<http://leaqua.mulple.com/blog/%EC%95%84%EB%91%90%EC%9D%B4%EB%85%B8%EC%99%80-%EB%9D%BC%EC%A6%88%EB%B2%A0%EB%A6%AC%ED%8C%8C%EC%9D%B4%EC%9D%98-uartserial-%ED%86%B5%EC%8B%A0/>

## 아두이노와 라즈베리 파이의 UART(serial) 통신

라즈베리파이와  아두이노 사이의 통신에는 SPI, I2C , UART 등을 사용할수 있습니다.

그중에 제일 만만한? UART(Serial) 을 테스트 해보았습니다.

같은 전압의 IO핀을 사용하면 바로 결선해서 사용하면 간단해 지겠지만 아두이노의 입출력 핀 전압 5V 와 라즈베리파이의 GPIO 의 입출력 핀전압 3.3V 의 차이 때문에 직접적인 결선은 문제가 있어서 회로구성이 불가피 해집니다.

UART 통신하는 회로의 구성에도 여러가지가 있습니다.

[MOSFET 과 저항으로 구성되는 방법](http://www.fritz-hut.com/connecting-an-arduino-and-raspberry-pi/) 과 아두이노와 라즈베리파이 사이를 USB케이블로 연결하는 방법 그리고  높은 레벨의 Logic 신호를 낮은 레벨의 Logic으로 바꿔주는 Logic IC를 사용 하는 방법등이 있습니다.

USB케이블은 따로 사용해야 하므로 제외하고 가격면이나 회로구성이 조금이나마 단순해 지는 Logic IC를 사용 하겠습니다.

제가 사용한 Logic IC 는 국내쇼핑몰 등에서 구입하기 쉬운 74HC4050 입니다. 74HC4049 를 사용해도 무관 합니다.

### 라즈베리 파이의 UART 설정 수정

라즈베리 파이는 기본적으로 115200 의 uart로 콘솔에 접속 하도록 되어 있습니다. 따라서 아두이노와  minicom등 을 사용하여 시리얼에 바로 연결하게 되면 정상적으로 작동하지 않습니다.

/boot/cmdline.txt  파일과  /etc/inittab 을 수정하여 콘솔역할을 제거하여 시리얼 연결로만 작동하도록 수정해줄 필요가 있습니다.

우선 cmdline.txt 를 수정합니다.

Shell



|  |  |
| --- | --- |
| 1 | $ sudo nano /boot/cmdline.txt |

다음 내용을 찾아서 붉은 글씨 부분을 제거 합니다.

dwc\_otg.lpm\_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

/boot/cmdline.txt



|  |  |
| --- | --- |
| 1 | dwc\_otg.lpm\_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait |

다음으로 inittab 을 수정합니다.

Shell



|  |  |
| --- | --- |
| 1 | $ sudo nano /etc/inittab |

T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100 를 찾아서 주석처리를 해주면 됩니다.

