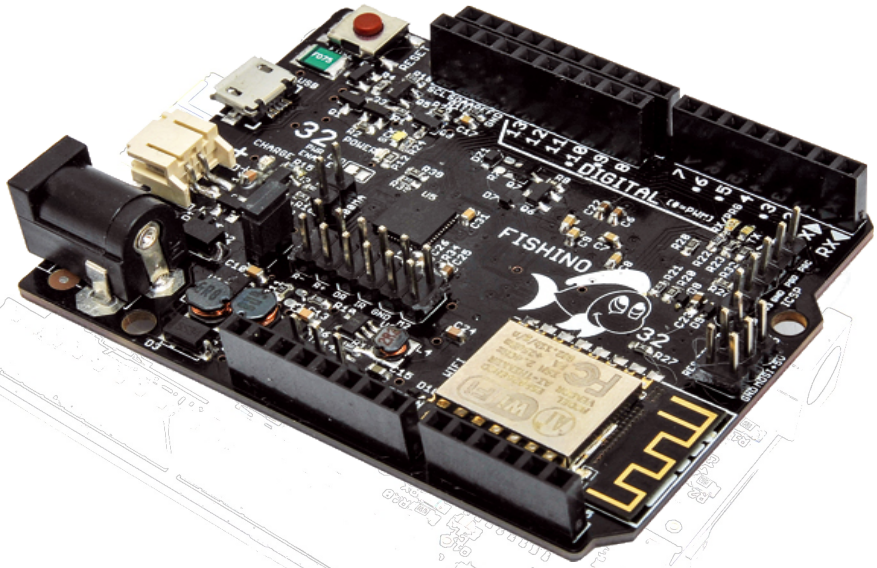
Fishino Library



Reference Manual

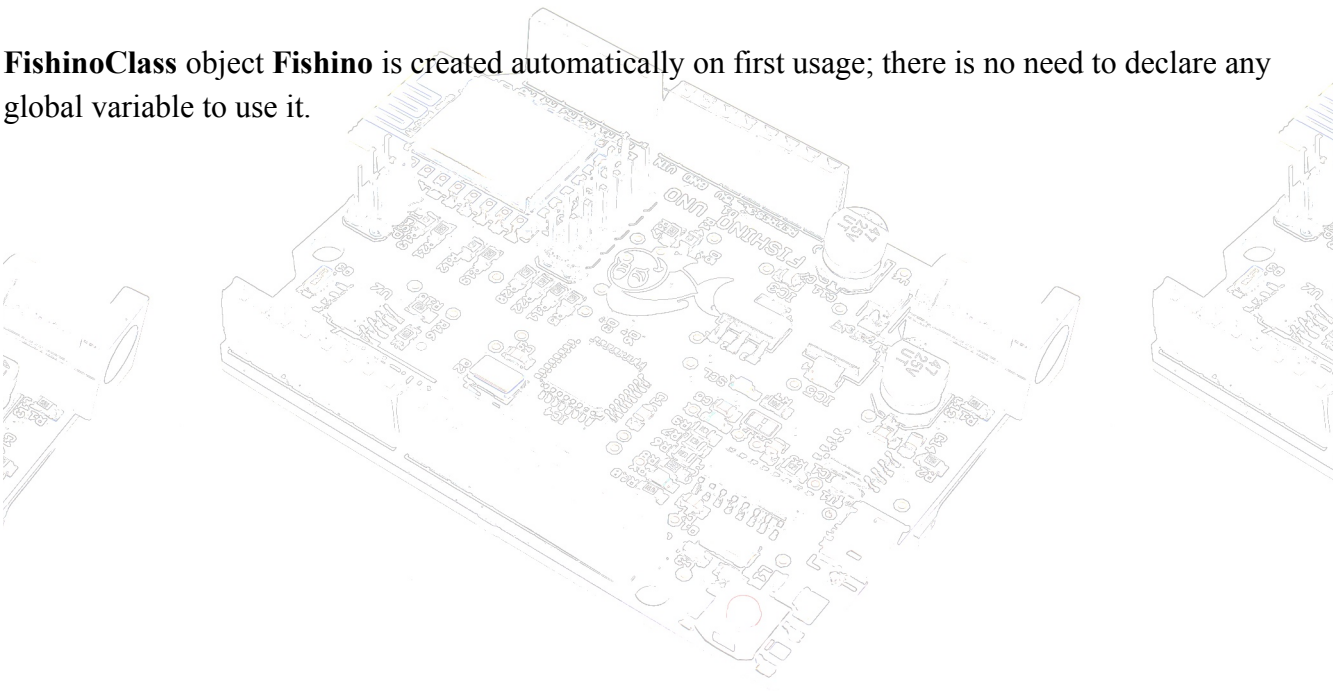
Class FishinoClass

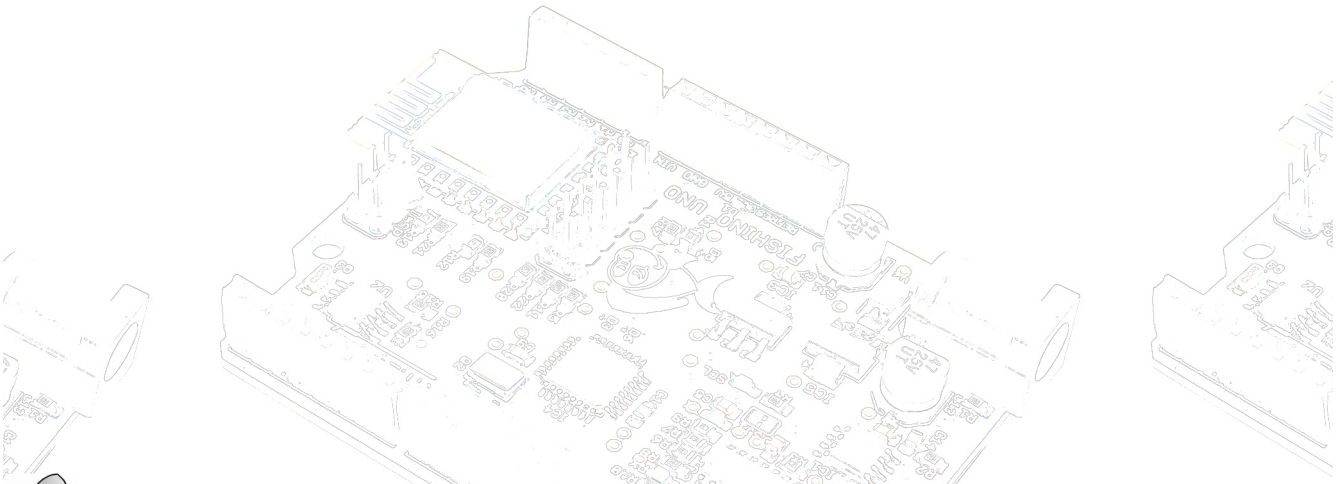
Class **FishinoClass** is the main interface of **Fishino Library**.

It has just **ONE** instance, called **Fishino**, which is a bridge between user code and the WiFi module.

Only **public members and enums** are documented here; the private ones are used by **FishinoClient**, **FishinoSecureClient**, **FishinoServer** and **FishinoSerial** classes which are documented in separate sections.

FishinoClass object **Fishino** is created automatically on first usage; there is no need to declare any global variable to use it.





analogRead

```
uint16_t FishinoClass::analogRead(void);
```

Read the value on WiFi module's analog input pin.

Parameters :

none

Return value :

The voltage value on analog input, with 10 bit resolution (from 0 for a 0 Volt input up to 1023 for about 3.3 Volt input)

Notes :

WARNING : the WiFi module operates on 3.3 Volt logic. DON'T apply a voltage greater than 3.3 Volt on its inputs, otherwise you could damage it!

Example :

```
uint16_t val = Fishino.analogRead();
double dVal = (double)val * 3.3 / 1024;
Serial.print("The voltage on analog pin is : ");
Serial.print(dVal)
Serial.println(" Volt");
```

Value corresponding to analog input pin is put on 'val' variable, scaled to input voltage and print on serial port

See also :

digitalRead

APINFO

```
struct FishinoClass::APINFO;
```

APINFO struct is used to retrieve information about the WiFi connection to an access point.

```
typedef struct
{
      uint8_t authmode;       // see AUTH_MODE enum
      char *ssid;             // SSID of joined AP
      int8_t rssi;            // wifi signal strenght
      uint8_t bssid[6];       // MAC address of joined AP
      uint8_t channel;        // channel of joined AP
} APINFO;
```

Notes :

Used internally

See also :

scanNetworks

AUTH_MODE

enum FishinoClass::AUTH_MODE;

List of WiFi connection authorization modes :

```
typedef enum
{
	AUTH_OPEN,
	AUTH_WEP,
	AUTH_WPA_PSK,
	AUTH_WPA2_PSK,
	AUTH_WPA_WPA2_PSK,
	AUTH_MAX
} AUTH_MODE;
```

Notes :

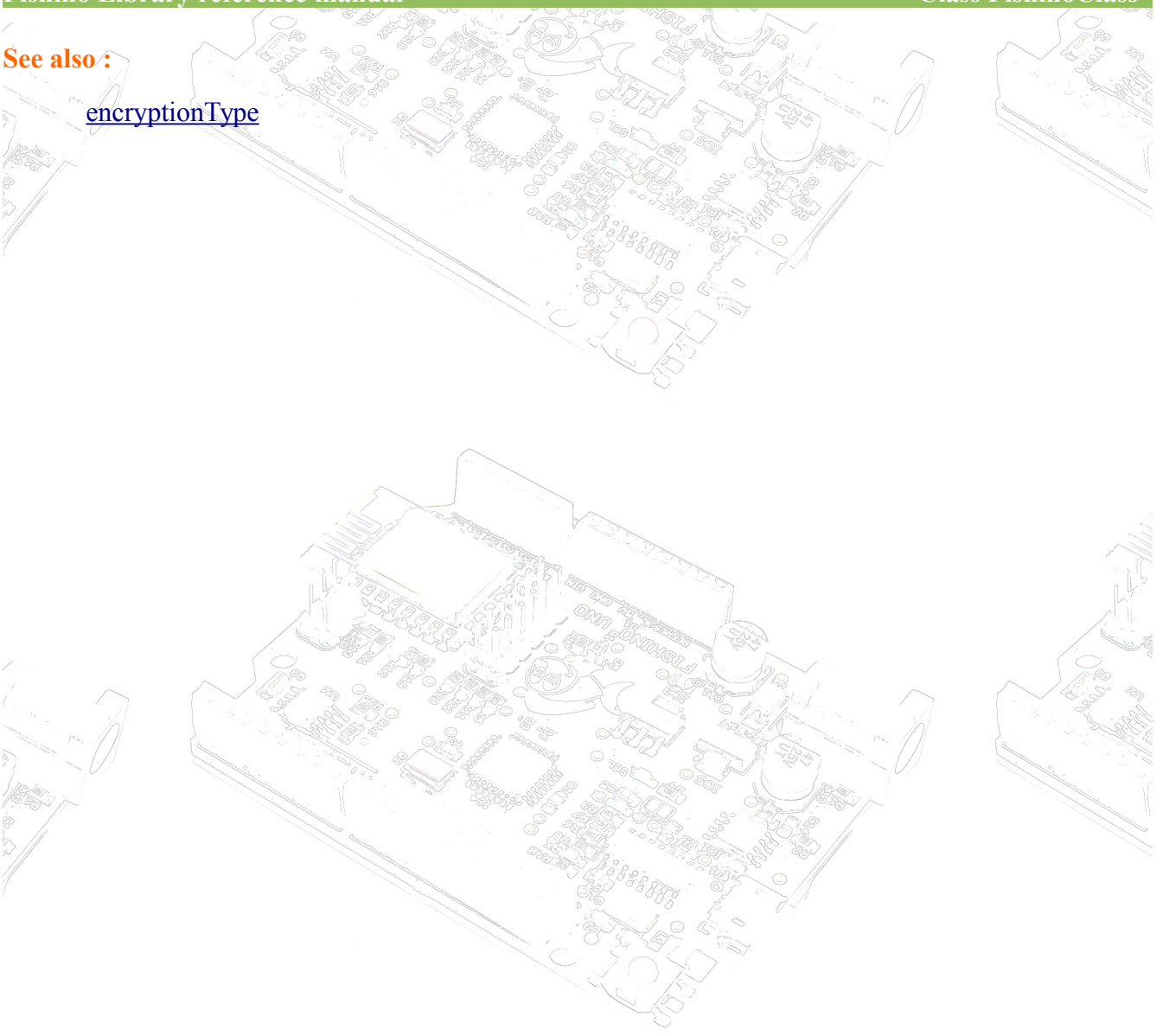
Example :

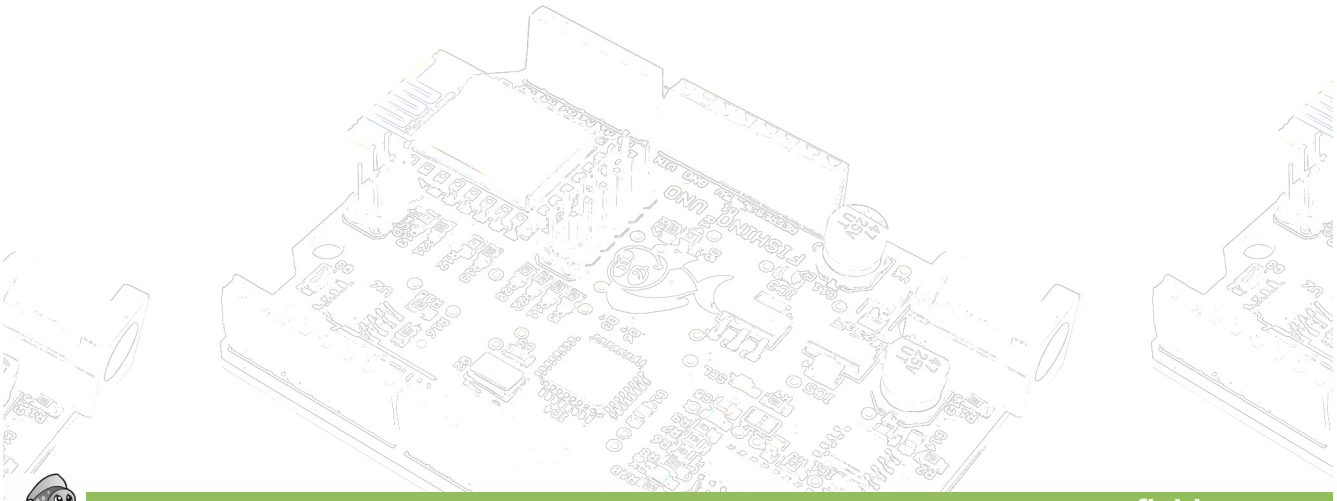
```
uint8_t encType = Fishino.encryptionType();
Serial.print("Auth type:");
switch(encType)
{
	case AUTH_OPEN:
		Serial.println("OPEN");
		break;
	case AUTH_WEP:
		Serial.println("WEP");
		break;
	case AUTH_WPA_PSK:
		Serial.println("WPA_PSK");
		break;
	case AUTH_WPA2_PSK:
		Serial.println("WPA2_PSK");
		break;
	case AUTH_WPA_WPA2_PSK:
		Serial.println("WPA_WPA2_PSK");
		break;
	default:
		Serial.println("UNKNOWN");
		break;
}
```

The example prints the authentication type of current connection.

See also :

encryptionType





begin

```
uint8_t FishinoClass::begin(const char *ssid);
uint8_t FishinoClass::begin(const __FlashStringHelper *ssid);
uint8_t FishinoClass::begin(const char* ssid, const char *passphrase);
uint8_t FishinoClass::begin(const __FlashStringHelper *ssid, const __FlashStringHelper *passphrase);
```

Start Wifi connection for OPEN or protected networks

Parameters :

- **ssid** Pointer to the SSID string.
- **passphrase** Pointer to password string.

Both strings can be normal or PROGMEM ones.

Return value :

return **true** on success, **false** otherwise

Notes :

Example :

```
bool res = Fishino.begin("MY_SSID", "MY_PASS");
```

Connects to AP with SSID “MY_SSID” and password “MY_PASS”

See also :

joinAp, quitAp, disconnect, setStaConfig, getStaConfig

BSSID

```
const uint8_t* FishinoClass::BSSID(void);
```

Return the current BSSID associated with the network.
It is the MAC address of the Access Point.

Parameters :

none

Return value :

pointer to uint8_t array with length WL_MAC_ADDR_LENGTH

Notes :

The returned array is static and get overwritten on each call.

Example :

```
uint8_t const *mac = Fishino.BSSID();
Serial.print("MAC ");
for(int i = 0; i < WL_MAC_ADDR_LENGTH; i++)
{
      Serial.print(":");
      Serial.print(mac[i], HEX);
}
Serial.println();
```

Prints AP's MAC address.

See also :

SSID, RSSI, encryptionType

clearLastError

```
void FishinoClass::clearLastError(void);
```

Clears last error code from WiFi module.

Parameters :

none

Return value :

none

Notes :

Last error code and string can be retrieved with getLastError() and getLastErrorString() functions

Example :

```
uint16_t errCode = Fishino.getLastError();
Fishino.clearLastError();
Serial.print("Last error code is :");
Serial.println(errCode);
```

The example reads and prints the last error code, then clears it.

See also :

getLastError, getLastErrorString, getErrorString, ErrorCodes, showErrors

config

```
bool FishinoClass::config(IPAddress local_ip);
bool FishinoClass::config(IPAddress local_ip, IPAddress gateway);
bool FishinoClass::config(IPAddress local_ip, IPAddress gateway, IPAddress subnet);
bool FishinoClass::config(IPAddress local_ip, IPAddress gateway, IPAddress subnet, IPAddress
dns_server);
```

Configure network parameters in station mode.

Parameters :

- **local_ip** the static IP address
- **gateway** the network gateway (usually router's address)
- **subnet** network address mask (example 255.255.255.0)
- **dns_server** the IP address of DNS server

Return value :

Function returns **true** on success, **false** otherwise

Notes :

The first version, with just local_ip parameter, assumes that you're on a "standard" network configuration, with a netmask of 255.255.255.0 and a gateway on xxx.xxx.xxx.1 address. The DNS is set to a default address.
If your network is not of this kind please use the more complete calls using all parameters; the DNS is always optional.

