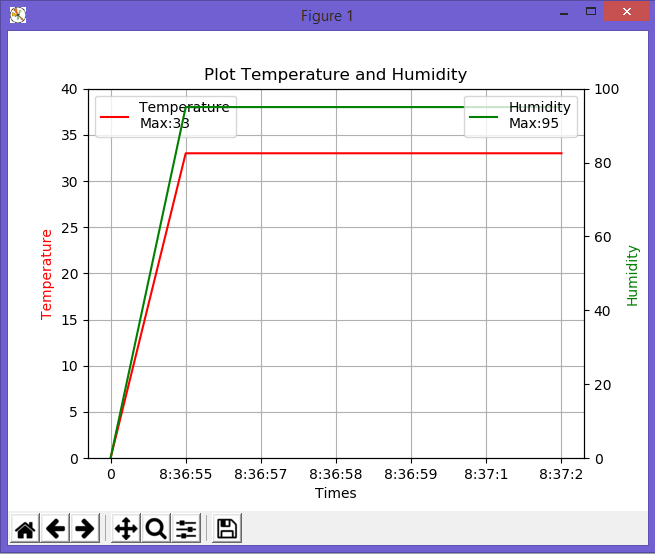


Dưới đây là hình ảnh app hoàn chỉnh sử dụng Tkinter:



Hình 15: Giao diện hiển thị nhiệt độ, độ ẩm

Sau khi nhấn nút “Views Graph” đồ thị 2D sẽ hiện ra như sau:



Hình 16: Giao diện đồ thị 2D sử dụng Tkinter

1. **Kết luận và hướng phát triển đề tài**
2. **Kết luận**

Đề tài thực hiện khá thành công, hiển thiện nhiệt độ - độ ẩm ổn định, giao diện xây dựng dễ nhìn. Tuy nhiên, đề tài còn một số nhược điểm như không theo dõi được ở khoảng cách xa, app xây dựng còn đơn giản.

## Hướng phát triển của đề tài

Với đề tài này chúng ta có thể phát triển với quy mô lớn hơn, không cần đến địa điểm trực tiếp theo dõi mà chỉ cần quan sát thông qua màn hình máy vi tính.

Nó là tiền đề để phát triển hệ thống nhà thông minh, tưới tiêu tự động, ứng dụng trong cảnh báo cháy nổ - đóng ngắt,…

# Tài liệu tham khảo

[1] <https://www.dientu360.com/kit-stm32f0-discovery-stm32f051r8t6>

[2] <https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python>

[3] <https://phocode.com/blog/2016/01/30/tkinter-gioi-thieu-ve-tkinter/>

[4] <https://datasheetspdf.com/pdf/732090/STMicroelectronics/STM32F051R8/9>

**Phụ Lục**

* **Source code STM32**

#include"hw\_stm32f051r8.h"

#include"sw\_rcc.h"

#include"sw\_gpio.h"

#include"sw\_interrupt.h"

#include"sw\_delay.h"

#include"sw\_lcd1602.h"

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include"sw\_uart.h"

/\*

\_\_\_\_\_\_\_\_\_\_\_DHT11\_\_\_\_\_\_\_\_\_

-----DATA------------PC5

\_\_\_\_\_\_\_\_\_\_\_LCD\_\_\_\_\_\_\_\_\_\_\_

-----RS------------PC10

-----EN------------PC11

-----DB4------------PC9

-----DB5------------PC8

-----DB6------------PC7

-----DB7------------PC6

\*/

void dht11\_start(){

//thiết lập chân OUTPUT

GPIO\_setmode('C',5,1);

GPIO\_setbit('C',5);

//đưa chân data về mức 0

GPIO\_resetbit('C',5);

delay\_ms(18);

//đưa chân data lên 1

GPIO\_setbit('C',5);

//thiết lập lại là INPUT

GPIO\_setmode('C',5,0);

delay\_us(3);

}

int I\_RH,D\_RH,I\_Temp,D\_Temp,CheckSum;

int respone(){

if(0==GPIO\_readpin('C',5)){

while(0==GPIO\_readpin('C',5));

while(1==GPIO\_readpin('C',5));

}

else{

lcd\_Clear();

lcd\_puts("error");

return 0;

}

return 1;

}

int Receive\_data(){

int q,c=0;

for (q=0; q<8; q++)

{

while(0==GPIO\_readpin('C',5));

delay\_us(4);

if(GPIO\_readpin('C',5)==1){

c = (c<<1)|(0x01);

}

else{

c = (c<<1);

}

while(1==GPIO\_readpin('C',5));

}

return c;

}

int main(){

char Data[40];

RCC\_HSE\_init();

RCC\_Usart\_init();

systick\_init();

RCC\_EnClkIOport('A');