/boot/cmdline.txt

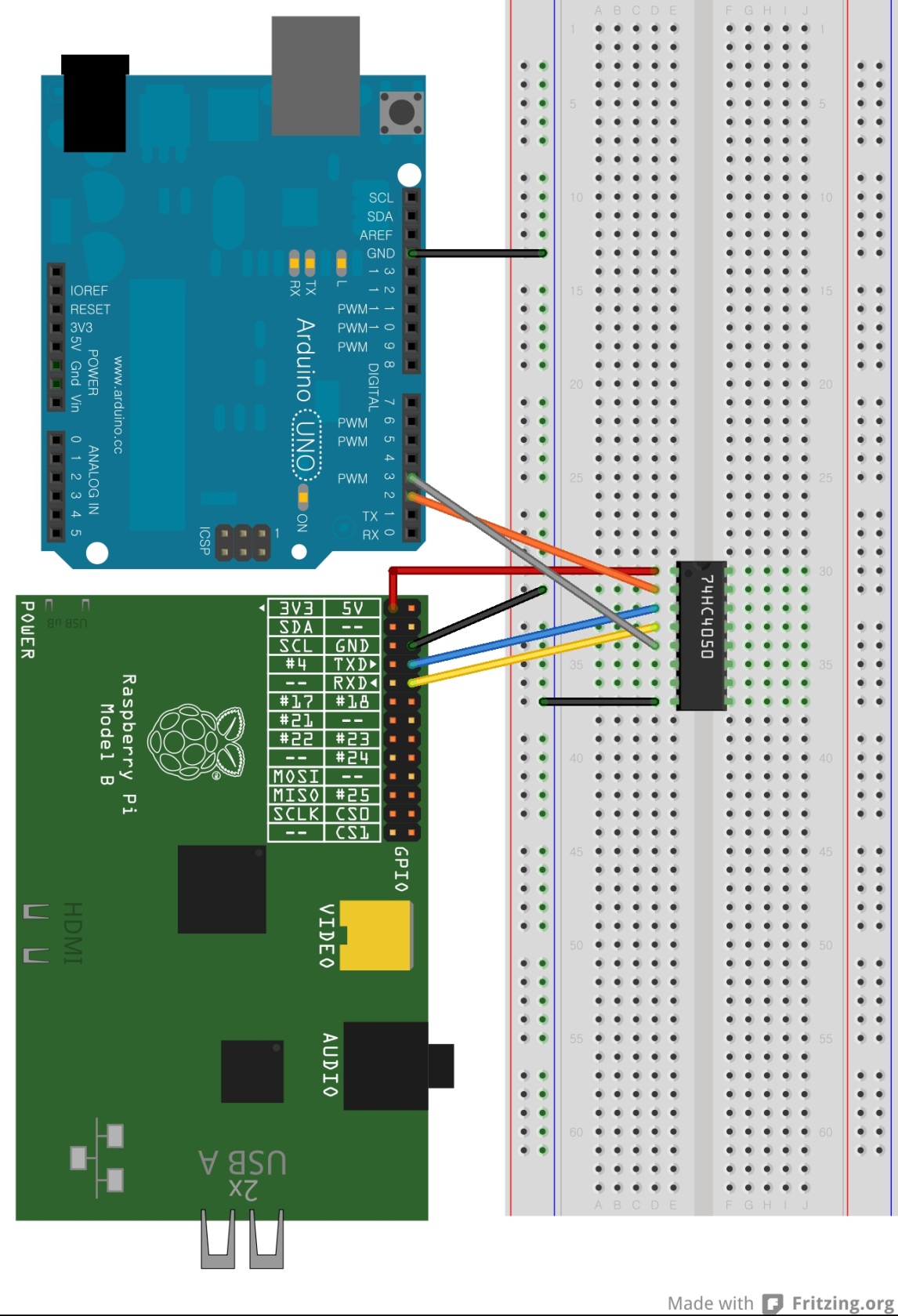


|  |  |
| --- | --- |
| 1 | #T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100 |

수정 후 리부팅을 해주면 콘솔역할을 막아주는 작업은끝이 납니다.

### 회로구성

회로의 구성은 아래 그림과 같습니다. 그림을 클릭해서 크게 보거나 그림 아래 첨부된 PDF를 보거나 아무거나 OK 입니다.

[](http://leaqua.mulple.com/wp-content/uploads/2013/02/74HC4050_bb.jpg)

회로도 PDF 파일 다운 :  [74HC4050\_schem](http://leaqua.mulple.com/wp-content/uploads/2013/02/74HC4050_schem.pdf)

브레드 보드 구성 PDF 파일 다운 : [74HC4050\_bb](http://leaqua.mulple.com/wp-content/uploads/2013/02/74HC4050_bb.pdf)

4050 IC의 2번핀 과 5번핀을 아두이노의 TX, RX 핀이 아닌 D2 , D3로 연결한 것은 테스트시 PC와 시리얼  통신을 하기 위해서 이고 4050 IC와의 통신은 SoftwareSerial 을 통해서 이루어 지게 됩니다.

**빵두이노에 연결**

### 아두이노 코드

Arduino Sketch



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | /\*  아두이노와 라즈베리 파이의 UART 연결  아두이노 : SoftSerial  라즈베리파이 : GPIO UART  \*/    #include    SoftwareSerial mySerial(2, 3); // RX, TX    void setup()  {    Serial.begin(9600);    Serial.println("Start UART test");  // PC의 시리얼 모니터에 표시합니다.    mySerial.begin(9600); //소프트 시리얼로 라즈베리 파이와 통신하게 됩니다.  }    void loop() {    // 라즈베리 파이에서 넘어온 데이터가 있으면 PC로 넘겨줍니다.    if (mySerial.available())      Serial.write(mySerial.read());    // PC에서 넘어온 데이터가 있으면 라즈베리파이로 넘겨줍니다.    if (Serial.available())      mySerial.write(Serial.read());  } |

위의 스케치를 업로드 합니다.

### minicom 설치 / 작동 테스트

시리얼 통신에 많이 사용하는 프로그램인 minicom 으로 작동을 테스트 해봅니다.