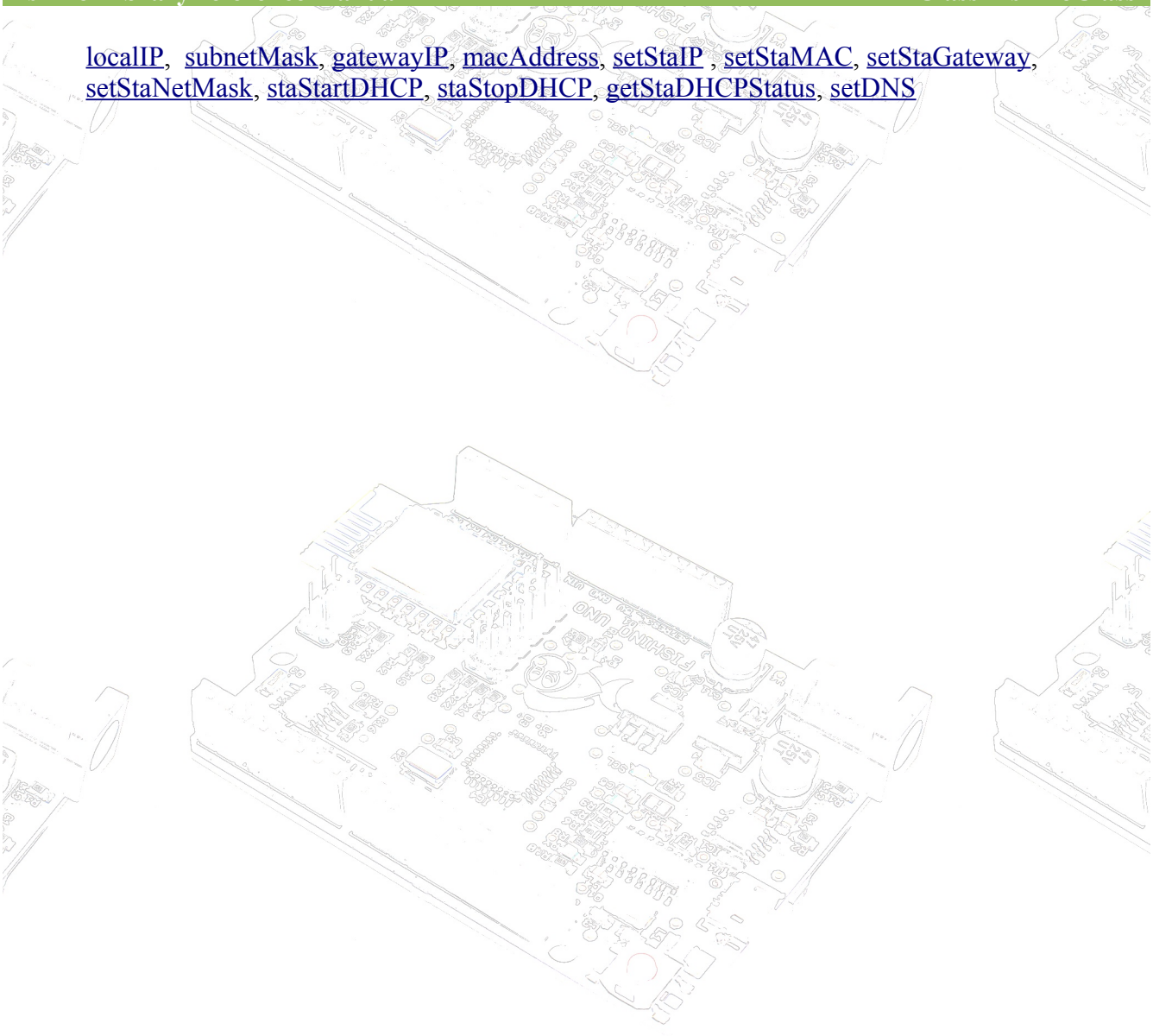
Example :

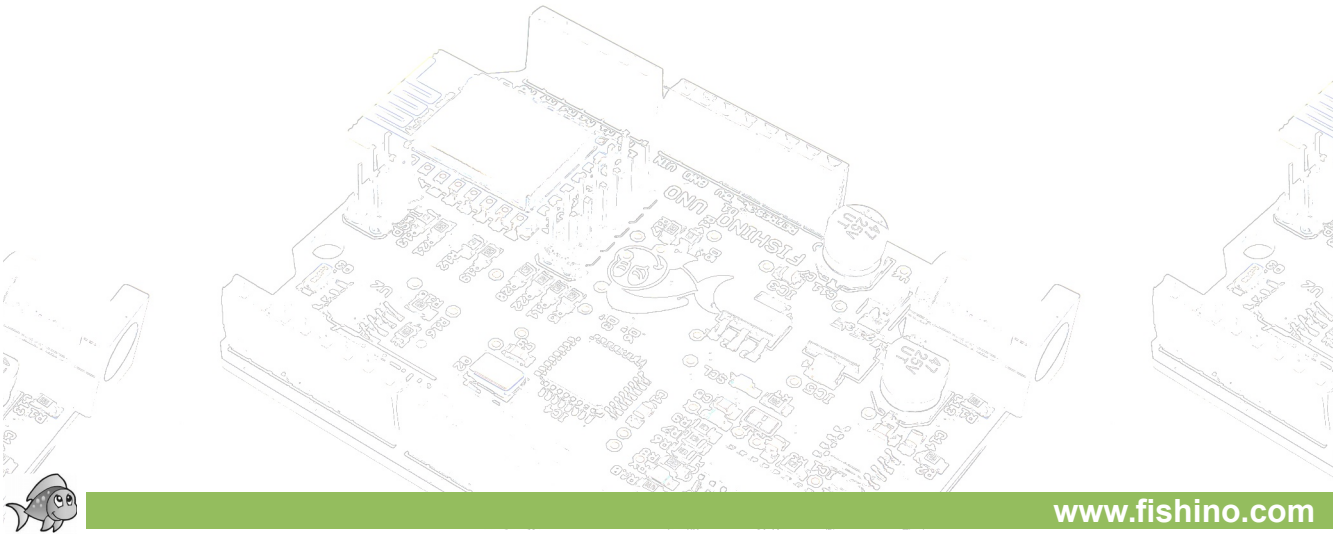
```
IPAddress IP(192.168.1.251);
IPAddress GW(192.168.1.1);
IPAddress NM(255, 255, 255, 0);
bool res = Fishino.config(IP, GW, NM);
```

Configures the network parameters with static IP 192.168.1.251, gateway 192.168.1.1 and netmask 255, 255, 255, 0.

See also :

localIP, subnetMask, gatewayIP, macAddress, setStaIP , setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS





deepSleep

```
bool deepSleep(uint32_t mSec);
```

Puts ESP module in deep sleep mode for requested time (milliseconds).

Parameters :

- **mSec** the requested sleep time

Return value :

Function returns **true** on success, **false** otherwise

Notes :

BEWARE, this function is meant to be used from inside FishinoLowPower library; to be awaken the WiFi module needs help from controller. If used alone, the function just puts the WiFi module in standby mode and issues e pulse on GPIO16 pin when requested time is elapsed.

The controller should attach an interrupt to an I/O port connected to GPIO16 (usually I/O 7 on 8 bit boards) and reset the module when triggered..

Example :

```
Fishino.deepSleep(2000);
```

Put WiFi module in deep sleep mode for 2 seconds. When the time is over the module will issue a pulse on GPIO16 pin.

See also :

digitalRead

```
uint8_t FishinoClass::digitalRead(uint8_t pin);
```

Read a digital I/O on WiFi module.

Parameters :

- **pin** the GPIO number on WiFi module

Return value :

a boolean value, true if pin is on HIGH level, false if it's on LOW level.

Notes :

Available GPIO pins on WiFi module are:

- **GPIO0** is used also during boot; DON'T keep it low at boot
- **GPIO1** it's the U0TXD pin, Tx pin for serial port. Free if serial port is not used
- **GPIO2** it's foreseen for WiFi sketch uploads. Free if not used.
- **GPIO3** it's the U0RXD, Rx pin for serial port. Free if serial port is not used
- **GPIO4** free GPIO with no limits
- **GPIO5** free GPIO with no limits

WARNING : the WiFi module operates on 3.3 Volt logic. DON'T apply a voltage greater than 3.3 Volt on its inputs, otherwise you could damage it!

Example :

```
Fishino.pinMode(5, INPUT);
bool val = Fishino.digitalRead(5);
Serial.print("GPIO5 is ");
Serial.println(val ? "HIGH" : "LOW");
```

Reads GPIO5 input and prints its value on serial port

See also :

pinMode, digitalWrite, analogRead

digitalWrite

```
bool FishinoClass::digitalWrite(uint8_t pin, uint8_t val);
```

Sets a digital I/O on WiFi module.

Parameters :

- **pin** the GPIO number on WiFi module
- **val** the value to assign to the GPIO (HIGH or LOW)

Return value :

a boolean value, **true** on comand success, **false** otherwise.

Notes :

Available GPIO pins on WiFi module are:

- **GPIO0** is used also during boot; DON'T keep it low at boot
- **GPIO1** it's the U0TXD pin, Tx pin for serial port. Free if serial port is not used
- **GPIO2** it's foreseen for WiFi sketch uploads. Free if not used.
- **GPIO3** it's the U0RXD, Rx pin for serial port. Free if serial port is not used
- **GPIO4** free GPIO with no limits
- **GPIO5** free GPIO with no limits

WARNING : the WiFi module operates on 3.3 Volt logic. DON'T apply a voltage greater than 3.3 Volt on its inputs, otherwise you could damage it!

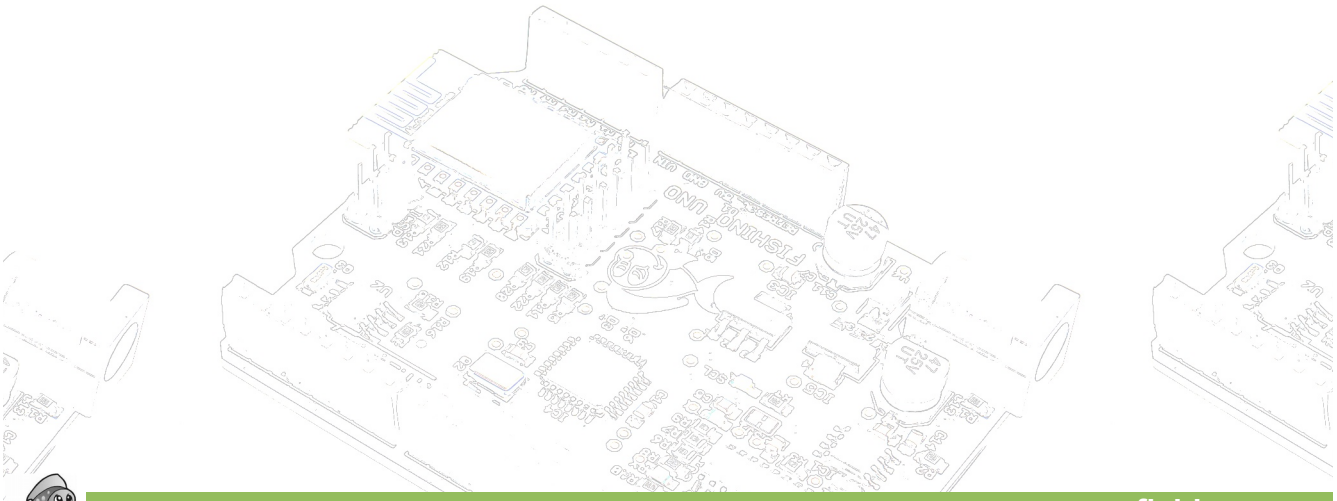
Example :

```
Fishino.pinMode(5, OUTPUT);
while(true)
{
      Fishino.digitalWrite(5, LOW);
      delay(500);
      Fishino.digitalWrite(5, HIGH);
      delay(500);
}
```

Blinks a led on GPIO5 output

See also :

pinMode, digitalRead, analogRead



disconnect

```
bool FishinoClass::disconnect(void);
```

Disconnects from an access point.

Parameters :

none

Return value :

a boolean, **true** on success, **false** otherwise

Notes :

Example :

```
Fishino.disconnect();
```

Disconnects from access point

See also :

begin, joinAp, quitAp

encryptionType

```
uint8_t encryptionType();
```

Return WiFi network encryption type, one of AUTH_MODE values.

Parameters :

none

Return value :

An AUTH_MODE enum value.

Notes :

Example :

```
uint8_t encType = Fishino.encryptionType();
Serial.print("Auth type:");
switch(encType)
{
      case AUTH_OPEN:
            Serial.println("OPEN");
            break;
      case AUTH_WEP:
            Serial.println("WEP");
            break;
      case AUTH_WPA_PSK:
            Serial.println("WPA_PSK");
            break;
      case AUTH_WPA2_PSK:
            Serial.println("WPA2_PSK");
            break;
      case AUTH_WPA_WPA2_PSK:
            Serial.println("WPA_WPA2_PSK");
            break;
      default:
            Serial.println("UNKNOWN");
            break;
}
```

The example prints the authentication type of current connection.

See also :

AUTH_MODE

ErrorCodes

```
enum ErrorCodes;
```

List of WiFi module's error codes :

```
typedef enum
{
	ERROR_NONE = 0,
	// Parameters errors
	ERROR_WRONG_PARAM_NUM,
	ERROR_INVALID_PARAM,
	// Function still not implemented
	ERROR_NOT_IMPLEMENTED,
	// Timeout errors
	ERROR_TIMEOUT,
	// WIFI station errors
	ERROR_WIFI_SCAN,
	ERROR_WIFI_BAD_MODE,
	ERROR_WIFI_NO_AP_FOUND,
	ERROR_WIFI_WRONG_PASSWORD,
	ERROR_WIFI_BAD_SSID,
	ERROR_WIFI_BAD_CHANNEL,
	ERROR_WIFI_BAD_HIDDEN_FLAG,
	ERROR_WIFI_CONNECTION_FAILED,
	ERROR_WIFI_TIMEOUT,
	// AP errors
	ERROR_AP_CONFIG_FAILED,
	ERROR_AP_CONFIG_BAD_MODE,
	ERROR_AP_CONFIG_BAD_DATA,
	// STATION errors
	ERROR_STATION_CONFIG_FAILED,
	ERROR_STATION_CONFIG_BAD_MODE,
	ERROR_STATION_CONFIG_BAD_DATA,
	// Socket errors
	ERROR_SOCKET_INVALID,
	ERROR_SOCKET_CLOSED,
	ERROR_SOCKET_OPENED,
	ERROR_NO_SOCKETS_AVAILABLE,
	// DHCP errors
	ERROR_DHCP,
	// MAC errors
	ERROR_MAC,
	// IP errors
	ERROR_IP,
	// PHY mode errors
	ERROR_BAD_PHY_MODE,
	// Server errors
```

```
    ERROR_SERVER_ALREADY,
    ERROR_SERVER_BAD_DATA,
    ERROR_SERVER_OFF,
    ERROR_SERVER_TIMEOUT,
    // Connection errors
    ERROR_CONNECTION_BAD_DATA,
    ERROR_CONNECTION_BAD_SOCKET,
    ERROR_CONNECTION_DNSFAIL,
    ERROR_CONNECTION_BAD_TYPE,
    ERROR_CONNECTION_TIMEOUT,
    ERROR_CONNECTION_GENERIC,
    // Disconnection errors
    ERROR_DISCONNECTION_BAD_DATA,
    ERROR_DISCONNECTION_BAD_SOCKET,
    ERROR_DISCONNECTION_TIMEOUT,
    ERROR_DISCONNECTION_GENERIC,
    // Send error
    ERROR_SEND_BAD_DATA,
    ERROR_SEND_BAD_SOCKET,
    ERROR_SEND_TIMEOUT,
    ERROR_SEND_TOO_BIG,
    ERROR_SEND_GENERIC,
    // Receive error
    ERROR_RECEIVE_BAD_DATA,
    ERROR_RECEIVE_BAD_SOCKET,
    ERROR_RECEIVE_TIMEOUT,
    ERROR_RECEIVE_OUT_OF_MEMORY,
    ERROR_RECEIVE_GENERIC,
    ERROR_RECEIVE_NO_DATA,
    // UART errors
    ERROR_UART_BAD_SPEED,
    ERROR_UART_BAD_PARITY,
    ERROR_UART_BAD_BITS,
    ERROR_UART_BAD_STOP_BITS,
    // PORTS ERROR
    ERROR_PORT_INVALID,
    ERROR_PORTMODE_INVALID,
    // UDP ERRORS
    ERROR_UDP_CONNECTION_BAD_DATA,
    ERROR_UDP_CONNECTION_BAD_SOCKET,
    ERROR_UDP_CONNECTION_DNSFAIL,
    ERROR_UDP_CONNECTION_GENERIC,
    ERROR_UDP_RECEIVE_BAD_DATA,
    ERROR_UDP_RECEIVE_BAD_SOCKET,
    ERROR_UDP_RECEIVE_TIMEOUT,
    ERROR_UDP_RECEIVE_OUT_OF_MEMORY,
    ERROR_UDP_RECEIVE_GENERIC,
    ERROR_UDP_RECEIVE_NO_PACKET_PARSED,
    ERROR_UDP_RECEIVE_NO_DATA,
```

```
        ERROR_UDP_SEND_BAD_DATA,
        ERROR_UDP_SEND_BAD_SOCKET,
        ERROR_UDP_SEND_TIMEOUT,
        ERROR_UDP_SEND_TOO_BIG,
        ERROR_UDP_SEND_GENERIC,
        ERROR_UDP_SEND_NO_DATA,
        ERROR_UNKNOWN,
        ERROR_MAX
    } ErrorCodes;
```

Notes :

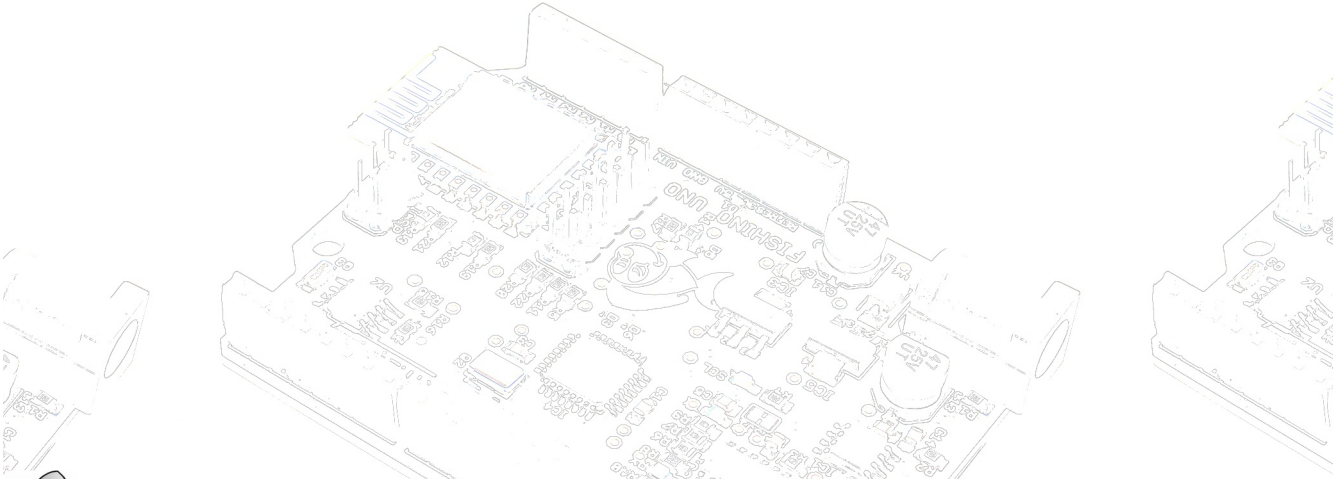
Example :

```
uint16_t errCode = Fishino.getLastError();
Serial.print("Last error code found : ");
Serial.println(errCode);
Serial.print("Corresponding error message : ");
const char *msg = Fishino.getErrorString(errCode);
Serial.println(msg);
free(msg);
```

The example prints last error code and its corresponding message on serial port.

