TH7.1

Để giao tiếp UART giữa Pi3 và PC thì chúng ta phải sử dụng 1 mạch chuyển USB to TTL là module chuyển đổi tín hiệu USB sang tín hiệu Serial tuần tự theo chuẩn TTL.

* Giải thích code

\*Các thư viện cần thiết :

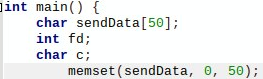


Ở đây bọn em sử dụng thư viện wiringSerial của wiringPi

Thư viện termios.h ở đây để phục vụ cho việc thay đổi option của hàm serialOpen (sẽ giải thích bên dưới)

\*Trong hàm main:

*- Khai báo biến và mảng:*



*- Mở và khởi tạo thiết bị nối tiếp và đặt tốc độ truyền (ở đây là 115200):*



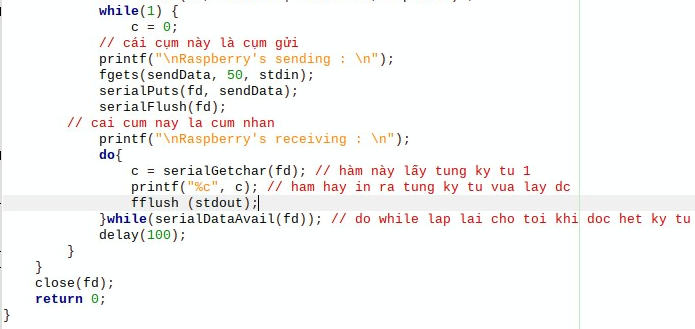
Hàm serialOpen() sẽ đặt cổng ở chế độ "raw" (input từng ký tự một và không thao tác gì khác với input và output). Đặt timeout là 10s, trả về file descriptor hoặc -1 nếu lỗi, trong trường hợp đó, lỗi sẽ được hiển thị.

*- Thay đổi cấu hình để chuyển sang canonical mode (truyền từng dòng) thay vì raw mode (truyền từng ký tự)*



Nếu không thay đổi option thì serialOpen sẽ giới hạn input buffer chỉ còn 8 byte. Sau khi chuyển sang canonical mode thì input buffer tăng lên thành 4096 byte.

*- Hàm while(1):*



* Đầu tiên là chức năng gửi, nhập từ bàn phím chuỗi ký tự muốn gửi, sau đó chuỗi sẽ được gửi vào mảng sendData nhờ hàm fgets().

fgets(<biến lưu>, <độ dài tối đa>, <nguồn>). Trong đó biến lưu là con trỏ string, tức là sendData như ở trên, nguồn là stdin nếu muốn đọc từ bàn phím nhập vào terminal.

Hàm fgets() lấy 1 dòng từ nguồn (ở đây dùng stdin tức là input vào terminal), với độ dài tối đa là n (cụ thể ở đây bọn em cho là 50).

Hàm serialPuts() để gửi 1 mảng hoặc 1 chuỗi ký tự đi qua cổng.

Hàm serialFlush() để chờ đến khi dữ liệu được gửi xong hoặc bỏ qua tất cả dữ liệu được gửi tới. Hàm này rất cần với uart, nếu không có sẽ không thể gửi chính xác được.

* Tiếp theo là chức năng nhận, ở trên máy tính dùng phần mềm Hercules, nhập input để gửi một hoặc một chuỗi ký tự tới Rasperry Pi3 thông qua cổng COM của USB - TTL.

Coi biến c là ký tự nhận được.

Hàm serialGetchar() để lấy từng ký tự một.

Khi nhập xâu thì kết thúc bằng một ký tự xuống dòng ‘\n’. Nhưng compiler vẫn nhận đây là một xâu cho tới bộ nhớ tạm thời và truyền nó vào biến xâu tiếp theo của chương trình.