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install minicom |

위와 같이 설치하고

Shell

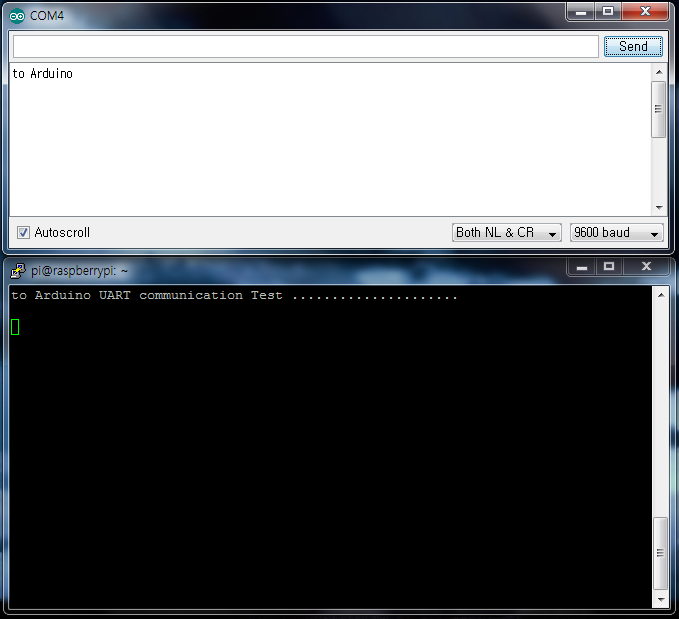


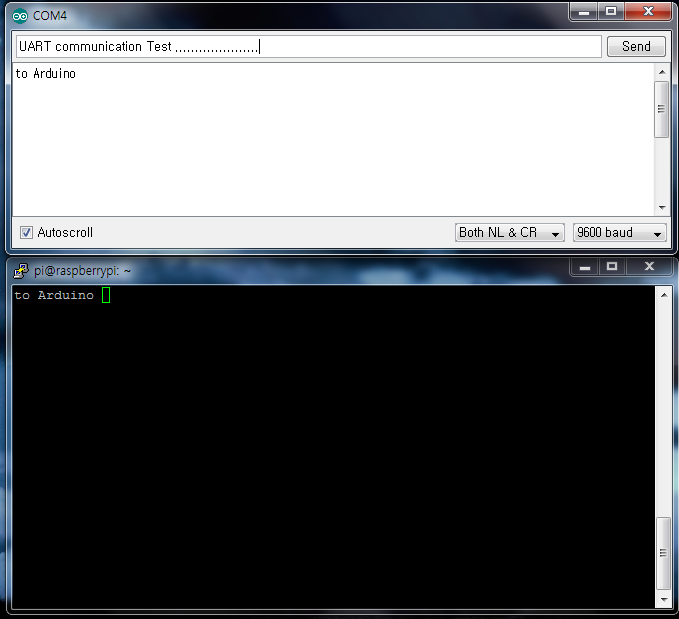
|  |  |
| --- | --- |
| 1 | $ minicom -b 9600 -o -D /dev/ttyAMA0 |

으로 실행합니다.

실행중에 종료는 ctrl-a 누르고 x 누르면 됩니다.

아래는 아두이노 IDE 와 라즈베리파이에서 실행중인 minicom 의 화면입니다. 양쪽 모두 Text 를 주고 받기가 정상적으로 이루어 집니다.

[](http://leaqua.mulple.com/wp-content/uploads/2013/02/minicom2.png)

[](http://leaqua.mulple.com/wp-content/uploads/2013/02/minicom3.png)

### 라즈베리 파이에 Python 용 Serial 라이브러리 설치

GCC 나 minicom 을 사용할 경우엔 필요가 없는 과정입니다. Python 으로 아두이노와 통신할 경우 필요합니다.

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install python-serial |

와 같이 설치 합니다.

패키지 설치를 원하지 않는경우엔 <http://pyserial.sourceforge.net/> 에서 최신 버전들 다운 받아서 라즈베리파이의 계정에 복사한뒤 쉘에서 다음을 진행합니다.

Shell



|  |  |
| --- | --- |
| 1  2  3 | $ tar zxf pyserial-2.6  $ cd pyserial-2.6  $ sudo python setup.py install |

설치가 정상적으로 이루어 지고 테스트에 들어갑니다.

Shell

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $ sudo python    >>> import serial  >>> ser = serial.Serial("/dev/ttyAMA0",9600)  # 시리얼 포트를 열고  >>> print ser.portstr       # 시리얼 포트가 정상적으로 열렸는지 확인  >>> ser.write("hello")      # 글자를 써 봅니다.  >>> ser.close()             # 시리얼 포트 닫고.. |

실행 화면입니다. 잘되는군요 

### [python](http://leaqua.mulple.com/wp-content/uploads/2013/02/python.png)

이상으로 Arduino 와 Raspberry pi 의 UART 통신 관련 포스트를 마칩니다.