See also :

getLastError, getLastErrorString, getErrorString, clearLastError, showErrors



firmwareVersion

```
uint32_t FishinoClass::firmwareVersion(void);
```

Read WiFi module's firmware version.

Parameters :

none

Return value :

The firmware version encoded in a 32 bit value; first word is major version, second word is composed by minor version (first byte) and release version (second byte).

Notes :

Example :

```
uint32_t ver = Fishino.firmwareVersion();
uint16_t major = ver >> 16;
uint8_t min = ver >> 8;
uint8_t rel = ver;
Serial.print("Firmware version : ");
Serial.print(major);
Serial.print(".");
Serial.print(min);
Serial.print(".");
Serial.println(rel);
```

Prints firmware version as 'maj.min.rel' on serial port.

See also :

firmwareVersionStr

firmwareVersionStr

```
char *FishinoClass::firmwareVersionStr(void);
```

Read WiFi module's firmware version in string format.

Parameters :

none

Return value :

The firmware version as a pointer to a static character array.

Notes :

The return value is a static character array which must NOT be freed.

Example :

```
const char *ver = Fishino.firmwareVersionStr();
Serial.print("Firmware version : ");
Serial.println(ver);
```

Prints firmware version on serial port.

See also :

firmwareVersion

freeRam

```
static uint32_t freeRam(void);
```

An utility function to display the available RAM memory.

Parameters :

none

Return value :

The available RAM memory, in bytes.

Notes :

Example :

```
Serial.print("Available RAM : ");
Serial.println(Fishino.freeRam());
```

Prints the amount of free RAM on serial port.

See also :

gatewayIP

```
IPAddress FishinoClass::gatewayIP();
```

Return the IP address of gateway.

Parameters :

none

Return value :

Gateway's IP address.

Notes :

Example :

```
IPAddress gw = Fishino.gatewayIP();
Serial.print("Gateway'IP address is : ");
Serial.println(gw);
```

Prints gateway's IP address on serial port.

See also :

config, localIP, subnetMask, macAddress, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

getApIPInfo

```
bool FishinoClass::getApIPInfo(IPAddress &ip, IPAddress &gateway, IPAddress &netmask);
```

Read Access Point mode IP informations.

Parameters :

- **ip** an IPAddress variable that will contain AP IP address
- **gateway** an IPAddress variable that will contain AP gateway address
- **netmask** an IPAddress variable that will contain AP netmask

Return value :

A boolean value, **true** on success, **false** otherwise
Results are returned in the 3 parameters

Notes :

Example :

```
IPAddress ip, gw, nm;
Fishino.getApIpInfo(ip, gw, nm);
Serial.print("Access point IP is :");
Serial.println(ip);
Serial.print("Access point Gateway is :");
Serial.println(gw);
Serial.print("Access point Netmask is :");
Serial.println(nm);
```

The example prints access point's ip, gateway and netmask values.

See also :

setApIPInfo

getErrorString

```
const char *FishinoClass::getErrorString(uint16_t errCode);
```

Gets an human-readable error message from an error code.

Parameters :

An error code, one of ErrorCodes enum

Return value :

A dynamically allocated string of characters that MUST be freed after usage.

Notes :

The returned string is dynamically allocater with malloc() and MUST be freed with free() after usage.

Example :

```
for(uint16_t err = ERROR_NONE; err < ERROR_MAX; err++)
{
	const char *errMsg = Fishino.getErrorString(err);
	Serial.println(errMsg);
	free(errMsg);
}
```

The example prints all error messages on serial port.

See also :

ErrorCodes, getLastError, getLastErrorString, clearLastError, showErrors

getHostName

```
String getHostName(void);
```

Gets the host name of this device.

Parameters :

None

Return value :

A String with requested host name.

Notes :

Example :

```
Serial.print("My host name is ");
Serial.println(Fishino.getHostName());
```

The example prints the device's host name.

See also :

setHostName

getLastError

```
int16_t FishinoClass::getLastError(void);
```

Gets last WiFi module error code.

Parameters :

none

Return value :

The last error code from WiFi module.

Notes :

Last error code is stored when the WiFi module's firmware finds an error in some operation. Can be cleared by FishinoClass::clearLastError().

Example :

```
uint16_t lastErr = Fishino.getLastError();
Serial.print("Last error code was : ");
Serial.println(lastErr);
```

This example prints last error code on serial port.

See also :

clearLastError, getLastErrorString, getErrorString, ErrorCodes, showErrors

getLastErrorString

```
const char *FishinoClass::getLastErrorString(void);
```

Get last error message from WiFi module.

Parameters :

none

Return value :

A dynamically allocated string of characters that MUST be freed after usage.

Notes :

The returned string is dynamically allocater with malloc() and MUST be freed with free() after usage.

Example :

```
const char *errMsg = Fishino.getLastErrorString();
Serial.print("Last error message is :");
Serial.println(errMsg);
free(errMsg);
```

The example prints last error message on serial port.

See also :

ErrorCodes, getLastError, getErrorString, clearLastError, showErrors

getMaxTcpConnections

```
uint8_t getMaxTcpConnections(void);
```

Get the number of concurrent TCP connections accepted by WiFi module.

Parameters :

none

Return value :

The number of concurrent allowed TCP connections.

Notes :

Example :

```
Serial.print("Allowed TCP connections :");
Serial.println(Fishino.getMaxTcpConnections());
```

The example prints the number of allowed TCP connections on serial port.

See also :

setMaxTcpConnections

getMode

```
uint8_t FishinoClass::getMode(void);
```

Reads WiFi module's operating mode.

Parameters:

none

Return value:

One of following values (see WIFI_MODE enum) :

- **STATION_MODE**
  Station mode. It needs an existing WiFi infrastructure to connect to. This one is the most common mode.
- **SOFTAP_MODE**
  Access point mode. It creates an autonomous WiFi infrastructure. Useful if no existing infrastructure is present.
- **STATIONAP_MODE**
  Dual mode. The WiFi module operates both as a station and as an access point.

Notes:

Example:

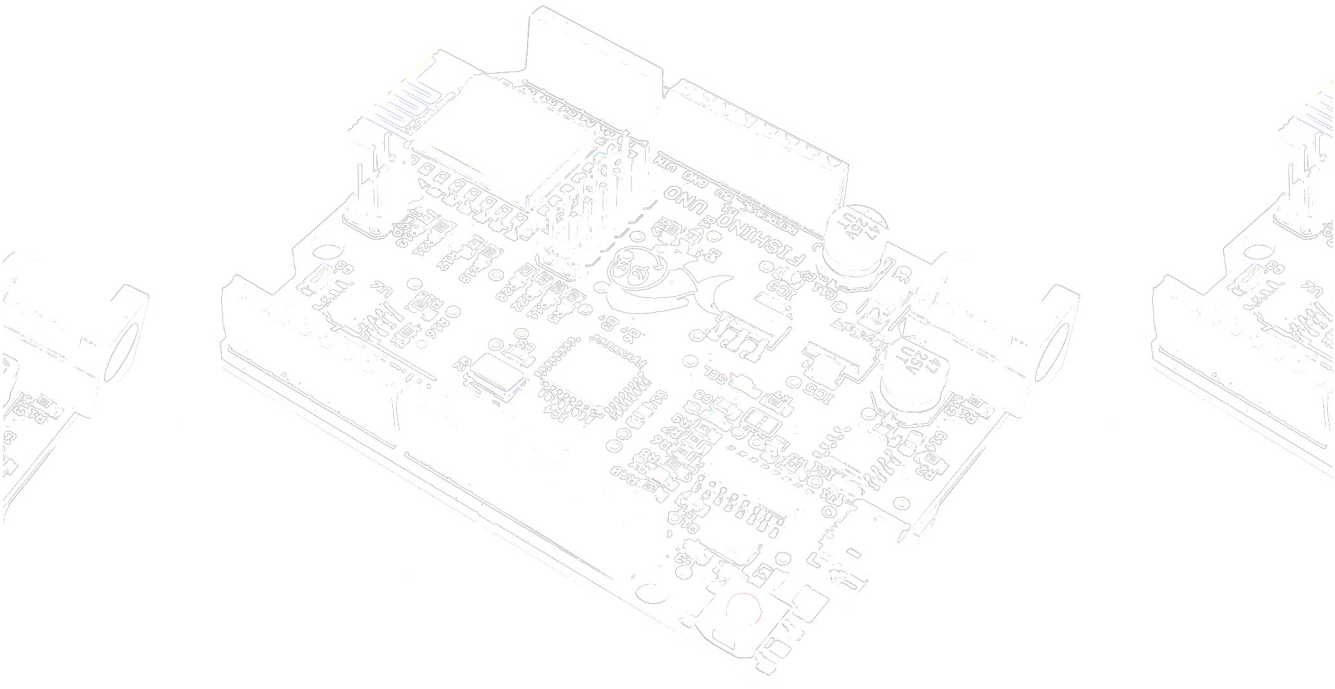
```
uint8_t mode = Fishino.getMode();
Serial.print("Current operation mode is :");
switch(mode)
{
	case STATION_MODE:
		Serial.println("STATION");
		break;
	case SOFTAP_MODE:
		Serial.println("ACCESS POINT");
		break;
	case STATIONAP_MODE:
		Serial.println("STATION + ACCESS POINT");
		break;
	default:
		Serial.println("UNKNOWN");
```

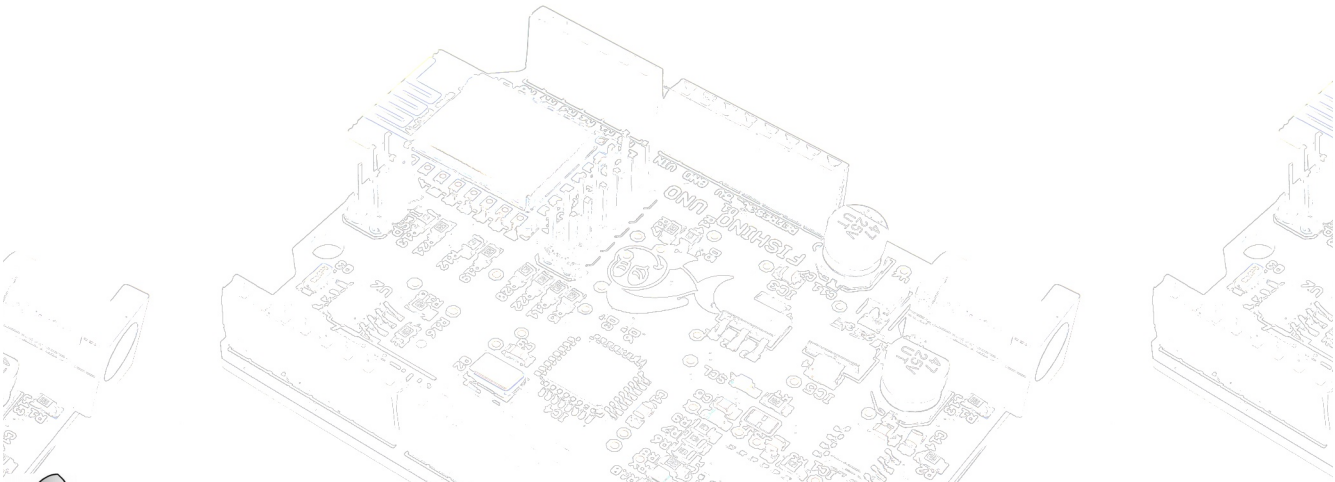
```
            break;
    }
```

The example prints current operation mode on serial port.

See also:

setMode, WIFI_MODE





getNumNetworks

```
uint8_t getNumNetworks();
```

Get the number of networks found on last scan without re-scanning, or re-scan if none

Parameters:

none

Return value:

The number of networks found on last scanNetworks() call.:

Notes:

Example:

```
Serial.print("Number of available WiFi networks is ");
Serial.println((int)Fishino.getNumNetworks());
```

The example prints the number of available WiFi networks

See also:

scanNetworks

getPhyMode

```
uint8_t FishinoClass::getPhyMode(void);
```

Gets PHY mode of currente WiFi connection (see PHY_MODE enum).

Parameters :

none

Return value :

One of PHY_MODE enum values :

- **PHY_MODE_UNKNOWN**
- **PHY_MODE_11B**
- **PHY_MODE_11G**
- **PHY_MODE_11N**

Notes :

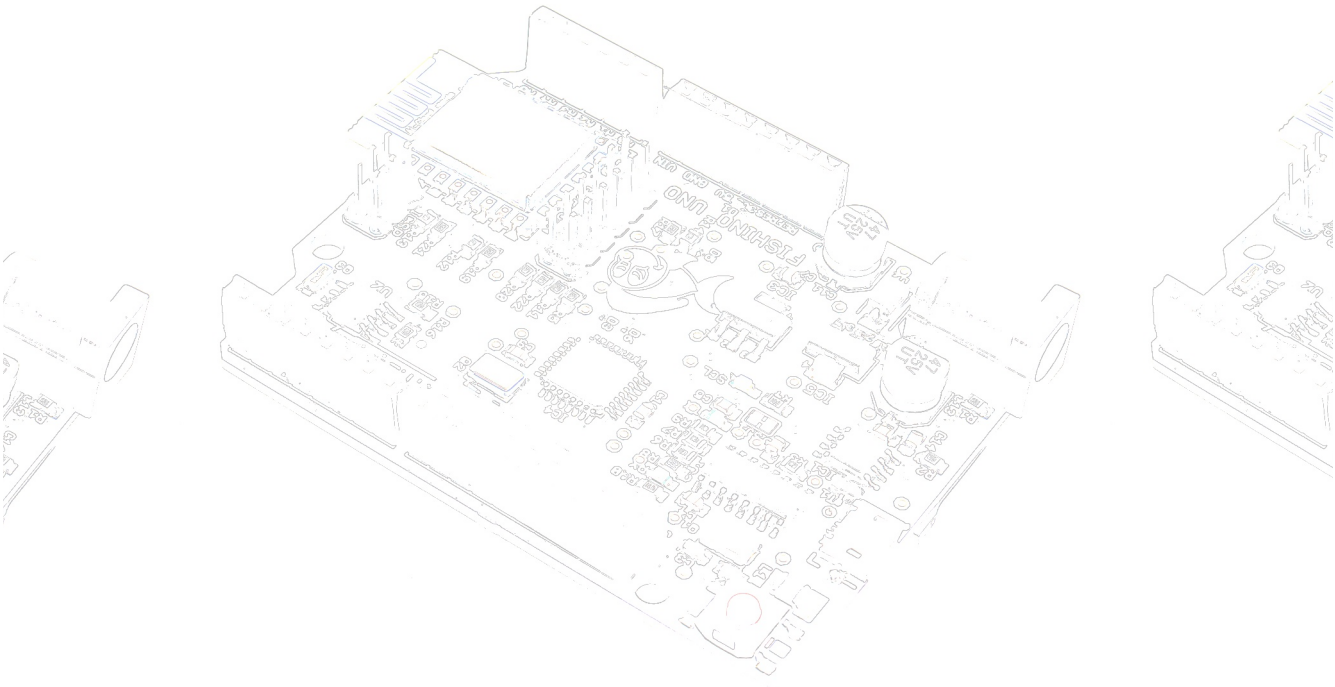
Example :

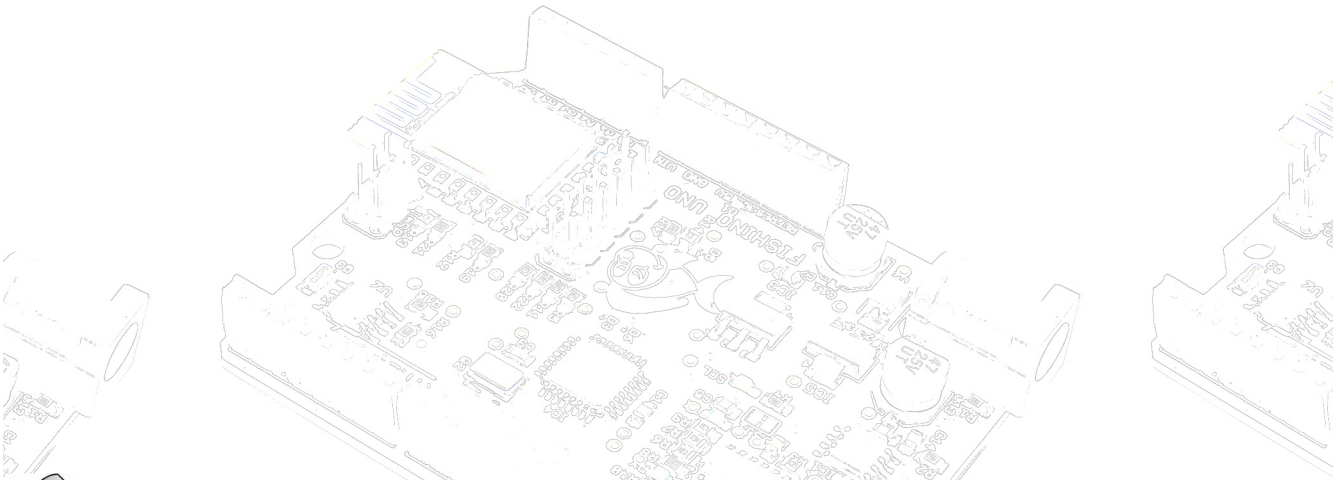
```
uint8_t mode = Fishino.getPhyMode();
Serial.print("Current PHY mode is : ");
switch(mode)
{
	case PHY_MODE_11B:
		Serial.println("11B");
		break;
	case PHY_MODE_11G:
		Serial.println("11G");
		break;
	case PHY_MODE_11N:
		Serial.println("11N");
	break;
	default:
		Serial.println("UNKNOWN");
		break;
}
```

Prints current PHY mode on serial port.