RCC\_EnClkIOport('C');

init\_pin();

lcd\_Init();

lcd\_Clear();

lcd\_gotoxy(0,0);

Usart1\_Init();

while(1){

dht11\_start();

respone();

I\_RH=Receive\_data(); /\* store first eight bit in I\_RH \*/

D\_RH=Receive\_data(); /\* store next eight bit in D\_RH \*/

I\_Temp=Receive\_data(); /\* store next eight bit in I\_Temp \*/

D\_Temp=Receive\_data(); /\* store next eight bit in D\_Temp \*/

CheckSum=Receive\_data();/\* store next eight bit in CheckSum \*/

if(CheckSum == I\_RH + D\_RH + I\_Temp + D\_Temp){

lcd\_Clear();

lcd\_gotoxy(0,0);

sprintf(Data,"Hum: %d.%d",I\_RH,D\_RH);

lcd\_puts(Data);

lcd\_gotoxy(0,1);

sprintf(Data,"Temp: %d.%d",I\_Temp,D\_Temp);

lcd\_puts(Data);

sprintf(Data,"%d\_%d\n",I\_Temp,I\_RH);

Usart1\_send\_string(Data);

}

else{

lcd\_Clear();

lcd\_gotoxy(5,0);

lcd\_puts("Fail!!");

}

delay\_ms(1500);

}

}

void RCC\_IRQHandler(){

//RCC\_CIR->HSERDYF(bit 3)

if((RCC\_CIR & (1<<3))!=0){

//RCC\_CIR->HSERDYC(bit 19) xoa co ngat

RCC\_CIR |= (1<<19);

//chuyen doi thanh muc HSEON(bit[1:0])

RCC\_CFGR &= ~(0x3<<0);

RCC\_CFGR |= (0x01<<0);

\_\_asm("cpsid i");//tat cac ngat toan cuc

}

}

void USART1\_IRQHandler(){

}

void SystemInit(){

RCC\_CR |= 0x00000001u;

RCC\_CFGR &= 0xF8FFB80Cu;

/\* Reset HSEON, CSSON and PLLON bits \*/

RCC\_CR &= 0xFEF6FFFFU;

/\* Reset HSEBYP bit \*/

RCC\_CR &= 0xFFFBFFFFU;

/\* Reset PLLSRC, PLLXTPRE and PLLMUL[3:0] bits \*/

RCC\_CFGR &= 0xFFC0FFFFU;

/\* Reset PREDIV[3:0] bits \*/

RCC\_CFGR &= 0xFFFFFFF0U;

// Reset USART1SW[1:0], I2C1SW, CECSW and ADCSW bits

RCC\_CFGR3 &= 0xFFFFFEACU;

/\* Reset USART1SW[1:0], I2C1SW, USBSW and ADCSW bits \*/

RCC\_CFGR3 &= 0xFFFFFE6CU;

// Set default USB clock to PLLCLK, since there is no HSI48

RCC\_CFGR3 |= 0x00000080u;

// /\* Reset HSI14 bit \*/

RCC\_CR2 &= 0xFFFFFFFEu;

// Disable all interrupts

RCC\_CIR = 0x00000000u;

}

* **Source code đồ thị :**

**File : recive.py**

import sqlite3 as sql

from datetime import datetime

import random

import time

import serial

from threading import\*

from tkinter import\*

from tkinter import ttk

from PIL import ImageTk,Image

import os

import sys

from tkinter.ttk import Combobox

from tkinter import messagebox

import serial.tools.list\_ports

import read\_data as live\_plot

port\_select = -1;

ser = serial.Serial();

# hàm được gọi khi chọn 1 giá trị trong combobox

def callback\_port(event):

global combox,port\_select;

port\_select = str(combox.get())

print(port\_select)

# Hàm khởi tạo cổng nối tiếp( tốc độ baund,port, timeout) và khởi tạo form hiển thị chính

def start():

global port\_select,t,check\_recive,ser;

if(port\_select==-1):

print("Please select")

else:

print("OK!!!",port\_select)

ser.port = str(port\_select)

ser.baudrate = 9600

ser.timeout = 2

try:

ser.open();

x = 1;

except:

x = 0;

if(x==1):

app.destroy();

ui\_init();

setup\_ui();

t.start();

check\_recive = 1;

ui.mainloop();

else:

print("Please choose others port!")

messagebox.showwarning("Error", "Port is aready\nPlease choose other port")

# hàm insert dữ liệu nhiệt độ độ ẩm vào databases

def insert\_data(temperature,humidity):

path = "database\\database\_of\_map.db";

con = sql.connect(path);

table\_name = "Table\_"+str(datetime.now().day) +"\_"+str(datetime.now().month)+ "\_" +str(datetime.now().year)

with con:

cur = con.cursor();

cmd = "CREATE TABLE IF NOT EXISTS " + table\_name + '''(

temperature INT(3) NOT NULL,

humidity INT(3) NOT NULL,

time CHAR(10) NOT NULL )