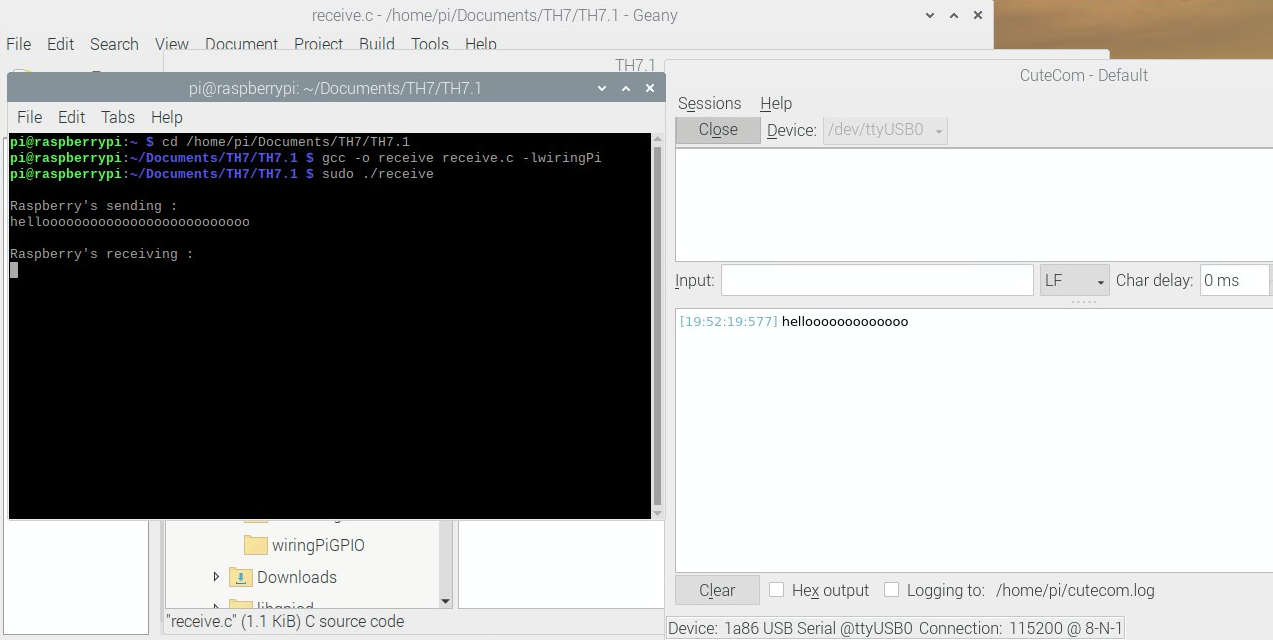
fflush(stdin); với mục đích xóa đi bộ nhớ tạm thời của chương trình để không tồn tại những giá trị rác như bên trên.

Hàm serialDataAvail() trả về số byte có thể đọc còn lại từ cổng serial. Hàm DO WHILE lặp lại tới khi hết ký tự đọc được ở cổng.