See also :

setPhyMode, PHY_MODE





getSoftApDHCPServerStatus

```
bool getSoftApDHCPServerStatus(void);
```

The function returns current soft Ap's DHCP server status : false if off or true if on.

Parameters :

none

Return value :

Boolean value, **true** if DHCP client is on , **false** if off.

Notes :

Example :

```
if(Fishino.getSoftApDHCPStatus())
      Serial.println("Fishino Ap DHCP server is ON");
else
      Serial.println("Fishino Ap DHCP server is OFF");
```

The example checks if soft Ap's DHCP server is enabled or not and displays the result on serial port.

See also :

softApStartDHCPServer, softApStopDHCPServer

getStaConfig

```
bool FishinoClass::getStaConfig(char *&ssid, char *&pass);
```

Return current station connection parameters (SSID and password).

Parameters :

- **SSID** a pointer to a character array that will receive current SSID
- **pass** a pointer to a character array that will receive current password

Return value :

A boolean value, true on success and false otherwise.
Connection data is returned into given parameters.

Notes :

Returned data is dinamycally allocated with malloc() and MUST be freed by free() after usage. See the example for details

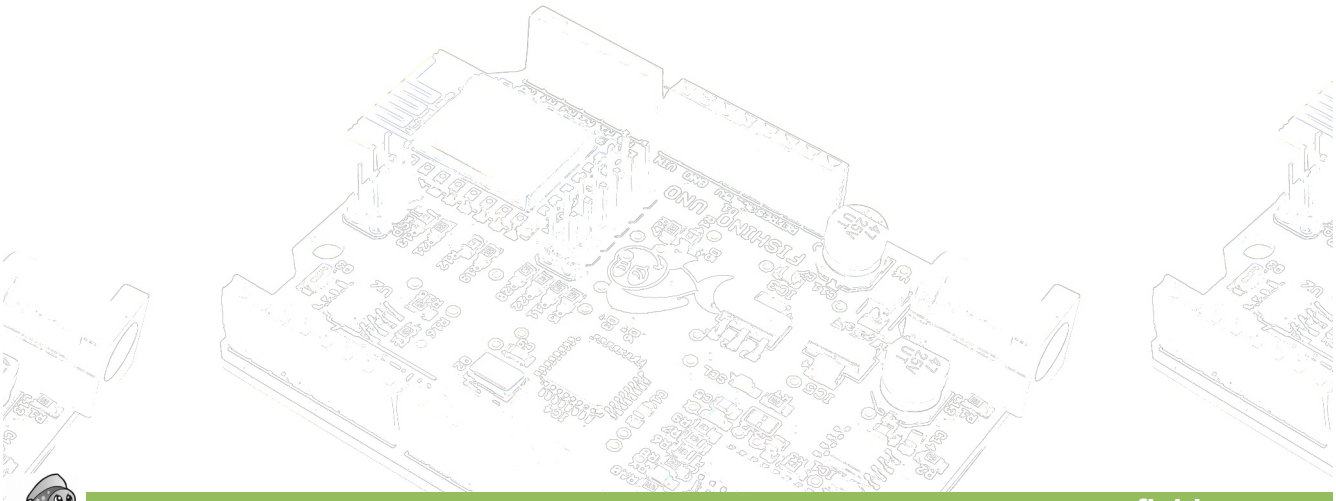
Example :

```
char *ssid, *pass;
bool res = Fishino.getStaConfig(ssid, pass);
if(res)
{
      Serial.print("SSID : ");
      Serial.println(ssid);
      Serial.print("PASSWORD : ");
      Serial.println(pass);
      free(ssid);
      free(pass);
}
else
      Serial.println("Error getting connection parameters");
```

This example prints SSID and password of current AP connection.

See also :

setStaConfig, begin, joinAp, quitAp, disconnect



getStaDHCPStatus

```
bool FishinoClass::getStaDHCPStatus(void);
```

The function returns current DHCP client status : false if off (static IP in use) or true if on (dynamic IP in use).

Parameters :

none

Return value :

Boolean value, **true** if DHCP client is on (dynamic IP), **false** if off (static IP).

Notes :

Example :

```
if(Fishino.getDHCPStatus())
      Serial.println("Fishino uses a dynamic IP");
else
      Serial.println("Fishino uses a static IP");
```

The example checks if we're using a dynamic or static IP address and print the result on serial port.

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, staStopDHCP, setDNS

hostByName

```
bool FishinoClass::hostByName(const char* aHostname, IPAddress& aResult);
```

The function queries the IP address of a given host.

Parameters :

- **aHostName** host of which we want to get the IP address (example "www.timberstruct.eu")
- **aResult** an IPAddress variable to hold the resulting IP

Return value :

A boolean, **true** on success, **false** otherwise.
Requested IP is returned on aResult parameter.

Notes :

Example :

```
IPAddress ip;
const char *host = "www.timberstruct.eu";
if(Fishino.hostByName(host, ip))
{
      Serial.print("IP address of '");
      Serial.print(host);
      Serial.print("'is :");
      Serial.println(ip);
}
else
      Serial.println("Error in hostByName function call");
```

The example queries for IP address of ‘www.timberstruct.eu’ host and prints it on serial port.

See also :

config, setDNS

joinAp

```
bool FishinoClass::joinAp(void);
```

Joins an access point. Connection parameters must be have been configured with a previous call to setStaConfig() or begin() functions.

Parameters :

none

Return value :

boolean, **true** on success, **false** otherwise

Notes :

AP parameters must have been set before this call.

Example :

```
Fishino.setStaConfig("MY_SSID", "MY_PASS");
if(Fishino.joinAp())
Serial.println("Connected succesfully to access point");
else
Serial.println("Failed to connect to access point");
```

The example configures station for the required access point and try to join it.

See also :

getStaConfig, setStaConfig, begin, quitAp, disconnect

JOIN_STATUS

```
enum JOIN_STATUS;
```

The connection status returned by FishinoClass::status() function; one of following values:

```
typedef enum
{
	STATION_IDLE,
	STATION_CONNECTING,
	STATION_WRONG_PASSWORD,
	STATION_NO_AP_FOUND,
	STATION_CONNECT_FAIL,
	STATION_GOT_IP,
	STATION_TIMEOUT,
	STATION_BAD_SSID,
} JOIN_STATUS;
```

Notes :

The connection is completed when the FishinoClass::status() function returns STATION_GOT_IP value.
STATION_IDLE means no connection in progress; STATION_CONNECTING means a connection in progress but still not completed.
Any other value is a connection error.

Example :

```
uint8_t stat = Fishino.status();
switch(stat)
{
	case STATION_IDLE :
		Serial.println("No connection in progress");
		break;
	case STATION_CONNECTING :
		Serial.println("Connection in progress");
		break;
	case STATION_WRONG_PASSWORD :
		Serial.println("Wrong password");
		break;
	case STATION_NO_AP_FOUND :
		Serial.println("Access point not found");
		break;
	case STATION_CONNECT_FAIL :
		Serial.println("Connection failed");
```

```
            break;
      case STATION_GOT_IP :
            Serial.println("Successfully connected to AP");
            break;
      case STATION_TIMEOUT :
            Serial.println("Timeout connecting to AP");
            break;
      case STATION_BAD_SSID :
            Serial.println("Bad SSID");
            break;
      default:
            Serial.println("Unknown connection error");
            break;
}
```

This example prints the status of connection to access point.

See also :

status

localIP

```
IPAddress FishinoClass::localIP();
```

Retrieve Fishino's IP in station mode (both dynamic and static).

Parameters :

none

Return value :

The local IP address

Notes :

Example :

```
IPAddress ip = Fishino.localIP();
Serial.print("Fishino's IP is : ");
Serial.println(ip);
```

The example prints local IP on serial port.

See also :

config, gatewayIP, subnetMask, macAddress, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

macAddress

```
const uint8_t* FishinoClass::macAddress(void);
```

Retrieves Fishino's MAC address in station mode.

Parameters :

none

Return value :

A pointer to a static buffer of WL_MAC_ADDR_LENGTH containing the requested MAC address
The returned buffer is static and must NOT be freed.

Notes :

The returned buffer is static and must NOT be freed.

Example :

```
uint8_t const *mac = Fishino.macAddress();
Serial.print("MAC ");
for(int i = 0; i < WL_MAC_ADDR_LENGTH; i++)
{
      Serial.print(":");
      Serial.print(mac[i], HEX);
}
Serial.println();
```

Prints Fishino's MAC address.

See also :

config, gatewayIP, subnetMask, localIP, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

ntpEpoch

```
uint32_t ntpEpoch(void);
```

Get Epoch (seconds since 1 january 1900 from NTP server

Parameters :

none

Return value :

The number of seconds elapsed since January 1, 1900.

Notes :

Example :

```
Serial.print("Seconds since January 1, 1900 : ");
Serial.println(Fishino.ntpEpoch());
```

Prints seconds since January 1, 1900.

See also :

ntpGetServer, ntpSetServer, ntpTime

ntpGetServer

```
bool ntpGetServer(IPAddress &ip);
```

Gets NTP server

Parameters :

- **ip** A variable that will receive the requested IP address

Return value :

boolean, **true** on success, **false** otherwise. The requested IP address is returned in ip parameter.

Notes :

Example :

```
IPAddress ip;
ntpGetServer(ip);
```

Gets current NTP server's IP address.

See also :

ntpEpoch, ntpSetServer, ntpTime

ntpSetServer

```
bool ntpSetServer(IPAddress const &ip);
bool ntpSetServer(const char *server);
```

Sets NTP server.

Parameters :

- **ip** the IP address of new NTP server
- **server** the hostname of new NTP server

Return value :

boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
Fishino.ntpSetServer("0.it.pool.ntp.org");
```

Changes active NTP server to "0.it.pool.ntp.org".

See also :

ntpEpoch, ntpGetServer, ntpTime

ntpTime

```
bool ntpTime(uint8_t &hour, uint8_t &minute, uint8_t &second);
```

Get current time (hour, minute and second) from NTP server.

Parameters :

- **hour** a variable that will receive current hour
- **minute** a variable that will receive current minute
- **second** a variable that will receive current second

Return value :

boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
uint8_t hour, minute, second;
if(Fishino.ntpTime(hour, minute, second))
{
	Serial.print("Current time : ");
	Serial.print((int)hour);
	Serial.print(":")
	Serial.print((int)minute);
	Serial.print(":")
	Serial.println((int)second);
}
else
	Serial.println("Error getting time from NTP server");
```

Prints current time in HH:MM:SS format on serial port.

See also :

ntpEpoch, ntpGetServer, ntpSetServer

PHY_MODE

```
enum PHY_MODE;
```

List of WiFi physical modes :

```
typedef enum
{
	PHY_MODE_UNKNOWN,
	PHY_MODE_11B,
	PHY_MODE_11G,
	PHY_MODE_11N
} PHY_MODE;
```

Notes :

Example :

```
uint8_t mode = Fishino.getPhyMode();
Serial.print("Current PHY mode is : ");
switch(mode)
{
	case PHY_MODE_11B:
		Serial.println("11B");
		break;
	case PHY_MODE_11G:
		Serial.println("11G");
		break;
	case PHY_MODE_11N:
		Serial.println("11N");
		break;
	default:
		Serial.println("UNKNOWN");
	break;
}
```

Prints current PHY mode on serial port.

See also :

setPhyMode, getPhyMode

pinMode

```
bool FishinoClass::pinMode(uint8_t pin, uint8_t mode);
```

Sets mode of a digital I/O pin (INPUT, INPUT_PULLUP or OUTPUT).

Parameters :

- **pin** the GPIO pin number
- **mode** the requested mode (INPUT, INPUT_PULLUP or OUTPUT)

Return value :

A boolean value, **true** on success, **false** otherwise

Notes :

Available GPIO pins on WiFi module are:

- GPIO0 is used also during boot; DON'T keep it low at boot
- GPIO1 it's the U0TXD pin, Tx pin for serial port. Free if serial port is not used
- GPIO2 it's foreseen for WiFi sketch uploads. Free if not used.
- GPIO3 it's the U0RXD, Rx pin for serial port. Free if serial port is not used
- GPIO4 free GPIO with no limits
- GPIO5 free GPIO with no limits

WARNING : the WiFi module operates on 3.3 Volt logic. DON'T apply a voltage greater than 3.3 Volt on its inputs, otherwise you could damage it!

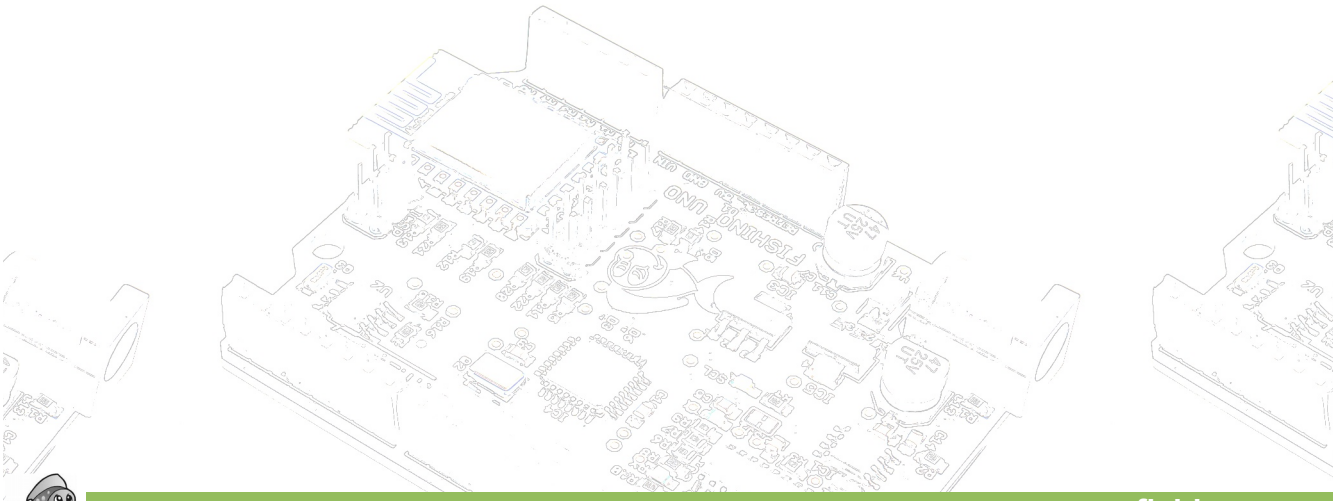
Example :

```
Fishino.pinMode(5, OUTPUT);
while(true)
{
      Fishino.digitalWrite(5, LOW);
      delay(500);
      Fishino.digitalWrite(5, HIGH);
      delay(500);
}
```

Blinks a led on GPIO5 output

See also :

digitalWrite, digitalRead, analogRead



quitAp

```
bool FishinoClass::quitAp(void);
```

Disconnects from an access point.

Parameters :

none

Return value :

a boolean, **true** on success, **false** otherwise

Notes :

Example :

```
Fishino.quitAp();
```

Disconnects from access point

See also :

begin, joinAp, disconnect

reset

```
bool FishinoClass::reset(void);
```

Send a reset command to WiFi module and waits for it to be ready.

Parameters :

none

Return value :

Boolean value, **true** on success, **false** otherwise

Notes :

The function may not return if it finds a wrong firmware version.
In that case it will send a message to serial port and hang the sketch.

Example :

```
Serial.print("Resetting WiFi module...");
while(!Fishino.reset())
{
      Serial.print(".");
      delay(100);
}
Serial.println("OK");
```

The example tries to reset the module forever and continues when succeed.

See also :

firmwareVersion, firmwareVersionStr

RSSI

```
int32_t FishinoClass::RSSI();
int32_t FishinoClass::RSSI(uint8_t networkItem);
```

The function returns the WiFi signal strength on currently active AP (no parameters) or another available AP (the parameter is requested AP number).

Parameters :

networkItem a number between 0 and the value returned by scanNetworks() function minus one

Return value :

The requested signal strength, as a 32 bit integer.

Notes :

Example :

```
Serial.print("Signal strength : ");
Serial.println(Fishino.RSSI());
```

The example prints signal strength of currently associated AP.

See also :

SSID, BSSID, encryptionType

scanNetworks

```
uint8_t FishinoClass::scanNetworks(void);
```

Scan for available WiFi networks.

Parameters :

none

Return value :

The number of available WiFi networks

Notes :

The function stores the found WiFi networks data (SSID, BSSID, RSSI) into an internal list which can be queried later.
See the example for details.

Example :

```
uint8_t numNets = Fishino.scanNetworks();
Serial.print("Found ");
Serial.print((int)numNets);
Serial.println(" WiFi networks");
Serial.println("List of SSIDs :");
for(int i = 0; i < numNets; i++)
{
      Serial.print("Network #");
      Serial.print(i);
      Serial.print(" SSID:");
      Serial.println(Fishino.SSID(i));
}
```

The example show the names (SSIDs) of all found WiFi networks.

See also :

SSID, BSSID, RSSI, encryptionType, getNumNetworks

setApGateway

```
bool FishinoClass::setApGateway(IPAddress gw);
```

Sets gateway for Access Point mode.

Parameters :

gw the AP gateway address

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
IPAddress gw(192.168.2.1);
Fishino.setApGateway(gw);
```

The example sets AP gateway

See also :

setApIP, setApMAC, setApNetMask, setApIPInfo, getApIPInfo

setApIP

```
bool FishinoClass::setApIP(IPAddress ip);
```

Sets IP for Access Point mode.

Parameters :

- **ip** the AP IP address

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
IPAddress ip(192.168.2.1);
Fishino.setApIP(ip);
```

The example sets AP IP address

See also :

setApGateway, setApMAC, setApNetMask, setApIPInfo, getApIPInfo

setApIPInfo

```
bool FishinoClass::setApIPInfo(IPAddress ip, IPAddress gateway, IPAddress netmask);
```

Sets IP, gateway and netmask for Access Point mode.

Parameters :

- **ip** the AP IP address
- **gateway** the AP gateway address
- **netmask** the AP netmask

Return value :

Boolean, true on success, false otherwise.

Notes :

Example :

```
IPAddress ip(192.168.2.1);
IPAddress gw(192.168.2.1);
IPAddress nm(255.255.255.0);
Fishino.setApIPInfo(ip, gw, nm);
```

The example sets AP IP info

See also :

setApIP, setApMAC, setApNetMask, setApGateway, getApIPInfo

setApMAC

```
bool FishinoClass::setApMAC(uint8_t const *mac);
```

Sets MAC address for Access Point mode.