'''

cur.execute(cmd)

cmd = "INSERT INTO " + table\_name +" VALUES(?,?,?)";

time\_read = str(datetime.now().hour) + ":" + str(datetime.now().minute) + ":" +str(datetime.now().second)

print("Temperature: ",temperature ,"humidity: ",humidity, " Time: ",time\_read);

cur.execute(cmd,(temperature,humidity,time\_read))

con.close();

# khởi tạo form chọn port ban đầu

def choose\_port\_init():

global app;

app = Tk();

app.title("SELECT\_PORT");

app.geometry("400x200+420+150");

app.resizable(height= False,width = False);

app.protocol("WM\_DELETE\_WINDOW",quit)

# Cài đặt cấu hình cho form chọn port ban đầu

def choose\_port\_setup():

global app,combox;

Ports = []

ports = serial.tools.list\_ports.comports()

for port in ports :

Ports.append(port.device)

print(Ports)

lbl\_select = Label(app,text = "Please select COM port!",bg = "white",fg= "black",font=("Times",16,'bold'));

lbl\_select.pack();

lbl\_select = Label(app,fg= "black",font=("Times",16,'bold'));

lbl\_select.pack();

combox = Combobox(app, values=Ports, width=36);

combox.set("please select COM port here");

combox.pack()

combox.bind("<<ComboboxSelected>>",callback\_port);

lbl\_select = Label(app,fg= "black",font=("Times",16,'bold'));

lbl\_select.pack();

btn\_select = Button(app,text = "Choose",bg = "white",fg= "green",font=("Times",16,'bold'),command= start)

btn\_select.pack();

# hàm được gọi khi nhấn nút close đỏ trên màn hình

def quit():

global check\_recive;

x = messagebox.askokcancel("Quit","Do you want to quit?");

if(x == True):

check\_recive = 0;

sys.exit();

try:

ui.destroy();

except:

print("khong co ui");

try:

app.destroy();

except:

print("khong co app");

exit();

# hàm khởi tạo cho form hiển thị chính

def ui\_init():

global ui;

ui = Tk();

ui.title("Recive");

ui.geometry("500x250");

ui.resizable(height= False,width = False);

ui.protocol("WM\_DELETE\_WINDOW",quit)

# hàm được gọi khi nhấn nút vẽ đồ thị

def graph():

global ser,check\_recive,port\_select;

ser.close();

ui.destroy();

check\_recive = 0;

live\_plot.data\_init();

live\_plot.serial\_init(str(port\_select));

live\_plot.draw\_plot();

# hàm cài đặt cấu hình cho form hiển thị chính

def setup\_ui():

global port,baudrate,status,temperature,humidity,ser,data,port\_select;

data = [0,0];

#print("please check other port")

port = StringVar();

baudrate = StringVar();

status = StringVar();

temperature = IntVar();

humidity = IntVar();

port.set(str(ser.port));

baudrate.set(str(ser.baudrate))

status.set(str(ser.is\_open))

temperature.set(0);

humidity.set(0);

lbl\_port = Label(ui,text = "Port: ",bg = "white",fg= "black",font=("Times",16,'bold'));

lbl\_baudrate = Label(ui,text = "Baudrate:",bg="white",fg = "black",font=("Times",16,'bold'));

lbl\_status = Label(ui,text = "Status:",bg="white",fg = "black",font=("Times",16,'bold'));

lbl\_temp = Label(ui,text = "Temperature: ",bg="white",fg = "black",font=("Times",16,'bold'));

lbl\_hum = Label(ui,text = "Humidity: ",bg="white",fg = "black",font=("Times",16,'bold'));

btn\_viewgraph = Button(ui,text = "Views Graph",bg="white",fg = "black",font=("Times",16,'bold'),command = graph);

lbl\_port.grid(row = 0,column = 0);

lbl\_baudrate.grid(row = 1,column = 0);

lbl\_status.grid(row = 2,column = 0);

lbl\_temp.grid(row = 3,column = 0);

lbl\_hum.grid(row = 3, column = 1);

lbl\_port\_val = Label(ui,textvariable = port,font=("Times",16,'bold'),height = 1,width = 20,fg = "green");

lbl\_baudrate\_val = Label(ui,textvariable = baudrate,font=("Times",16,'bold'),height = 1,width = 20,fg = "green");

lbl\_status\_val = Label(ui,textvariable = status,font=("Times",16,'bold'),height = 1,width = 20,fg = "green");

lbl\_temp\_val = Label(ui,textvariable = temperature,font=("Times",16,'bold'),height = 1,width = 20,fg = "green");

