@mainpage

WeeESP8266

An ESP8266 library for Arduino providing an easy-to-use way to manipulate ESP8266.

Source

Source can be download at https://github.com/itead/ITEADLIB_Arduino_WeeESP8266.

You can clone it by:

```
git clone https://github.com/itead/ITEADLIB_Arduino_WeeESP8266.git
```

Documentation

Online API documentation can be reached at http://docs.iteadstudio.com/ITEADLIB_Arduino_WeeESP8266/index.html.

Offline API documentation can be found under directory
doc.

How to get started

On the home page of API documentation, the tabs of Examples, Classes and Modules
will be useful for Arduino lovers.

API List

```
bool    kick (void) : Verify ESP8266 whether live or not.

bool    restart (void) : Restart ESP8266 by "AT+RST".

String  getVersion (void) : Get the version of AT Command Set.

bool    setOprToStation (void) : Set operation mode to staion.

bool    setOprToSoftAP (void) : Set operation mode to softap.

bool    setOprToStationSoftAP (void) : Set operation mode to station + softap.

String  getAPList (void) : Search available AP list and return it.

bool    joinAP (String ssid, String pwd) : Join in AP.

bool    leaveAP (void) : Leave AP joined before.

bool    setSoftAPParam (String ssid, String pwd, uint8_t chl=7, uint8_t ecn=4) : Set SoftAP
parameters.

String  getJoinedDeviceIP (void) : Get the IP list of devices connected to SoftAP.

String  getIPStatus (void) : Get the current status of connection(UDP and TCP).

String  getLocalIP (void) : Get the IP address of ESP8266.
```

```
bool    enableMUX (void) : Enable IP MUX(multiple connection mode).

bool    disableMUX (void) : Disable IP MUX(single connection mode).

bool    createTCP (String addr, uint32_t port) : Create TCP connection in single mode.

bool    releaseTCP (void) : Release TCP connection in single mode.

bool    registerUDP (String addr, uint32_t port) : Register UDP port number in single mode.

bool    unregisterUDP (void) : Unregister UDP port number in single mode.

bool    createTCP (uint8_t mux_id, String addr, uint32_t port) : Create TCP connection in multiple
mode.

bool    releaseTCP (uint8_t mux_id) : Release TCP connection in multiple mode.

bool    registerUDP (uint8_t mux_id, String addr, uint32_t port) : Register UDP port number in
multiple mode.

bool    unregisterUDP (uint8_t mux_id) : Unregister UDP port number in multiple mode.

bool    setTCPServerTimeout (uint32_t timeout=180) : Set the timeout of TCP Server.

bool    startServer (uint32_t port=333) :  Start Server(Only in multiple mode).

bool    stopServer (void) : Stop Server(Only in multiple mode).

bool    startTCPServer (uint32_t port=333) : Start TCP Server(Only in multiple mode).

bool    stopTCPServer (void) : Stop TCP Server(Only in multiple mode).

bool    send (const uint8_t *buffer, uint32_t len) : Send data based on TCP or UDP builded already
in single mode.

bool    send (uint8_t mux_id, const uint8_t *buffer, uint32_t len) : Send data based on one of TCP
or UDP builded already in multiple mode.

uint32_t    recv (uint8_t *buffer, uint32_t buffer_size, uint32_t timeout=1000) : Receive data from
TCP or UDP builded already in single mode.

uint32_t    recv (uint8_t mux_id, uint8_t *buffer, uint32_t buffer_size, uint32_t timeout=1000) :
Receive data from one of TCP or UDP builded already in multiple mode.

uint32_t    recv (uint8_t *coming_mux_id, uint8_t *buffer, uint32_t buffer_size, uint32_t
timeout=1000) : Receive data from all of TCP or UDP builded already in multiple mode.
```

Mainboard Requires

- RAM: not less than 2KBytes
- Serial: one serial (HardwareSerial or SoftwareSerial) at least

Suppported Mainboards

- Arduino UNO and its derivatives
- Arduino MEGA and its derivatives
- Iteaduino UNO

- WBoard Pro

Using SoftwareSerial

If you want to use SoftwareSerial to communicate with ESP8266, you need to modify the line in file `ESP8266.h`:

```
//#define ESP8266_USE_SOFTWARE_SERIAL
```

After modification, it should be:

```
#define ESP8266_USE_SOFTWARE_SERIAL
```

Hardware Connection

WeeESP8266 library only needs an uart for hardware connection. All communications are done via uart. In each example, you must specify the uart used by mainboard to communicate with ESP8266 flashed AT firmware.

MEGA and WBoard Pro

For MEGA and WBoard Pro, `Serial1` will be used if you create an object (named wifi) of class ESP8266 in your code like this:

```
#include "ESP8266.h"
ESP8266 wifi(Serial1);
```

The connection should be like these:

```
ESP8266_TX->RX1(D19)
ESP8266_RX->TX1(D18)
ESP8266_CH_PD->3.3V
ESP8266_VCC->3.3V
ESP8266_GND->GND
```

UNO

To use SoftwareSerial, `mySerial` will be used if you create an object (named wifi) of class ESP8266 in your code like this:

```
#include "ESP8266.h"
#include <SoftwareSerial.h>

SoftwareSerial mySerial(3, 2); /* RX:D3, TX:D2 */
ESP8266 wifi(mySerial);
```

The connection should be like these:

```
ESP8266_TX->RX(D3)
ESP8266_RX->TX(D2)
ESP8266_CH_PD->3.3V
ESP8266_VCC->3.3V
ESP8266_GND->GND
```

Attention

The size of data from ESP8266 is too big for arduino sometimes, so the library can't receive the whole buffer because the size of the hardware serial buffer which is defined in HardwareSerial.h is too small.

Open the file from `\arduino\hardware\arduino\avr\cores\arduino\HardwareSerial.h`.
See the follow line in the HardwareSerial.h file.

```
#define SERIAL_BUFFER_SIZE 64
```

The default size of the buffer is 64. Change it into a bigger number, like 256 or more.

The End!