Parameters :

- **mac** an array of WL_MAC_ADDR_LENGTH containing the AP MAC address

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
uint8_t mac[WL_MAC_ADDR_LENGTH] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66};
Fishino.setApMAC(mac);
```

The example sets AP MAC address

See also :

setApIP, setApGateway, setApNetMask, setApIPInfo, getApIPInfo

setApNetMask

```
bool FishinoClass::setApNetMask(IPAddress nm);
```

Sets netmask for Access Point mode.

Parameters :

- **nm** the AP netmask

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
IPAddress nm(255.255.255.0);
Fishino.setApNetMask(nm);
```

The example sets AP netmask

See also :

setApIP, setApMAC, setApGateway, setApIPInfo, getApIPInfo

setDNS

```
bool setDNS(IPAddress dns_server1);
bool setDNS(IPAddress dns_server1, IPAddress dns_server2);
```

The function sets one or both DNS addresses

Parameters :

- **dns_server1** first DNS server
- **dns_server2** second DNS server

Return value :

Boolean, **true** on success, **false** otherwise

Notes :

If you don't use this function the DNS servers are set to default values.

Example :

```
IPAddress dns1(208, 67, 222, 222);
IPAddress dns2(208, 67, 220, 220);
Fishino.setDNS(dns1, dns2);
```

The example sets DNS to OpenDNS addresses

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus

setHostName

```
bool setHostName(const char *hostName);
bool setHostName(const __FlashStringHelper *hostName);
```

Sets the host name of this device.

Parameters :

A C string containing the requested host name.

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
Fishino.setHostName("MyFishinoBoard"));
```

The example changes the device's host name.

See also :

getHostName

setMaxTcpConnections

```
bool setMaxTcpConnections(uint8_t n);
```

Sets the maximum number of allowed concurrent TCP connections.

Parameters :

- **n** The number of TCP connections.

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
Fishino.setMaxTcpConnections(3);
```

The example changes sets the maximum number of concurrent TCP connections to 3.

See also :

getMaxTcpConnections

setMode

```
bool FishinoClass::setMode(uint8_t mode);
```

Sets WiFi module's operating mode.

Parameters :

mode parameter can be one of following:

- **STATION_MODE**
  Station mode. It needs an existing WiFi infrastructure to connect to. This one is the most common mode.
- **SOFTAP_MODE**
  Access point mode. It creates an autonomous WiFi infrastructure. Useful if no existing infrastructure is present.
- **STATIONAP_MODE**
  Dual mode. The WiFi module operates both as a station and as an access point.

Return value :

boolean, **true** on success, **false** otherwise

Notes :

Example :

```
Fishino.setMode(STATION_MODE);
```

Set operating mode to **STATION**

See also:

getMode, WIFI_MODE

setPhyMode

```
bool FishinoClass::setPhyMode(uint8_t mode);
```

Sets PHY mode of currente WiFi connection (see PHY_MODE enum).

Parameters :

- **mode** The requested PHY mode, one of the values in PHY_MODE enum:
  PHY_MODE_11B
  PHY_MODE_11G
  PHY_MODE_11N

Return value :

Boolean, **true** on success, **false** otherwise

Notes :

Example :

```
Fishino.setPhyMode(PHY_MODE_11G);
```

Sets 11G mode.

See also :

getPhyMode, PHY_MODE

setStaConfig

```
bool FishinoClass::setStaConfig(const char *ssid);
bool FishinoClass::setStaConfig(const __FlashStringHelper *ssid);
bool FishinoClass::setStaConfig(const char *ssid, const char *passphrase);
bool FishinoClass::setStaConfig(const __FlashStringHelper *ssid, const __FlashStringHelper
*passphrase);
```

Set AP connection parameters without joining it.

Parameters :

- **ssid** SSID of access point
- **passphrase** WiFi password

Return value :

Boolean, **true** on success, **false** otherwise

Notes :

Both parameters can be memory or PROGMEM strings.

Example :

```
Fishino.setStaConfig("MY_SSID", "MY_PASS");
Fishino.joinAp();
```

Sets access point parameters and join it.

See also :

getStaConfig, begin, joinAp, quitAp, disconnect

setStaGateway

```
bool FishinoClass::setStaGateway(IPAddress gw);
```

Sets gateway for station mode.

Parameters :

- **gw** address of the gateway

Return value :

Boolean, **true** on success, **false** otherwise

Notes :

Example :

```
IPAddress gw(192, 168, 1, 1);
Fishino.setStaGateway(gw);
```

The example sets station gateway.

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaIP, setStaMAC, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

setStaIP

```
bool FishinoClass::setStaIP(IPAddress ip);
```

Sets IP for station mode.

Parameters :

- **ip** IP address for station mode

Return value :

Boolean, **true** on success, **false** otherwise

Notes :

Example :

```
IPAddress ip(192, 168, 1, 251);
Fishino.setStaIP(ip);
```

The example sets station IP.

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaGateway, setStaMAC, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

setStaMAC

```
bool FishinoClass::setStaMAC(uint8_t const *mac);
```

Sets MAC address for station mode.

Parameters :

- **mac** an array of WL_MAC_ADDR_LENGTH containing the AP MAC address

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
uint8_t mac[WL_MAC_ADDR_LENGTH] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66};
Fishino.setStaMAC(mac);
```

The example sets station MAC address

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaGateway, setStaIP, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

setStaNetMask

```
bool FishinoClass::setStaNetMask(IPAddress nm);
```

Sets netmask for station mode.

Parameters :

- **nm** the netmask address for station mode

Return value :

Boolean, **true** on success, **false** otherwise

Notes :

Example :

```
IPAddress nm(255, 255, 255, 0);
Fishino.setStaNetMask(nm);
```

The example sets station netmask.

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaGateway, setStaMAC, setStaIP, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

showErrors

```
void showErrors(bool s);
```

Enable/disable error messages display

Parameters :

- **s** boolean, true to enable messages, false otherwise.

Return value :

none

Notes :

Example :

```
Fishino.showErrors(false);
```

The example disable display of error messages.

See also :

clearLastError, ErrorCodes, getErrorString, getLastError, getLastErrorString

SOCK_STATUS

```
enum SOCK_STATUS;
```

TCP socket status, used mainly inside FishinoClient and FishinoServer classes; one of following values:

```
typedef enum _SOCK_STATUS
{
      SOCK_INVALID,
      SOCK_DISCONNECTED,
      SOCK_CONNECTED,
} SOCK_STATUS;
```

Notes :

Example :

See also :

softApConfig

```
bool FishinoClass::softApConfig(const char *SSID, const char *pass, uint8_t channel, bool hidden =
false);
bool FishinoClass::softApConfig(const __FlashStringHelper *SSID, const __FlashStringHelper
*passphrase, uint8_t channel, bool hidden = false);
```

Sets AP connection parameters.

Parameters :

- **SSID** the network name (SSID) of access point
- **pass** the password for access point (leave blank for open network)
- **channel** the access point operating channel
- **hidden** true if you want to create an hidden network, false otherwise

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

To create an open network leave the password blank (“”).
SSID and pass parameters can be normal (RAM) strings or PROGMEM (flash) ones.

Example :

```
Fishino.softApConfig("MY_SSID", "MY_PASS", 7, false);
```

The example create a the network MY_SSID with password MY_PASS on channel 7 with visible SSID.

See also :

softApGetConfig, softApGetChannel, softApGetHidden, softApGetPassword, softApGetSSID, softApStartDHCPServer, softApStopDHCPServer, getSoftApDHCPServerStatus

softApGetChannel

```
uint8_t FishinoClass::softApGetChannel(void);
```

Gets soft AP operating channel.

Parameters :

none

Return value :

The requested channel

Notes :

Example :

```
Serial.print("Fishino's AP operating channel : ");
Serial.println((int)Fishino.softApGetChannel());
```

The example prints current operating channel on serial port.

See also :

softApGetConfig, softApGetHidden, softApGetPassword, softApGetSSID, softApStartDHCPServer, softApStopDHCPServer, getSoftApDHCPServerStatus, softApConfig

softApGetConfig

```
bool FishinoClass::softApGetConfig(char *&SSID, char *&pass, uint8_t &channel, bool &hidden);
```

Gets soft AP configuration data.

Parameters :

- **SSID** a char* variable that will hold retrieved SSID
- **pass** a char* variable that will hold retrieved password
- **channel** an uint8_t variable that will hold retrieved channel
- **hidden** a boolean variable that will hold retrieved hidden state

Return value :

Boolean, true on success, false otherwise

Notes :

SSID and pass will receive both 2 dynamically allocated buffers that MUST be freed by free() after usage.
See the example for details.

Example :

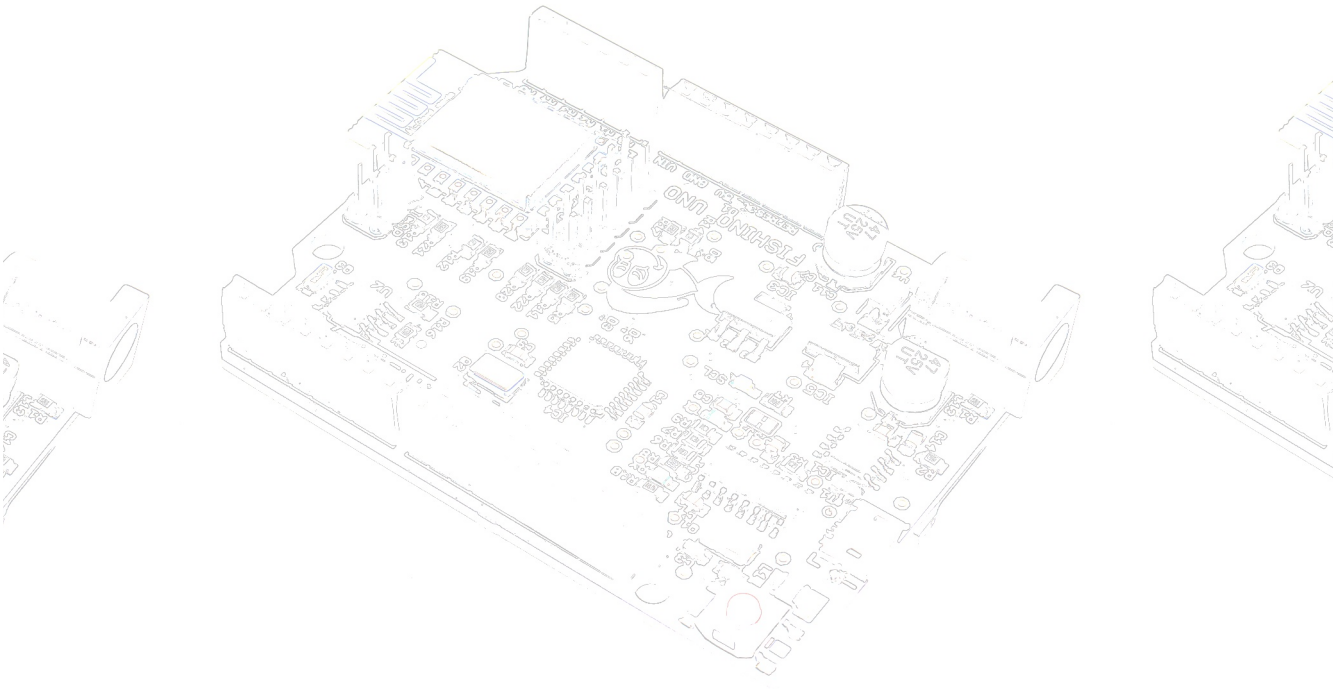
```
char *ssid, *pass;
uint8_t channel;
bool hidden;
if(Fishino.softApGetConfig(ssid, pass, channel, hidden))
{
      Serial.println("SoftAP configuration data");
      Serial.print("SSID : ");
      Serial.println(ssid);
      Serial.print("PASSWORD : ");
      Serial.println(pass);
      Serial.print("CHANNEL : ");
      Serial.println((int)channel);
      Serial.print("HIDDEN : ");
      Serial.println(hidden ? "true" : "false");
      // remember to free SSID and PASSWORD !!!!
      free(ssid);
      free(pass);
}
```

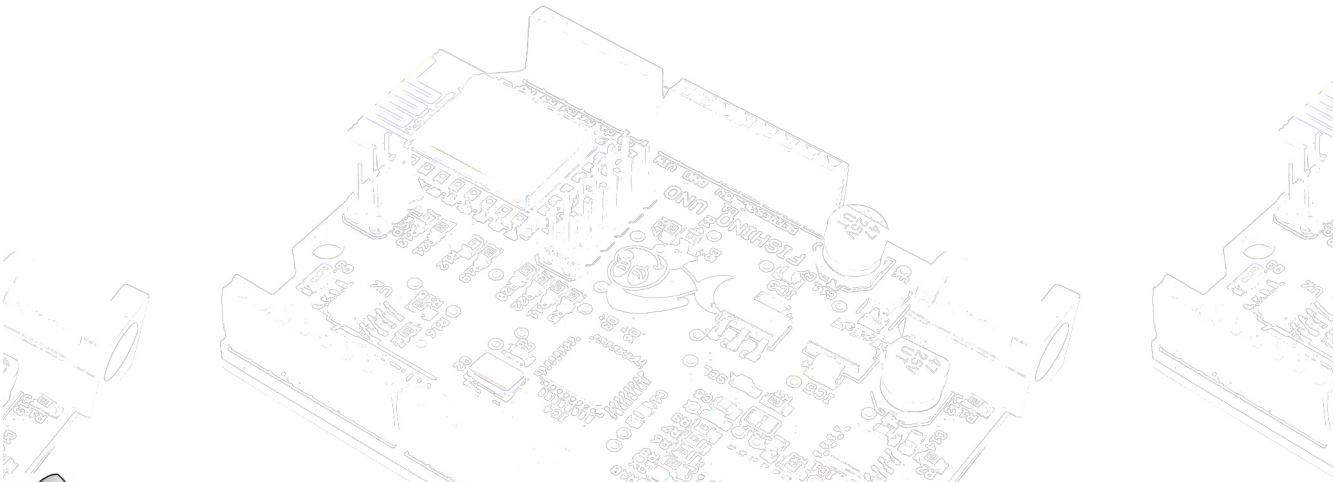
```
    else
          Serial.println("Failed to retrieve AP configuration data");
```

The example retrieves configuration data and prints it on serial port.

See also :

softApConfig, softApGetChannel, softApGetHidden, softApGetPassword, softApGetSSID, softApStartDHCPServer, softApStopDHCPServer, getSoftApDHCPServerStatus





softApGetHidden

```
bool FishinoClass::softApGetHidden(void);
```

Get hidden state of access point mode network.

Parameters :

none

Return value :

Boolean, **true** if SSID is hidden, **false** otherwise

Notes :

Example :

```
Serial.print("SSID is ");
if(Fishino.softApGetHidden())
Serial.println("HIDDEN");
else
Serial.println("NOT HIDDEN");
```

The example check if Fishino's network has hidden SSID and prints the result on serial port.

See also :

softApGetConfig, softApGetChannel, softApGetPassword, softApGetSSID, softApStartDHCPServer, softApStopDHCPServer, getSoftApDHCPServerStatus, softApConfig

softApGetPassword

```
char *FishinoClass::softApGetPassword(void);
```

The function retrieves softAP's password.

Parameters :

none

Return value :

A dynamically allocated buffer with the requested password on success, NULL on error.

Notes :

Retrieved password is dynamically allocated and MUST be freed by free().

Example :

```
char *pass = Fishino.softApGetPassword();
if(pass)
{
      Serial.print("Password is : ");
      Serial.println(pass);
      free(pass);
}
else
      Serial.println("Failed to retrieve password");
```

The example retrieves softAP's password and prints it on serial port.

See also :

softApGetConfig, softApGetHidden, softApGetChannel, softApGetSSID, softApStartDHCPServer, softApStopDHCPServer, getSoftApDHCPServerStatus, softApConfig

softApGetSSID

```
char *FishinoClass::softApGetSSID(void);
```

The function retrieves softAP's SSID.

Parameters :

none

Return value :

A dynamically allocated buffer with the requested SSID on success, NULL on error.

Notes :

Retrieved SSID is dynamically allocated and MUST be freed by free().

Example :

```
char *ssid = Fishino.softApGetSSID();
if(ssid)
{
      Serial.print("SSID is : ");
      Serial.println(ssid);
      free(ssid);
}
else
      Serial.println("Failed to retrieve SSID");
```

The example retrieves softAP's SSID and prints it on serial port.

See also :

softApGetConfig, softApGetHidden, softApGetChannel, softApGetPassword, softApStartDHCPServer, softApStopDHCPServer, getSoftApDHCPServerStatus, softApConfig

softApStartDHCPServer

```
bool FishinoClass::softApStartDHCPServer(void);
bool FishinoClass::softApStartDHCPServer(IPAddress startIP, IPAddress endIP);
```

The function starts softAP's internal DHCP server.
First form without parameters use all IP's available; the second one uses the specified IP range.

Parameters :

- **startIP** start IP of DHCP pool
- **endIP** end IP of DHCP pool

Return value :

Boolean, **true** on success, **false** otherwise

Notes :

Example :

```
IPAddress start(192.168.1.200);
IPAddress end(192.168.1.230);
Fishino.startDHCPServer(start, end);
```

The example starts a DHCP server with IP range from 192.168.1.200 to 192.168.1.230.

See also :

softApGetConfig, softApGetHidden, softApGetChannel, softApGetSSID, softApGetPassword, softApStartDHCPServer, softApStopDHCPServer, getSoftApDHCPServerStatus, softApConfig

softApStopDHCPServer

```
bool FishinoClass::softApStopDHCPServer(void);
```

The function stops internal softAP's DHCP server.

Parameters :

none

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
Fishino.softApStopDHCPServer();
```

The example stops the DHCP server.

See also :

softApGetConfig, softApGetHidden, softApGetChannel, softApGetSSID, softApGetPassword, softApStartDHCPServer, getSoftApDHCPServerStatus, softApConfig

SSID

```
const char* FishinoClass::SSID(void);
```

The function retrieves the SSID of currently associated access point.

Parameters :

none

Return value :

The required SSID on success, NULL on failure.
The value is stored into an internal buffer and must NOT be freed.

Notes :

Returned SSID belongs to an internal buffer and is valid only up to next scanNetworks() call.
The buffer must NOT be freed.

Example :

```
char *ssid = Fishino.SSID();
if(ssid)
{
      Serial.print("SSID is : ");
      Serial.println(ssid);
}
else
      Serial.println("Failed to get SSID");
```

The example retrieves the SSID and prints it on serial port.

See also :

setStaConfig, begin, joinAp, quitAp, disconnect

status

```
uint8_t FishinoClass::status();
```

Returns status of connection with an access point.
The value returned is one of the JOIN_STATUS enum.