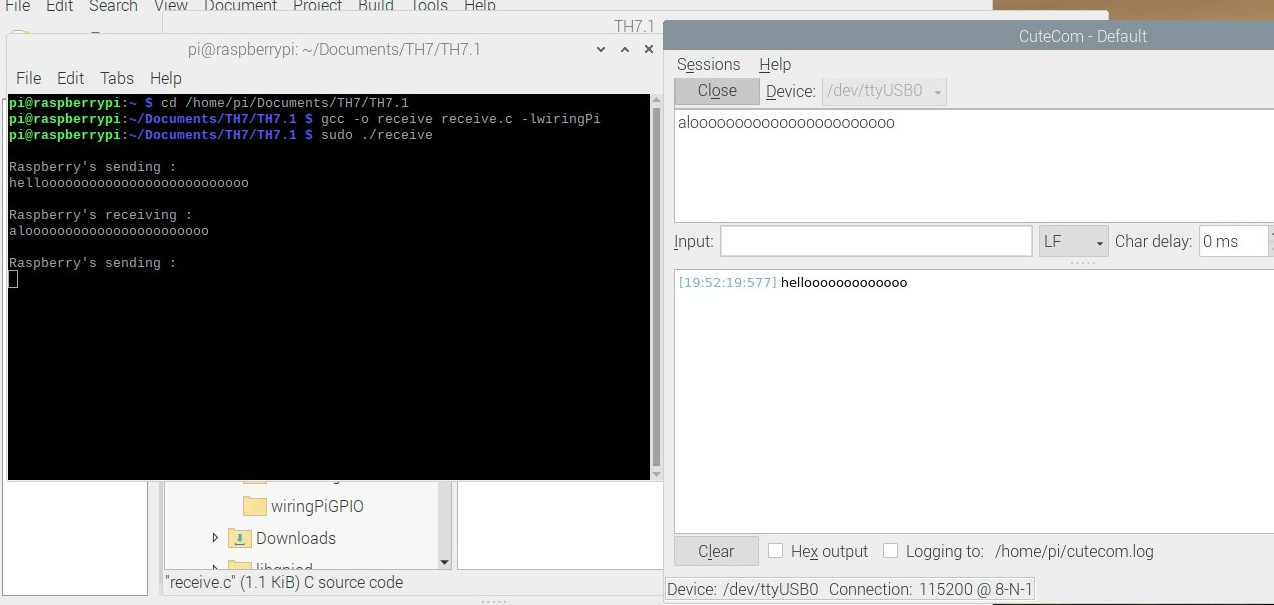
* Kết quả thực hiện

\*Khi gửi:

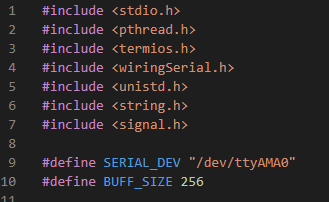
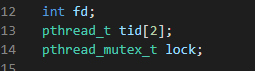
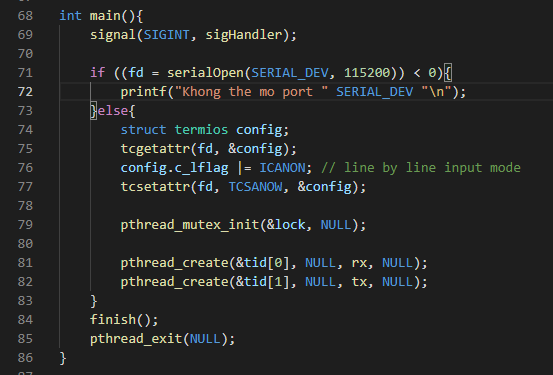
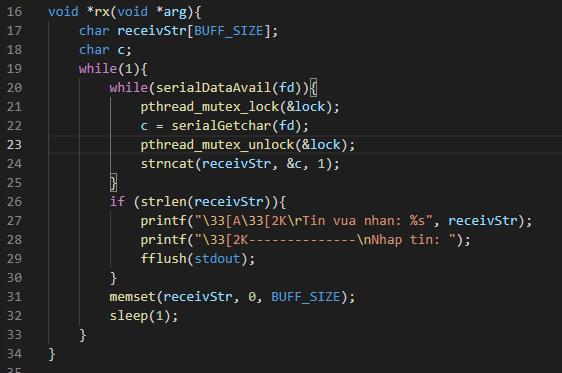
(bởi vì Raspberry Pi thực chất cũng là 1 máy tính, để chạy thử chương trình thì bọn em cắm luôn USB – TTL vào cổng USB của raspberry. Lúc này chương trình sẽ thực hiện giao tiếp giữa GPIO UART <–> USB-TTL <–> USB port. Nếu cắm usb TTL sang máy tính khác chương trình cũng vẫn hoạt động y như vậy).



\* Khi nhận:



TH7.2:

* Thư viện và define:  
  
* Các biến toàn cục bao gồm đặc tả file, mảng thread ID, mutex:  
  
* Hàm main(): Sử dụng SIGINT để kết thúc (ctr+C). Dùng hàm serialOpen của thư viện wiringSerial để khởi tạo port. Sau đó chuyển thành canonical mode để đọc input từng dòng. Cuối cùng là tạo 2 thread RX và TX.  
  
* Thread RX: luôn kiểm tra input buffer của UART, nếu có chuỗi được gửi đến thì ngay lập tức in ra chuỗi đó.  
  
* Thread TX: Nhập chuỗi từ bàn phím, sau đó gửi chuỗi đi  
  
* Hàm ngừng thread + đóng port và signal handler  
  