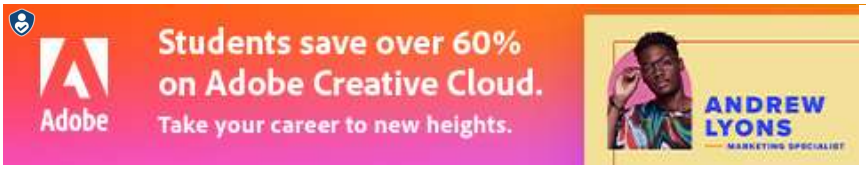


ESP8266 Serial Communication

cember 16, 2016 IoT Tutorials ESP8266, Internet of Things, iot

SEMRUSH

l interface is common requirement for most of the application development. We are discussing w to do 3.3V to 5V level conversion for converting serial TTL to RS232 level from 3.3V you can use 232 it operates at 3.3V levels.

P8266 we have one hardware serial i.e. GPIO2 (Tx) and GPIO3 (Rx).

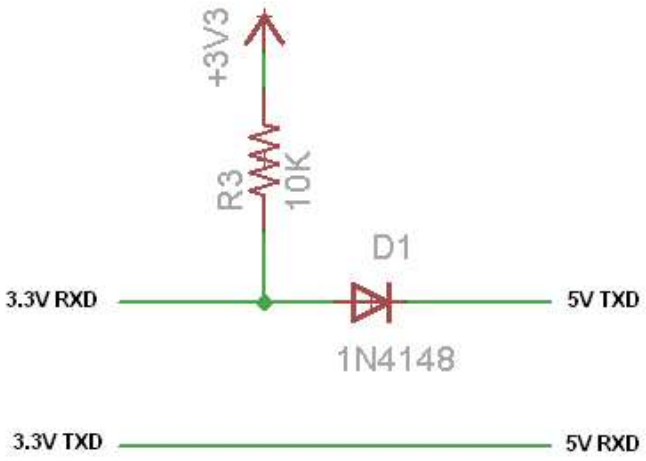
Hardware Serial Programming is similar to the Arduino Serial. Remember that few USB to Serial converter does not support higher baud rate. It is better to keep baud rate below 115200.

Serial interface is useful for debugging the programs by sending some debug info to serial.

Hardware Serial Communication

Level conversion 3.3V to 5V

For Level conversion from 3.3V to 5V we need only two components.



Logic Level Conversion

Assuming that you have already connected serial with your USB to Serial converter or You are using ESP Witty, Node MCU.

Serial communication on pins TX/RX uses TTL logic levels 3.3V. Don't connect these pins directly to S232 serial port; they operate at +/- 12V and can damage your ESP8266 board.

You can use the Arduino IDE environment's built-in serial monitor to communicate with an ESP board. Click the Tools>>Serial monitor button in the toolbar and select the same baud rate used in the call to begin().

Before we start our program lets understand commonly used serial commands

ial.begin(9600);

ommand is used to initialize serial port with 9600 baud rate. It is used only when you initialize i.e. in **Setup()** or when you want to change the baud rate. Baud rate is number of bits transmit- er second. Higher the baud rate higher the speed of communication.

ower baud rate if cable length is more. Standard baud rates are 300, 1200, 2400, 4800, 9600, , 38400, 57600, 115200, 230400, 250000.

```
void setup() {
    Serial.begin(9600); // opens serial port, sets data rate to 9600
bps
}

void loop() {
}
```

Serial.print(); and Serial.println();

Serial.print prints text on same line. Seial.println puts carriage return at the end of line.

Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. For example:

Serial.print(65) gives “65”

Serial.print(char(65)) gives “A”, 65 is ASCII code of A

Serial.print(1.23456) gives “1.23”

Serial.print(‘N’) gives “N”

Serial.print(“Hello world.”) gives “Hello world.”

Serial.print(78, BIN) gives "1001110"

Serial.print(78, OCT) gives "116"

Serial.print(78, DEC) gives "78"

.print(78, HEX) gives "4E"

.println(1.23456, 0) gives "1"

.println(1.23456, 2) gives "1.23"

.println(1.23456, 4) gives "1.2346"

.print("\t"); // prints a tab

Serial.print("\n"); // inserts line break

Serial.write();

Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the **print()** function instead.