Parameters :

none

Return value :

A value from JOIN_STATUS enum:

```
typedef enum
{
	STATION_IDLE,
	STATION_CONNECTING,
	STATION_WRONG_PASSWORD,
	STATION_NO_AP_FOUND,
	STATION_CONNECT_FAIL,
	STATION_GOT_IP,
	STATION_TIMEOUT,
	STATION_BAD_SSID,
} JOIN_STATUS;
```

Notes :

The connection is completed when the FishinoClass::status() function returns STATION_GOT_IP value.
STATION_IDLE means no connection in progress; STATION_CONNECTING means a connection in progress but still not completed.
Any other value is a connection error.

Example :

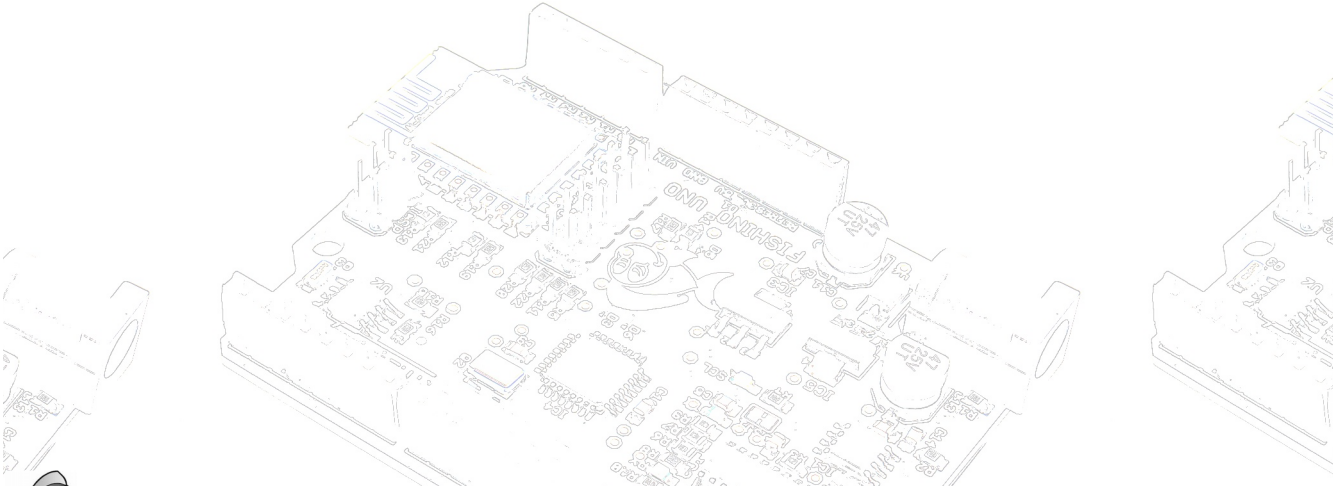
```
uint8_t stat = Fishino.status();
switch(stat)
{
	case STATION_IDLE :
		Serial.println("No connection in progress");
		break;
```

```
        case STATION_CONNECTING :
              Serial.println("Connection in progress");
              break;
        case STATION_WRONG_PASSWORD :
              Serial.println("Wrong password");
              break;
        case STATION_NO_AP_FOUND :
              Serial.println("Access point not found");
              break;
        case STATION_CONNECT_FAIL :
              Serial.println("Connection failed");
              break;
        case STATION_GOT_IP :
              Serial.println("Successfully connected to AP");
              break;
        case STATION_TIMEOUT :
              Serial.println("Timeout connecting to AP");
              break;
        case STATION_BAD_SSID :
              Serial.println("Bad SSID");
              break;
        default:
              Serial.println("Unknown connection error");
              break;
}
```

This example prints the status of connection to access point.

See also :

JOIN_STATUS



staStartDHCP

```
bool FishinoClass::staStartDHCP(void);
```

Starts station's DHCP client, requiring a dynamic IP to access point.

Parameters :

none

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
Fishino.staStartDHCP();
```

The example starts the DHCP client and requires a dynamic IP to access point.

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStopDHCP, getStaDHCPStatus, setDNS

staStopDHCP

```
bool FishinoClass::staStopDHCP(void);
```

Stops station's DHCP client, allowing static IP usage.

Parameters :

none

Return value :

Boolean, **true** on success, **false** otherwise.

Notes :

Example :

```
Fishino.staStopDHCP();
```

The example stops the DHCP client and allows static IP usage.

See also :

config, localIP, subnetMask, gatewayIP, macAddress, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, getStaDHCPStatus, setDNS

subnetMask

```
IPAddress FishinoClass::subnetMask();
```

Return the station subnet mask.

Parameters :

none

Return value :

Station subnet mask.

Notes :

Example :

```
IPAddress nm = Fishino.subnetMask();
Serial.print("Subnet mask is : ");
Serial.println(nm);
```

Prints station subnet mask on serial port.

See also :

config, localIP, gatewayIP, macAddress, setStaIP, setStaMAC, setStaGateway, setStaNetMask, staStartDHCP, staStopDHCP, getStaDHCPStatus, setDNS

WIFI_MODE

```
enum WIFI_MODE;
```

List of WiFi operating modes :

```
typedef enum
{
	NULL_MODE,
	STATION_MODE,
	SOFTAP_MODE,
	STATIONAP_MODE
} WIFI_MODE;
```

Notes :

Example :

```
uint8_t mode = Fishino.getMode();
Serial.print("Current operation mode is :");
switch(mode)
{
	case STATION_MODE:
		Serial.println("STATION");
		break;
	case SOFTAP_MODE:
		Serial.println("ACCESS POINT");
		break;
	case STATIONAP_MODE:
		Serial.println("STATION + ACCESS POINT");
		break;
	default:
		Serial.println("UNKNOWN");
		break;
}
```

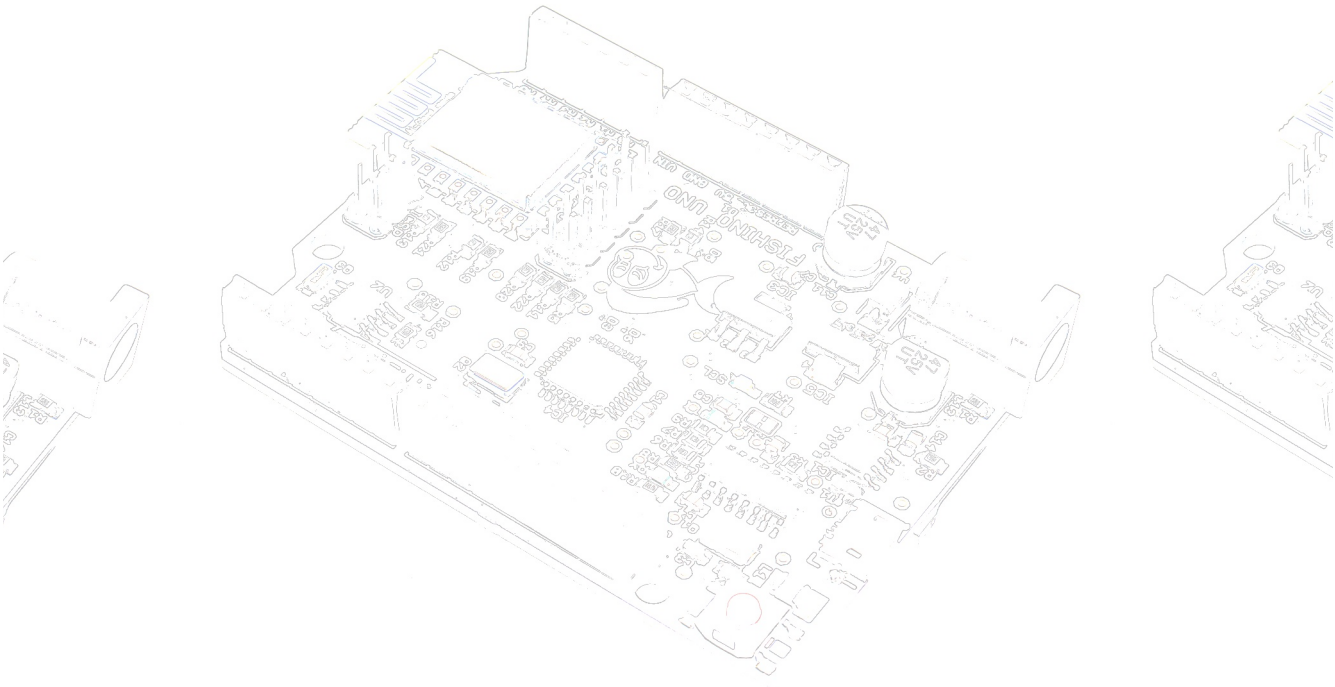
The example prints current operation mode on serial port.

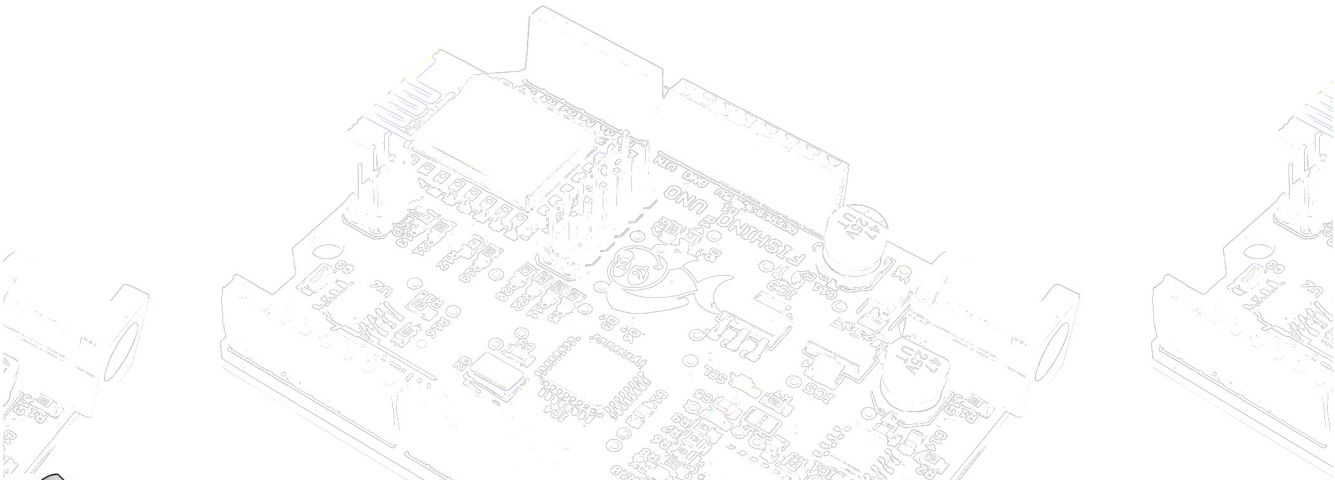
See also :

setMode, getMode

Class FishinoClient

FishinoClient provides TCP client features; it allows to share data with other TCP nodes.





available

```
virtual int available(void);
```

Get number of available bytes on client.

Parameters :

none

Return value :

Number of available bytes to read.

Notes :

Example :

```
FishinoClient client;
client.connect("www.fishino.it", 80);
client.println("GET /pub/WWW/TheProject.html HTTP/1.1");
client.println("Host: www.w3.org");
client.println();
while(client.available())
      Serial.print((char)client.read());
client.stop();
```

The example sends an HTTP GET request to a server, reads the answer and prints it on serial port.

See also :

connected, peek, write

connect

```
virtual int connect(IPAddress ip, uint16_t port);
virtual int connect(const char *host, uint16_t port);
```

Description

Parameters :

- **ip** the remote ip
- **host** the remote host
- **port** the remote port on which we want to connect

Return value :

An integer different from zero (true) if ok, zero on error.

Notes :

Example :

```
FishinoClient client;
client.connect("www.fishino.it", 80);
client.println("GET /pub/WWW/TheProject.html HTTP/1.1");
client.println("Host: www.w3.org");
client.println();
while(client.available())
      Serial.print((char)client.read());
client.stop();
```

The example sends an HTTP GET request to a server, reads the answer and prints it on serial port.

See also :

connected, status, stop

connected

```
virtual uint8_t connected(void);
virtual operator bool(void);
```

Check whether the client is connected to a remote host.

Parameters :

none

Return value :

An integer different from zero (true) if connected, zero (false) if not.

Notes :

Example :

```
FishinoClient client;
client.connect("www.fishino.it", 80);
if(client.connected())
      Serial.println("Client is connected to host");
else
      Serial.println("Client is not connected to host");
```

The example connects to a remote host and check connection result.

See also :

connect, status, stop

flush

```
virtual void flush(void);
```

Flush client's read and write buffers.

Parameters :

none

Return value :

none

Notes :

The behaviour is different from serial port's flush function : FishinoClient flush sends write buffer AND discards read buffer.

Example :

```
client.println("Some text");
client.flush();
```

The example sends some data to remote host and flushes the buffers, forcing immediate sending.

See also :

peek, read, write

getBufferedMode

```
bool getBufferedMode(void);
```

Gets buffered mode for tcp client

Parameters :

none

Return value :

Boolean, **true** if buffered mode is active, **false** otherwise.

Notes :

When buffering is enabled, data sent to remote host is cumulated in a local buffer for some (small) time, in order to speed up the send process. It's normally a desired behaviour, but shall be disabled if we want an immediate answer on each byte sent.

Example :

```
Serial.print("Client buffered mode is ");
if(client.getBufferedMode())
      Serial.println("active");
else
      Serial.println("disabled");
```

The example checks if buffered mode is active for client and prints result on serial port.

See also :

setBufferedMode, getNoDelay, setNoDelay

getForceCloseTime

```
uint32_t getForceCloseTime(void);
```

Gets inactivity timeout in milliseconds for tcp client

Parameters :

none

Return value :

A time in milliseconds, or zero if timeout is disabled.

Notes :

When force close time is active (time > 0) all clients that has no activities for the selected amount of time gets closed automatically. This avoids to have stray clients that consume resources, but can also close clients with low activity.

Example :

```
uint32_t tim = client. getForceCloseTime();
if(tim)
{
      Serial.print("Client force close time is ")
      Serial.print(tim);
      Serial.println(" mSec");
}
else
      Serial.println("Force close time is disabled for client");
```

The example reads force close time for client and prints result on serial port.

See also :

setForceCloseTime

getNoDelay

```
bool getNoDelay(void);
```

Gets nagle status for tcp client

Parameters :

none

Return value :

Boolean, **true** if Nagle algorithm is disabled, **false** if enabled.

Notes :

Nagle algorithm avoids flooding the network connection delaying the acknowledge answer on incoming packets; when NoDelay is true, the algorithm is disabled. This is usually the expected behaviour on embedded systems with low activity which shall get fast answers from hosts.

Example :

```
Serial.print("Nagle algorithm for client is ");
if(client.getNoDelay())
      Serial.println("disabled");
else
      Serial.println("enabled");
```

The example displays on serial port the status of Nagle algorithm for client.

See also :

setNoDelay

getSocket

```
uint8_t getSocket(void) const;
```

Get low-level socket descriptor for client.

Parameters :

none

Return value :

An uint8_t socket number, or 0xff if client is not connected.

Notes :

This shall not be needed by user code; it's mostly used internally.

Example :

```
Serial.print("Client socket is ");
Serial.println(client.getSocket(), HEX);
```

The example print client's socket on serial port.

See also :

connect, connected, stop

peek

```
virtual int peek(void);
```

Get next available byte from client (or -1 if none) without removing from input buffer, so next read() will find it.

Parameters :

none

Return value :

The byte that has been read, or -1 if none.

Notes :

Example :

```
int i = client.read();
if(i < 0)
      Serial.println("no bytes on client");
else
{
      Serial.print("Got ");
      Serial.print(i);
      Serial.println(" from client");
}
```

The example peeks a byte from client and displays result on serial port.

See also :

available, read

read

```
virtual int read(void);
virtual int read(uint8_t *buf, size_t size);
```

Description

Parameters :

- **buf** a buffer on which the read data will be placed (must be at least 'size' bytes big)
- **size** the number of bytes to read

Return value :

The first form, without parameters, will return the byte read, or -1 if none; second form will return the number of bytes actually read, which can be less than 'size'

Notes :

On second version, the buffer size MUST be at least 'size' bytes, or memory corruption may arise.

Example :

```
uint8_t buf[50];
size_t res = client.read(buf, 50);
Serial.print((int)res);
Serial.println(" bytes has been read from client");
```

The example try to read 50 bytes from client and displays on serial port the number of bytes actually read.

See also :

available, peek, write

setBufferedMode

```
bool setBufferedMode(bool b);
```

Sets buffered mode for tcp client

Parameters :

- **b** boolean, true to enable buffered mode, false otherwise

Return value :

Boolean, **true** on success, **false** on error.

Notes :

When buffering is enabled, data sent to remote host is cumulated in a local buffer for some (small) time, in order to speed up the send process. It's normally a desired behaviour, but shall be disabled if we want an immediate answer on each byte sent.

Example :

```
client.setBufferedMode(false);
```

The example disables buffering on client.

See also :

getBufferedMode

setForceCloseTime

```
bool setForceCloseTime(uint32_t tim);
```

Sets inactivity timeout for tcp client

Parameters :

- **tim** The requested timeout, or 0 to disable the timeout

Return value :

Boolean, **true** on success, **false** on error.

Notes :

When force close time is active (time > 0) all clients that has no activities for the selected amount of time gets closed automatically. This avoids to have stray clients that consume resources, but can also close clients with low activity.

Example :

```
client.setForceCloseTime(0);
```

The example disabled timeout on client.

See also :

getForceCloseTime

setNoDelay

```
bool setNoDelay(bool b);
```

Enable/disable nagle for tcp client

Parameters :

- **b** boolean, true to disable Nagle algorithm, false to enable it

Return value :

Boolean, **true** on success, **false** on error.

Notes :

Nagle algorithm avoids flooding the network connection delaying the acknowledge answer on incoming packets; when NoDelay is true, the algorithm is disabled. This is usually the expected behaviour on embedded systems with low activity which shall get fast answers from hosts.

Example :

```
client.setNoDelay(true);
```

The example disables Nagle algorithm on client.

See also :

getNoDelay

status

```
bool status(void);
```

Check whether the client is connected to a remote host.

Parameters :

none

Return value :

Boolean, **true** if connected, **false** if not.

Notes :

Example :

```
if(client.status())
      Serial.println("Client is connected");
else
      Serial.println("Client is not connected");
```

The example displays client's connection status on serial port.

See also :

connect, connected, stop

stop

```
virtual void stop(void);
```

Terminates client's connection to remote host.

Parameters :

none

Return value :

none

Notes :

Example :

```
client.stop();
```

The example closes client's connection.

See also :

connect, connected, status

write