lbl\_hum\_val = Label(ui,textvariable = humidity,font=("Times",16,'bold'),height = 1,width = 20,fg = "green");

progressbar\_temp = ttk.Progressbar(ui,mode='determinate',orient="horizontal", length= 100,variable = temperature,maximum = 100)

progressbar\_hum = ttk.Progressbar(ui,mode='determinate',orient="horizontal", length= 100,variable = humidity,maximum = 100)

lbl\_port\_val.grid(row = 0,column = 1);

lbl\_baudrate\_val.grid(row = 1,column = 1);

lbl\_status\_val.grid(row = 2,column = 1);

lbl\_temp\_val.grid(row = 4,column = 0);

lbl\_hum\_val.grid(row = 4,column = 1);

progressbar\_temp.grid(row = 5,column = 0)

progressbar\_hum.grid(row = 5,column = 1)

btn\_viewgraph.grid(row = 6,column = 0,columnspan=2)

# hàm đọc dữ liệu từ cổng COM và hiển thị dữ liệu lên form chính

def recive\_data():

global temperature,humidity,check\_recive,ser,data;

print("Connected!")

while check\_recive == 1:

try:

data\_recive = ser.read\_until();

data = data\_recive.decode('utf-8')

data = data[:-1]

data = data.split('\_')

print(data)

temperature.set(data[0]);

humidity.set(data[1]);

insert\_data(data[0],data[1]);

except:

print("Please check port connect!!")

if \_\_name\_\_ == '\_\_main\_\_':

global check\_recive,t;

t = Timer(0,recive\_data);

check\_recive = 0;

choose\_port\_init();

choose\_port\_setup();

mainloop();

**File : read\_data.py**

import serial

import matplotlib.pyplot as plt

from drawnow import \*

from datetime import datetime

import sys

import sqlite3 as sql

temp = []

hum = []

data = [0,0]

times = []

plt.ion()

cnt = 0

maxTemp = 0;

maxHum = 0;

ser = serial.Serial();

# hàm khởi tạo cổng nối tiếp

def serial\_init(port\_select):

ser.port = str(port\_select)

ser.baudrate = 9600

ser.timeout = 2

ser.open();

# hàm insert dữ liệu vào cơ sở sữ liệu

def insert\_data(temperature,humidity):

path = "database\\database\_of\_map.db";

con = sql.connect(path);

table\_name = "Table\_"+str(datetime.now().day) +"\_"+str(datetime.now().month)+ "\_" +str(datetime.now().year)

with con:

cur = con.cursor();

cmd = "CREATE TABLE IF NOT EXISTS " + table\_name + '''(

temperature INT(3) NOT NULL,

humidity INT(3) NOT NULL,

time CHAR(10) NOT NULL )

'''

cur.execute(cmd)

cmd = "INSERT INTO " + table\_name +" VALUES(?,?,?)";

time\_read = str(datetime.now().hour) + ":" + str(datetime.now().minute) + ":" +str(datetime.now().second)

print("Temperature: ",temperature ,"humidity: ",humidity, " Time: ",time\_read);

cur.execute(cmd,(temperature,humidity,time\_read))

con.close();

# hàm cấu hình cho đồ thị

def plotValues():

plt.title('Dữ Liệu Từ Trung tâm Khí Tượng Thủy Văn\nTam Mao')

plt.grid(True)

plt.xlabel('Times')

plt.ylabel('Temperature',color='r')

plt.plot(times,temp, 'r', label='Temperature\nMax:'+str(maxTemp))

plt.ylim(0,40)

plt.legend(loc='upper left')

plt2 = plt.twinx()

plt2.plot(times,hum, 'g', label='Humidity\nMax:'+str(maxHum))

plt2.set\_ylabel('Humidity', color='g')

plt2.legend(loc='upper right')

plt.ylim(0,100)

# hàm khởi tạo dữ liệu ban đầu

def data\_init():

for i in range(0,10):

temp.append(0)

hum.append(0);

times.append(0)

# hàm vẽ đồ thị thời gian thực

def draw\_plot():

while True:

try:

data\_recive = ser.read\_until();

data = data\_recive.decode('utf-8')

print(data)

data = data[:-1]

data = data.split('\_')

print(data)

TempInInt = int(data[0])

HumInInt = int(data[1])

insert\_data(TempInInt,HumInInt)

time\_read = str(datetime.now().hour) + ":" + str(datetime.now().minute) + ":" +str(datetime.now().second)

times.append(time\_read)

times.pop(0)

temp.append(TempInInt)

temp.pop(0)

maxTemp = max(temp)

hum.append(HumInInt)

hum.pop(0)

maxHum = max(hum)

drawnow(plotValues,stop\_on\_close=True)

except:

ser.close();

print("Bye Bye!!!");

sys.exit();