Wifi communication between ESP8266 and PC using TCP/IP or UDP/broadcast protocol

Code for wifi communication between ESP8266 wifi module and PC using either TCP/IP protocol or UDP using the broadcast IP address. The code peforms an *echo* from/to the Serial port to/from the Wifi. The ESP8266 acts as a server, at which the PC connects, and echos the incoming data in Serial to the PC over wifi. For TCP/IP protocol

> MCU ----(Serial)---> ESP8266 (server) ----(TPC/IP)---> PC (client)

while for UDP/broadcast multiple clients can connect to the ESP8266

> MCU ----(Serial)---> EPS8266 (server) ----(UDP/broadcast)-->PC1 (client)
>
> -->PC2 (client)
>
> -->PCn (clinet)

For the TCP/IP protocol, the communication may be also bidirectional: PC sends also data to the ESP8266, which echos to the Serial

> MCU <---(Serial)--->ESP8266 (server) <---(TPC/IP)---> PC (client)

Bidirectional communication is not available for the UDP/broadcast protocol.

The ESP8266 can either act in AP mode or in STA mode:

- AP mode: ESP8266 acts as a router which generates the Wifi and PC connects directly to ESP8266
- STA mode: both ESP8266 and PC are connected to the same Wifi, which is generated by another router.

See also This for code tips.

Building

Use ArduinoIDE to build and upload the code to the ESP8266, by using board *Generic ESP8266 Board* from EPS8266 Boards (get it from Here). Connections for programming ESP8266 module using Arduino (e.g. Uno) are:

- supply on GND and 3V3 (use 3.3 volts!)
- RST of Arduino to GND during all uploading (to bypass arduino)

- RST of ESP8266 on ground before uploading for hard resetting
- RX to Arduino RX (suggest using voltage divider to have 3.3V level)
- TX to Arduino TX
- GPIO_0 of ESP8266 on ground during uploading

Configurations

At the beginning of the code there are few defines for user configuration:

- `MODE_AP` : define this to enable AP mode
- `MODE_STA` : define this to enable STA mode
- `MODE_TCP` : define this to use TCP/IP mode
- `MODE_UDP` : define this to use UDP mode
- `BIDIRECTIONAL` : define this to enable the bidirectional communication
- `DEBUG` : define this for debugging message over serial port.

Other (advanced) configurations include:

```
#define UART_BAUD 115200 // UART baudrate. Must be consisted with that used in the
Teensy
#define IP 192, 168, 0, 1 //IP address of ESP. ONLY WITH MODE_AP. With MODE_STA the
router assigns the IP
#define MASK 255, 255, 255, 0 //Netmask of ESP. ONLY WITH MODE_AP. Not used with
MODE_STA.
#define PORT 9876 //Port of the service. Must be consistent with that used in the PC
#define CHANNEL 2 //Channel of ESP. ONLY WITH MODE_AP. Not used with MODE_STA.
#define MAX_CONN 1 //Max number of simulatous connection. ONLY WITH MODE_AP. Not used
with MODE_STA.
#define PWR_LEVEL 20.5 // Wifi power. Max is 20.5. ONLY WITH MODE_AP. Not used with
MODE_STA.
#define BUF_SIZE 256 // Buffer size (maximum bytes read at a time)
#define TIMEOUT 1000 // Timeout after which the buffer read is sent (us)
```

Usage

The code peforms an *echo* from/to the Serial port to/from the Wifi. However, before sending the desired data from the microcontroller to the ESP8266 via the Serial port, it is necessary to set the wifi name and password. Depending on the selected mode, these are:

- AP mode: name and password of the wifi generated by the ESP8266. PC must make use of the name and password specified for connection to the ESP8266

- STA mode: name and password of the wifi at which the ESP8266 connects.

To set the name and password correctly, you must send over the Serial port the name and password with the following format:

```
name:password\n
```

After sending the desired name and password, you can just use the Serial port as you want from the microcontroller, e.g. `Serial.println("Hello World!");`.