```
virtual size_t write(uint8_t b);
virtual size_t write(const uint8_t *buf, size_t size);
```

Description

Parameters :

- **b** a byte to send
- **buf** a buffer (array of bytes) to send
- **size** the number of bytes to send

Return value :

The number of bytes actually sent.

Notes :

Example :

```
client.write("hello", strlen(hello));
```

The example sends "hello" string (without terminating zero) to remote host.

See also :

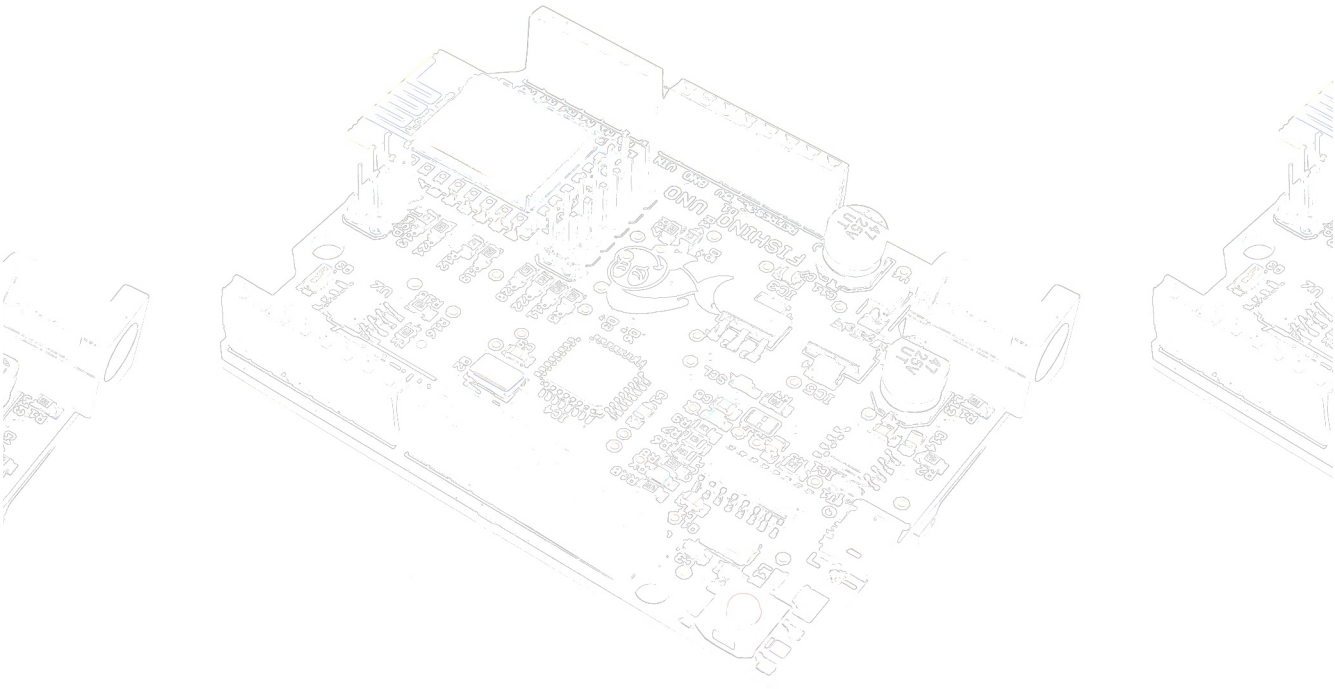
available, flush, peek, read

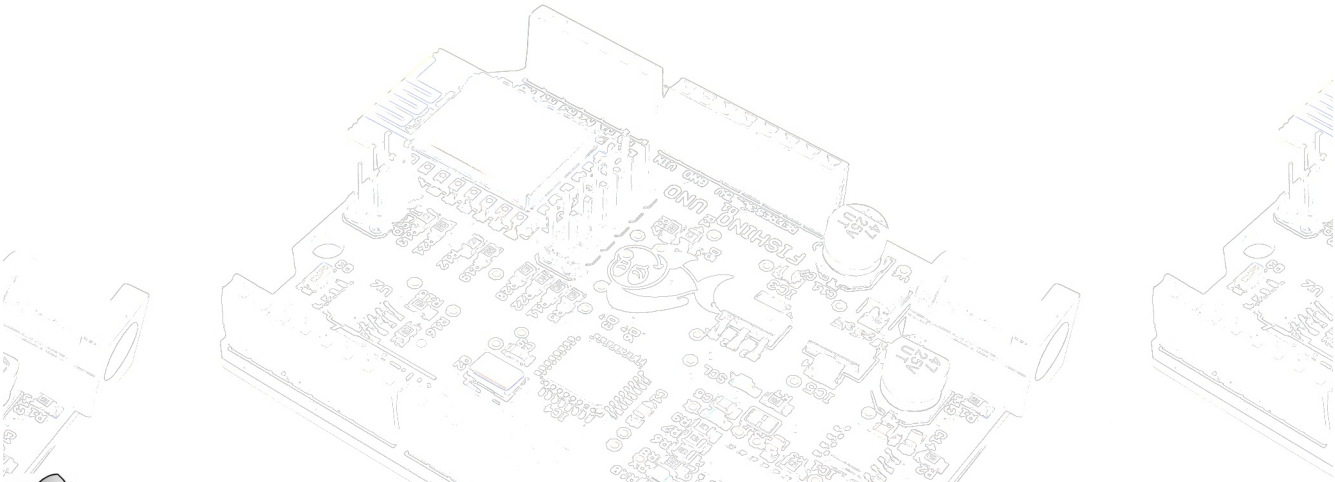
Class FishinoSecureClient

The FishinoSecureClient class implements TCP clients for secure (SSL/HTTPS) connections.

Only one SSL connection is allowed at once.

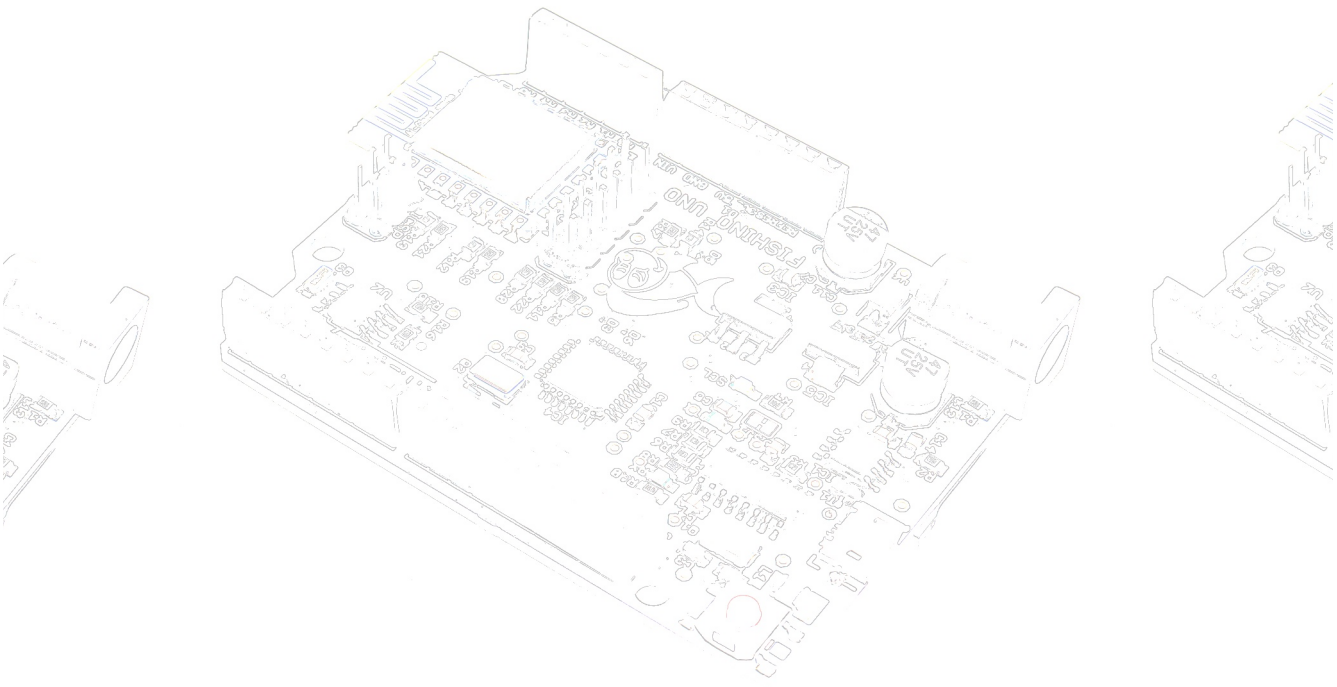
FishinoSecureClient class is derived from FishinoClient, so all the documentation of the latter is valid for it.

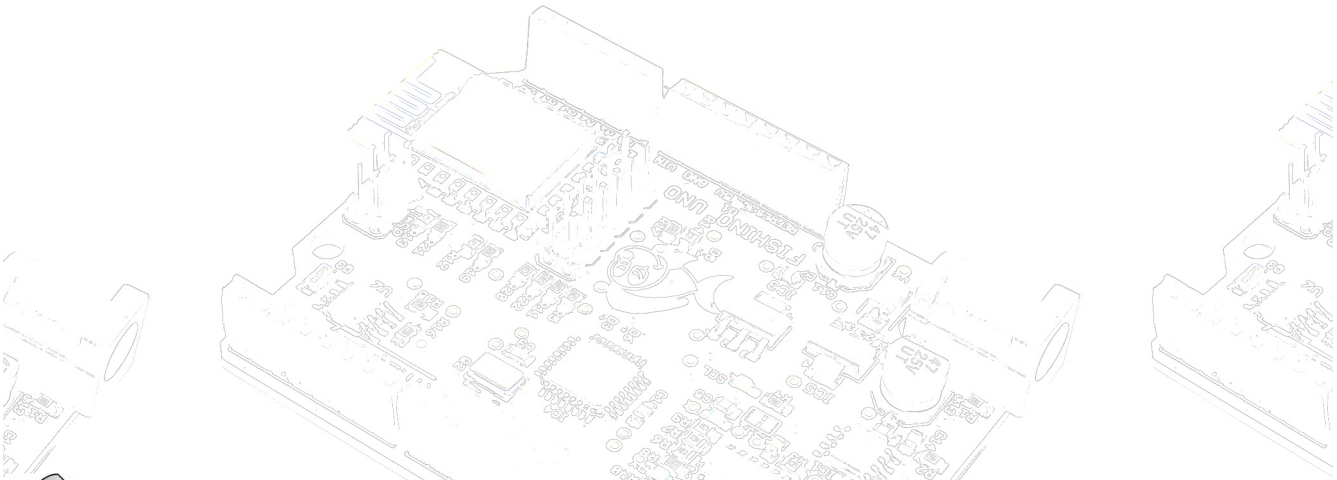




Class FishinoUDP

FishinoUDP class implements UDP clients/servers.





available

```
virtual int available();
```

Gets the number of bytes available for read in the current packet

Parameters :

none

Return value :

The number of available bytes.

Notes :

Example :

```
Serial.print("Packet has ");
Serial.print(udp.available());
Serial.println(" bytes available for reading");
```

The example prints the number of available bytes in current udp packet

See also :

parsePacket, peek, read, flush

begin

```
virtual uint8_t begin(uint16_t);
```

initialize, start listening on specified port. Returns 1 if successful, 0 if there are no sockets available to use

Parameters :

The local port on which we want to listen for incoming packets.

Return value :

A value different than 0 (true) on success, 0 (false) otherwise.

Notes :

Example :

```
// local port to listen on
unsigned int localPort = 2390;

// buffer to hold incoming packet
char packetBuffer[255];

// a string to send back
char  ReplyBuffer[] = "acknowledged";

// the UDP client/server
FishinoUDP Udp;

// starts listening on local port
Udp.begin(localPort);

while(true)
{
      // if there's data available, read a packet
      int packetSize = Udp.parsePacket();
      if (packetSize)
      {
            Serial << F("Received packet of size ");
            Serial << packetSize << "\n";
            Serial << F("From ");
```

```
            IPAddress remoteIp = Udp.remoteIP();
            Serial << remoteIp;
            Serial << F(", port ");
            Serial << Udp.remotePort() << "\n";

            // read the packet into packetBufffer
            int len = Udp.read(packetBuffer, 255);
            if (len > 0)
                  packetBuffer[len] = 0;

            Serial << F("Contents:\n");
            Serial.println(packetBuffer);

            // send a reply, to the IP address and port that sent us the
            // packet we received
            Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
            Udp.write(ReplyBuffer);
            Udp.endPacket();
      }
}
```

The example starts an UDP server on port 2390, waits for incoming packets; when a packet is received prints some info about it and sends a reply back to remote host.

See also :

stop, beginPacket, endPacket, read, write, parsePacket, remoteIP, remotePort

beginPacket

```
virtual int beginPacket(IPAddress ip, uint16_t port);
virtual int beginPacket(const char *host, uint16_t port);
```

Start building up a packet to send to the remote host specific in ip or host and port.

Parameters :

- **ip** the IP address of remote host
- **host** the remote host
- **port** the remote port

Return value :

1 (true) if successful, 0 (false) if there was a problem with the supplied IP address, host or port

Notes :

Example :

```
// NTP time stamp is in the first 48 bytes of the message
const int NTP_PACKET_SIZE = 48;

//buffer to hold incoming and outgoing packets
byte packetBuffer[NTP_PACKET_SIZE];

// A UDP instance to let us send and receive packets over UDP
FishinoUDP Udp;

// set all bytes in the buffer to 0
memset(packetBuffer, 0, NTP_PACKET_SIZE);

// Initialize values needed to form NTP request

// LI, Version, Mode
packetBuffer[0] = 0b11100011;

// Stratum, or type of clock
packetBuffer[1] = 0;

// Polling Interval
packetBuffer[2] = 6;
```

```
// Peer Clock Precision
packetBuffer[3] = 0xEC;

// 8 bytes of zero for Root Delay & Root Dispersion
packetBuffer[12]  = 49;
packetBuffer[13]  = 0x4E;
packetBuffer[14]  = 49;
packetBuffer[15]  = 52;

// all NTP fields have been given values, now
// you can send a packet requesting a timestamp:

// NTP requests are to port 123
// beginPacket() just opens the connection
Udp.beginPacket(IPAddress(129, 6, 15, 28), 123);

// fill UDP buffer with packet data
Udp.write(packetBuffer, NTP_PACKET_SIZE);

// ends and send the UDP packet
Udp.endPacket();
```

The example sends an UDP request to an NTP time server.

See also :

endPacket, write

endPacket

```
virtual int endPacket();
```

Finish off this packet and send it.

Parameters :

none

Return value :

1 (true) if the packet was sent successfully, 0 (false) if there was an error.

Notes :

Example :

```
// NTP time stamp is in the first 48 bytes of the message
const int NTP_PACKET_SIZE = 48;

//buffer to hold incoming and outgoing packets
byte packetBuffer[NTP_PACKET_SIZE];

// A UDP instance to let us send and receive packets over UDP
FishinoUDP Udp;

// set all bytes in the buffer to 0
memset(packetBuffer, 0, NTP_PACKET_SIZE);

// Initialize values needed to form NTP request

// LI, Version, Mode
packetBuffer[0] = 0b11100011;

// Stratum, or type of clock
packetBuffer[1] = 0;

// Polling Interval
packetBuffer[2] = 6;

// Peer Clock Precision
packetBuffer[3] = 0xEC;
```

```
// 8 bytes of zero for Root Delay & Root Dispersion
packetBuffer[12]  = 49;
packetBuffer[13]  = 0x4E;
packetBuffer[14]  = 49;
packetBuffer[15]  = 52;

// all NTP fields have been given values, now
// you can send a packet requesting a timestamp:

// NTP requests are to port 123
// beginPacket() just opens the connection
Udp.beginPacket(IPAddress(129, 6, 15, 28), 123);

// fill UDP buffer with packet data
Udp.write(packetBuffer, NTP_PACKET_SIZE);

// ends and send the UDP packet
Udp.endPacket();
```

The example sends an UDP request to an NTP time server.

See also :

something

flush

```
virtual void flush();
```

Finish reading the current packet skipping remaining data.

Parameters :

none

Return value :

none

Notes :

Example :

```
udp.read(buf, 50);
udp.flush();
```

The example reads 50 bytes from current packet and skip remaining ones.

See also :

peek, read

parsePacket

```
virtual int parsePacket();
```

Start processing the next available incoming packet.

Parameters :

none

Return value :

The size of the packet in bytes, or 0 if no packets are available

Notes :

Example :

```
// local port to listen on
unsigned int localPort = 2390;

// buffer to hold incoming packet
char packetBuffer[255];

// a string to send back
char  ReplyBuffer[] = "acknowledged";

// the UDP client/server
FishinoUDP Udp;

// starts listening on local port
Udp.begin(localPort);

while(true)
{
      // if there's data available, read a packet
      int packetSize = Udp.parsePacket();
      if (packetSize)
      {
            Serial << F("Received packet of size ");
            Serial << packetSize << "\n";
            Serial << F("From ");
            IPAddress remoteIp = Udp.remoteIP();
            Serial << remoteIp;
```

```
            Serial << F(", port ");
            Serial << Udp.remotePort() << "\n";

            // read the packet into packetBufffer
            int len = Udp.read(packetBuffer, 255);
            if (len > 0)
                  packetBuffer[len] = 0;

            Serial << F("Contents:\n");
            Serial.println(packetBuffer);

            // send a reply, to the IP address and port that sent us the
            // packet we received
            Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
            Udp.write(ReplyBuffer);
            Udp.endPacket();
      }
}
```

The example starts an UDP server on port 2390, waits for incoming packets; when a packet is received prints some info about it and sends a reply back to remote host.

See also :

begin, available, remoteIP, remotePort, peek, read

peek

```
virtual int peek();
```

Return the next byte from the current packet without moving on to the next byte

Parameters :

none

Return value :

The peeked byte.

Notes :

Example :

```
int b = udp.peek();
if(b >= 0)
{
      Serial.print("Next available byte is ");
      Serial.println(b, HEX);
}
else
      Serial.println("No more bytes available in packet");
```

The example looks if there's at least one byte for reading into current packet and displays it.

See also :

read, available

read