Serial.print(65); gives "65"

Serial.print(char(65)); gives "A", 65 is ASCII code of A

Serial.write(65); gives "A", 65 is ASCII code of A

Serial.available();

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).

Serial.read();

Reads incoming serial data as single byte.

Hardware Serial Program for ESP8266

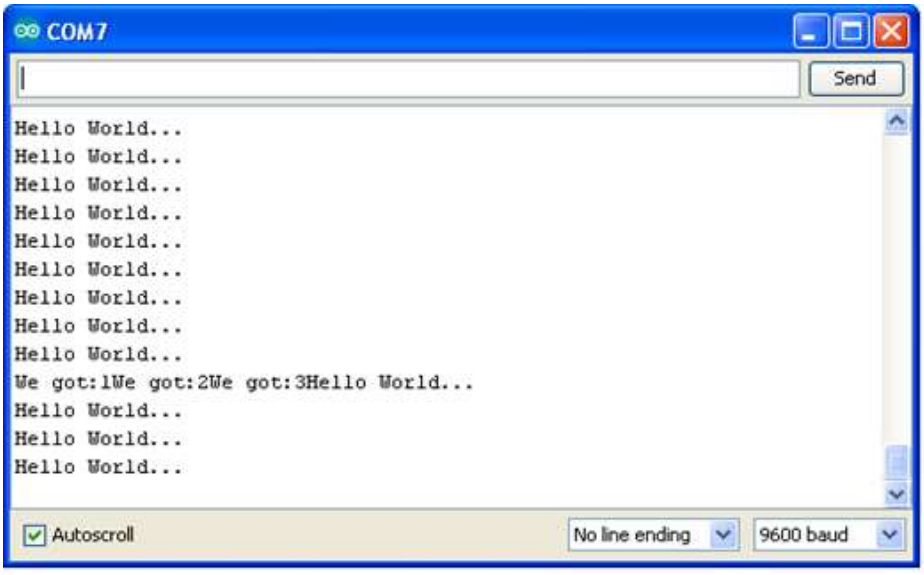
Programming of ESP is same as arduino.

```
/*
  ESP8266 Serial Communication
```

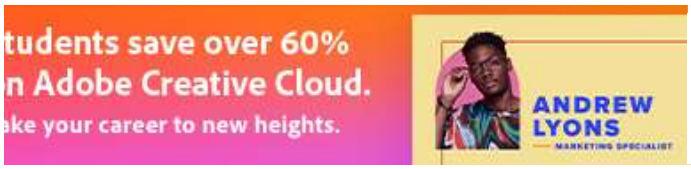

```
*/

void setup() {
  Serial.begin(9600);     // Initialize the Serial interface with
baud rate of 9600
}

// the loop function runs over and over again forever
void loop() {
 if(Serial.available()>0)    //Checks is there any data in buffer
 {
   Serial.print("We got:");
   Serial.print(char(Serial.read()));  //Read serial data byte and
send back to serial monitor
 }
 else
 {
   Serial.println("Hello World..."); //Print Hello word every one
second
   delay(1000);                      // Wait for a second
 }
}
```

Results

Open serial monitor and observe the data we get “Hello World…” every second and when we send any data it shows “We got:data”. as shown in below figure.

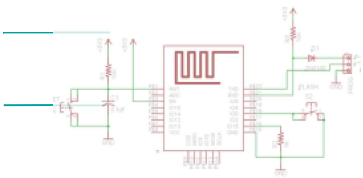


ESP8266 Serial Communication



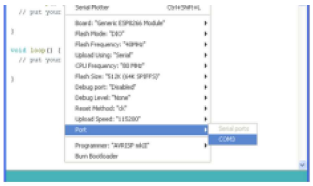
Khắc chính xác tại mọi điểm

Related



IO Level Conversion 3.3V to 5V for ESP8266
December 14, 2016
In "IoT Tutorials"

ESP8266 arduino digital IO
February 20, 2018
In "ESP8266"



ESP8266 LED Blink
December 16, 2016
In "IoT Tutorials"