**TH8.1:**

\*Khai báo các thư viện cần dùng

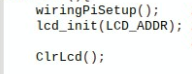


* Do không có nhiều thư viện C hỗ trợ điều khiển LCD16x2 qua I2C và những bài hướng dẫn trên mạng đa số đều dùng Python. Nên nhóm em tự viết một thư viện C đơn giản, cung cấp phương thức điều khiển LCD 16x2 qua giao tiếp I2C bằng ngôn ngữ C. Chi tiết thư viện có tại link: <https://github.com/vinhcatba/i2c1602>

\*Khai báo địa chỉ I2C và tạo struct chứa địa chỉ I2C

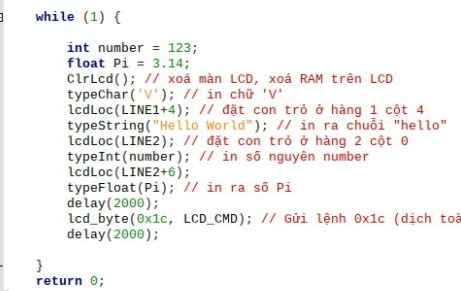


\*Các hàm cần thiết



* wiringPiSetup() mặc dù không sử dụng các hàm của thư viện wiringPi, vẫn cần phải gọi hàm wiringPiSetup() để lấy thời gian cho các hàm delay() và delayMicroseconds() dùng trong thư viện i2c1602.h.
* lcd\_init (LCD\_ADDR) dùng để khởi tạo lcd tại địa chỉ LCD\_ADDR, sau khi khởi tạo thì lcd sẽ đặt ở chế độ ghi 4-bit, hiển thị con trỏ tắt, nháy con trỏ tắt, hướng con trỏ từ trái sang phải, xoá nội dung màn hình và nội dung RAM của lcd.

\*Hàm while()  
Có thể ghi nhiều loại dữ liệu khác nhau lên LCD bằng các hàm typeInt, typeFloat, typeChar và typeString. Để di chuyển vị trí con trỏ có thể sử dụng hàm lcdLoc(LINEx+y) trong đó x có thể là LINE1 hoặc LINE2 tương ứng với hàng 1 và hàng 2. y có thể là các số nguyên từ 0 tới 15 tương ứng với 16 cột.



Kết quả chạy:



**TH8.2:**

+Trên Raspberry (master)

\*Khai báo các thư viện cần dùng



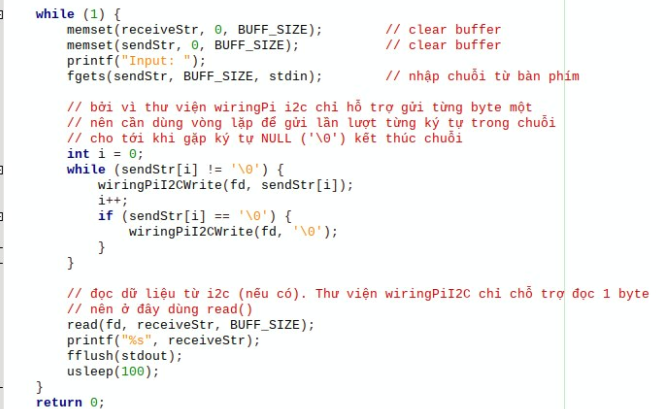
\*Khai báo địa chỉ I2C của Arduino và kích thức Buffer



\*Mở file i2c và khai báo các buffer

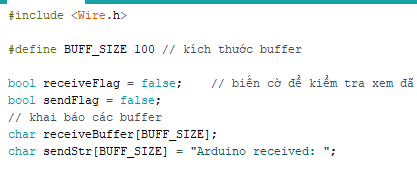


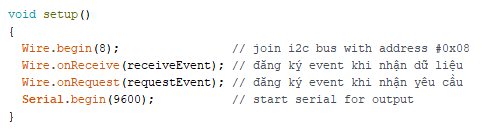
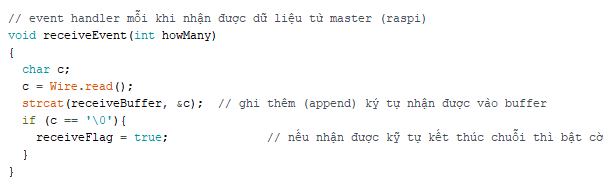
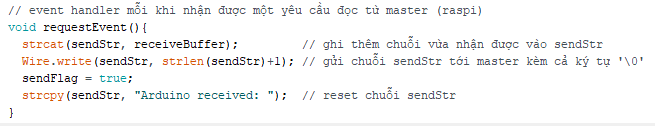
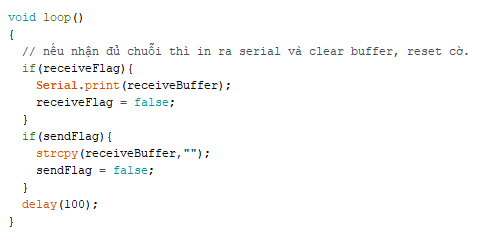
\*Hàm while(1)



**+**Trên Arduino (slave):

1. Sử dụng thư viện Wire.h, khai báo kích thước buffer và các buffer. Khai báo các biến cờ để kiểm tra việc nhận và gửi chuỗi.



1. Hàm setup(): Đăng ký event nhận dữ liệu và event nhận request.  
   
2. Event handler cho nhận dữ liệu: Do raspberry gửi từng ký tự một nên mỗi khi nhận được một receiveEvent thì ta phải ghi thêm nó vào buffer, cho tới khi nhận được ký tự ‘\0’ thì bật cờ flag để báo hiệu đã nhận đủ chuỗi.  
   
3. Event handler cho nhận request: Khi nhận được một read request từ raspberry thì ghi thêm chuỗi vừa nhận vào cuối sendStr, rồi gửi. Sau đó bật cờ sendFlag báo hiệu đã gửi và reset buffer sendStr.  
   
4. Hàm loop(): Kiểm tra cờ receiveFlag, nếu true tức là đã nhận đủ chuỗi, in chuỗi đó ra serial monitor. Tiếp đó kiểm tra cờ sendFlag, nếu true tức là đã gửi phản hồi và reset receiveBuffer.  
   

Kết quả chạy chương trình: Khi raspberry gửi 1 chuỗi tới Arduino thì Arduino sẽ ghi chuỗi đó ra serial monitor đồng thời gửi lại chuỗi “Arduino received: ” + nội dung chuỗi vừa nhận được từ Pi.