```
virtual int read(void);
virtual int read(unsigned char* buffer, size_t len);
virtual int read(char* buffer, size_t len);
```

Read bytes from the current packet

Parameters :

- buffer a buffer (bytes or chars) that will receive requested data
- len the requested data size

Return value :

The number of bytes actually read. Can be less than requested size.

Notes :

The first form, without parameters, returns an integer; if >= 0 it's the received byte, if < 0 is an error signaling that no bytes were available.

Example :

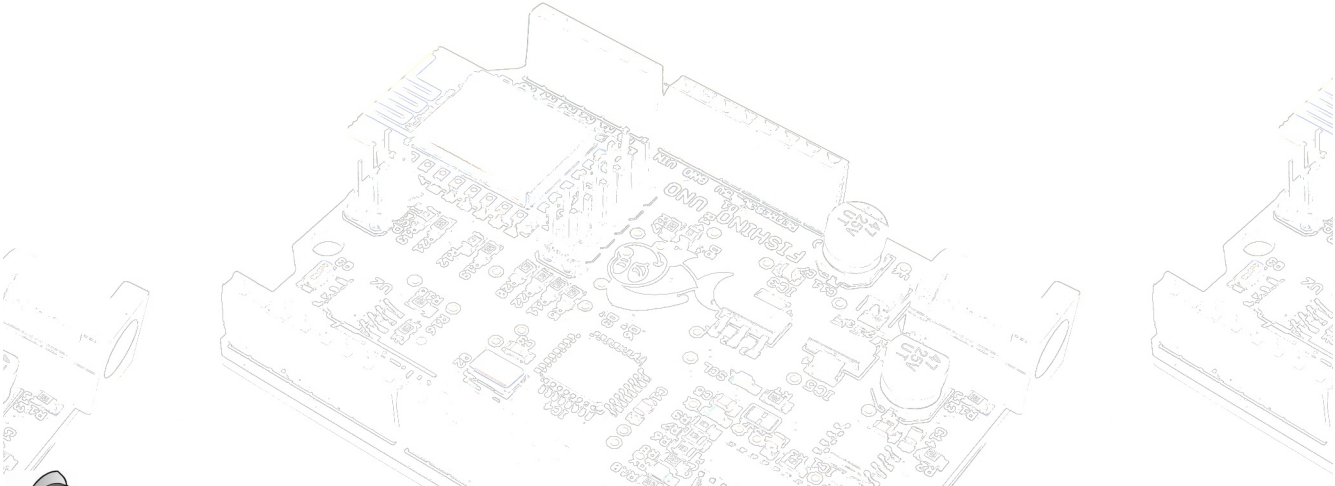
```
// local port to listen on
unsigned int localPort = 2390;

// buffer to hold incoming packet
char packetBuffer[255];

// a string to send back
char  ReplyBuffer[] = "acknowledged";

// the UDP client/server
FishinoUDP Udp;

// starts listening on local port
Udp.begin(localPort);

while(true)
{
      // if there's data available, read a packet
      int packetSize = Udp.parsePacket();
      if (packetSize)
```

```
    {
        Serial << F("Received packet of size ");
        Serial << packetSize << "\n";
        Serial << F("From ");
        IPAddress remoteIp = Udp.remoteIP();
        Serial << remoteIp;
        Serial << F(", port ");
        Serial << Udp.remotePort() << "\n";

        // read the packet into packetBufffer
        int len = Udp.read(packetBuffer, 255);
        if (len > 0)
          packetBuffer[len] = 0;

        Serial << F("Contents:\n");
        Serial.println(packetBuffer);

        // send a reply, to the IP address and port that sent us the
        // packet we received
        Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
        Udp.write(ReplyBuffer);
        Udp.endPacket();
    }
}
```

The example starts an UDP server on port 2390, waits for incoming packets; when a packet is received prints some info about it and sends a reply back to remote host.

See also :

available, peek



remoteIP

```
virtual IPAddress remoteIP();
```

Return the IP address of the host who sent the current incoming packet

Parameters :

none

Return value :

The IP address of remote peer.

Notes :

Example :

```
// local port to listen on
unsigned int localPort = 2390;

// buffer to hold incoming packet
char packetBuffer[255];

// a string to send back
char  ReplyBuffer[] = "acknowledged";

// the UDP client/server
FishinoUDP Udp;

// starts listening on local port
Udp.begin(localPort);

while(true)
{
      // if there's data available, read a packet
      int packetSize = Udp.parsePacket();
      if (packetSize)
      {
            Serial << F("Received packet of size ");
            Serial << packetSize << "\n";
            Serial << F("From ");
            IPAddress remoteIp = Udp.remoteIP();
            Serial << remoteIp;
```

```
            Serial << F(", port ");
            Serial << Udp.remotePort() << "\n";

            // read the packet into packetBufffer
            int len = Udp.read(packetBuffer, 255);
            if (len > 0)
                  packetBuffer[len] = 0;

            Serial << F("Contents:\n");
            Serial.println(packetBuffer);

            // send a reply, to the IP address and port that sent us the
            // packet we received
            Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
            Udp.write(ReplyBuffer);
            Udp.endPacket();
      }
}
```

The example starts an UDP server on port 2390, waits for incoming packets; when a packet is received prints some info about it and sends a reply back to remote host.

See also :

remotePort, begin, beginPacket, endPacket, read, write

remotePort

```
virtual uint16_t remotePort();
```

Return the port of the host who sent the current incoming packet

Parameters :

none

Return value :

The port of remote peer.

Notes :

Example :

```
// local port to listen on
unsigned int localPort = 2390;

// buffer to hold incoming packet
char packetBuffer[255];

// a string to send back
char  ReplyBuffer[] = "acknowledged";

// the UDP client/server
FishinoUDP Udp;

// starts listening on local port
Udp.begin(localPort);

while(true)
{
      // if there's data available, read a packet
      int packetSize = Udp.parsePacket();
      if (packetSize)
      {
            Serial << F("Received packet of size ");
            Serial << packetSize << "\n";
            Serial << F("From ");
            IPAddress remoteIp = Udp.remoteIP();
            Serial << remoteIp;
```

```
            Serial << F(", port ");
            Serial << Udp.remotePort() << "\n";

            // read the packet into packetBufffer
            int len = Udp.read(packetBuffer, 255);
            if (len > 0)
                  packetBuffer[len] = 0;

            Serial << F("Contents:\n");
            Serial.println(packetBuffer);

            // send a reply, to the IP address and port that sent us the
            // packet we received
            Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
            Udp.write(ReplyBuffer);
            Udp.endPacket();
      }
}
```

The example starts an UDP server on port 2390, waits for incoming packets; when a packet is received prints some info about it and sends a reply back to remote host.

See also :

remoteIP, begin, beginPacket, endPacket, read, write

stop

```
virtual void stop();
```

Finish with the UDP socket

Parameters :

none

Return value :

none

Notes :

Example :

```
udp.stop();
```

The example stops the UDP server.

See also :

begin

write

```
virtual size_t write(uint8_t b);
virtual size_t write(const uint8_t *buffer, size_t size);
```

Write bytes into current packet

Parameters :

- **b** a single byte to write into the packet
- **buffer** a buffer of bytes to write into the packet
- **size** the number of bytes to write into the packet

Return value :

The number of bytes actually written into the packet. Can be less than size.

Notes :

Example :

```
// NTP time stamp is in the first 48 bytes of the message
const int NTP_PACKET_SIZE = 48;

//buffer to hold incoming and outgoing packets
byte packetBuffer[NTP_PACKET_SIZE];

// A UDP instance to let us send and receive packets over UDP
FishinoUDP Udp;

// set all bytes in the buffer to 0
memset(packetBuffer, 0, NTP_PACKET_SIZE);

// Initialize values needed to form NTP request

// LI, Version, Mode
packetBuffer[0] = 0b11100011;

// Stratum, or type of clock
packetBuffer[1] = 0;

// Polling Interval
packetBuffer[2] = 6;
```

```
// Peer Clock Precision
packetBuffer[3] = 0xEC;

// 8 bytes of zero for Root Delay & Root Dispersion
packetBuffer[12]  = 49;
packetBuffer[13]  = 0x4E;
packetBuffer[14]  = 49;
packetBuffer[15]  = 52;

// all NTP fields have been given values, now
// you can send a packet requesting a timestamp:

// NTP requests are to port 123
// beginPacket() just opens the connection
Udp.beginPacket(IPAddress(129, 6, 15, 28), 123);

// fill UDP buffer with packet data
Udp.write(packetBuffer, NTP_PACKET_SIZE);

// ends and send the UDP packet
Udp.endPacket();
```

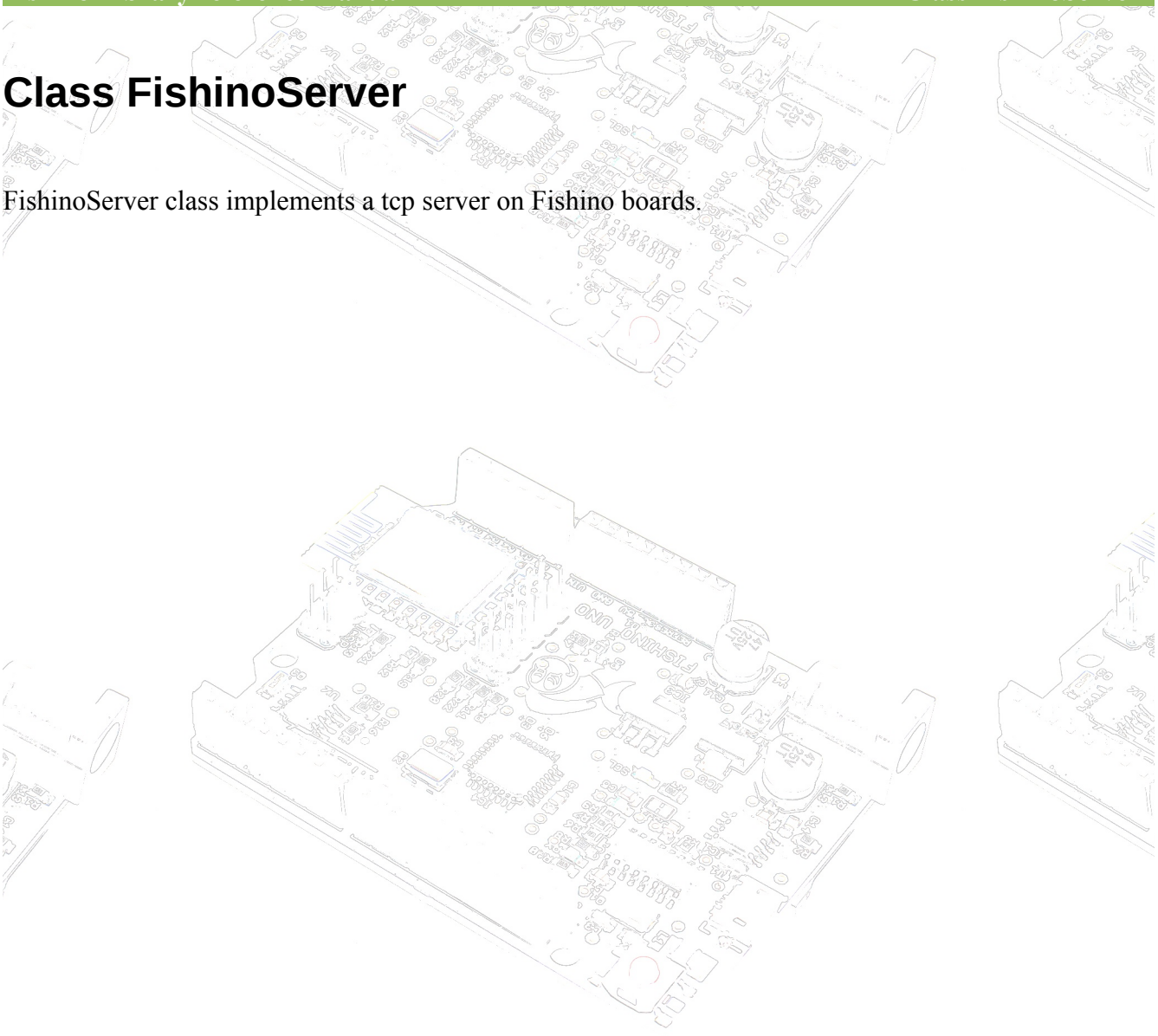
The example sends an UDP request to an NTP time server.

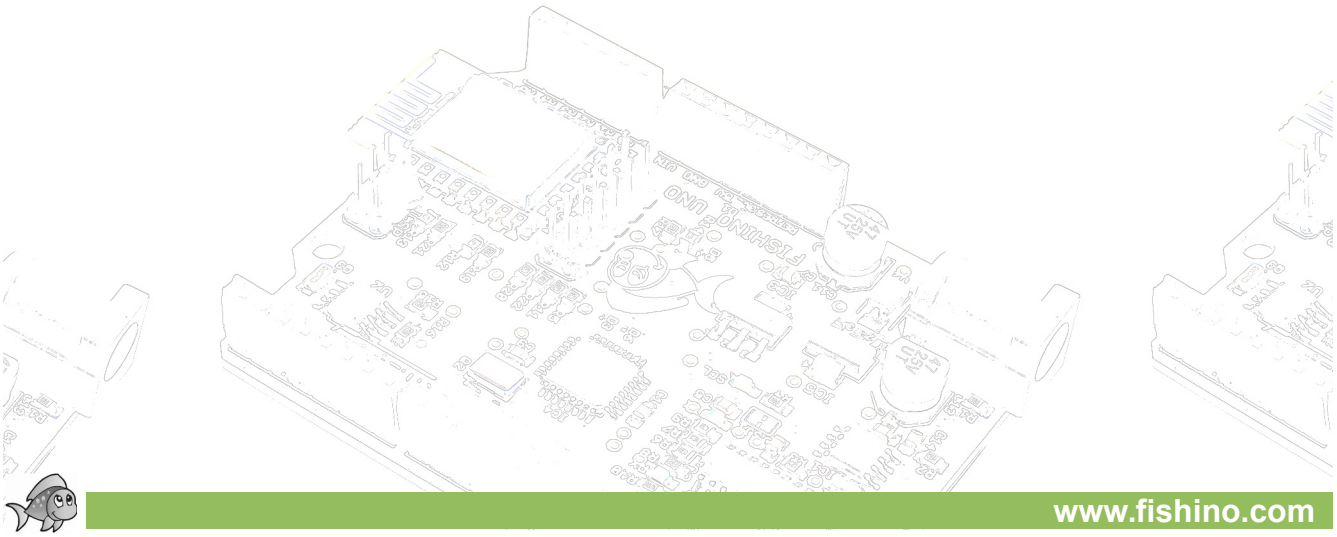
See also :

beginPacket, endPacket

Class FishinoServer

FishinoServer class implements a tcp server on Fishino boards.





available

```
FishinoClient available();
```

Get client for next incoming available connection. Can be a non-connected client if no incoming connections are available.

Parameters :

none

Return value :

A FishinoClient object

Notes :

Received client can be not connected, if no incoming connection is available. It shall be tested with boolean operator or with connected() function which is the same. See example for details.

Example :

```
FishinoClient client = server.available();
if(client)
      Serial.println("Got a client!");
```

The example checks if there's an incoming connection and, if yes, displays a message on serial port.

See also :

hasClients

begin

```
virtual void begin(void);
```

Starts listening for incoming connections.

Parameters :

none

Return value :

none

Notes :

Example :

```
FishinoServer server(3333);
server.begin();
```

Starts a server listening on port 3333.

See also :

close, stop

close

```
virtual void close(void);
```

Stops listening for incoming connections.

Parameters :

none

Return value :

none

Notes :

Example :

```
server.close();
```

Stop listening for incoming connections

See also :

begin, stop

getBufferedMode

```
bool getBufferedMode(void);
```

Get server buffered mode for incoming clients.

Parameters :

none

Return value :

A boolean value, **true** if buffered mode is active, **false** otherwise

Notes :

See FishinoClient.getBufferedMode(). The buffered mode is managed automatically for incoming connections.

Example :

```
if(servef.getBufferedMode())
      Serial.println("Buffered mode is active for incoming connections");
else
      Serial.println("Buffered mode is active for incoming connections");
```

The example checks if buffered mode is active for incoming connections and displays the result on serial port.

See also :

setBufferedMode, FishinoClient.getBufferedMode, FishinoClient.setBufferedMode

getClientsForceCloseTime

```
uint32_t getClientsForceCloseTime(void);
```

Get server clients timeout.

Parameters :

none

Return value :

The active clients timeout value, or 0 if no timeout.

Notes :

Timeout on clients is useful to avoid to "forget" connected clients which consume resources. You should be careful on the value of timeout, because clients with low activity could be closed automatically even if still needed.

Example :

```
uint32_t tim = server.getClientsForceCloseTime();
if(tim)
{
      Serial.print("Clients timeout is ");
      Serial.print(tim);
      Serial.println(" mSec");
}
else
      Serial.println("No clients timeout");
```

The example checks if a timeout has been set for incoming connections and shows result on serial port.

See also :

setClientsForceCloseTime

getMaxClients

```
uint8_t getMaxClients(void);
```

Get max number of server clients.

Parameters :

none

Return value :

The number of allowed incoming tcp connections at once.

Notes :

Because of limited resources, avoid to set a big number of allowed incoming connections. The number is anyways limited by firmware design to a value around 5.
Connections exceeding the allowed ones are simply ignored.

Example :

```
Serial.print("Number of allowed incoming connections is ");
Serial.println(server.getMaxClients());
```

The example displays the number of allowed clients on serial port.

See also :

setMaxClients

getNoDelay

```
virtual bool getNoDelay(void);
```

Gets nagle status for incoming tcp clients

Parameters :

none

Return value :

Boolean, **true** if Nagle algorithm is disabled, **false** if enabled.

Notes :

Nagle algorithm avoids flooding the network connection delaying the acknowledge answer on incoming packets; when NoDelay is true, the algorithm is disabled. This is usually the expected behaviour on embedded systems with low activity which shall get fast answers from hosts.

Example :

```
Serial.print("Nagle algoritm is ");
if(server.getNoDelay())
      Serial.print("disabled");
else
      Serial.print("enabled");
Serial.println(" for incoming connections");
```

The example displays status of Nagle algorithm for incoming connections on serial port.

See also :

setNoDelay

hasClients

```
virtual bool hasClients();
```

Check if there are waiting incoming connections on server.

Parameters :

none

Return value :

A boolean value, **true** if server has incoming connections, **false** otherwise

Notes :

Example :

```
if(server.hasClients())
      Serial.println("Clients are available!");
```

The example prints a message on serial port if some incoming connection is available.

See also :

available

setBufferedMode

```
bool setBufferedMode(bool b);
```

Get server buffered mode.

Parameters :

- **b** boolean, **true** to enable buffered mode for clients, **false** to disable it

Return value :

A boolean value, **true** on success, **false** otherwise

Notes :

See FishinoClient.setBufferedMode(). The buffered mode is managed automatically for incoming connections.

Example :

```
server.setBufferedMode(true);
```

The example enables buffered mode for server's clients.

See also :

getBufferedMode, FishinoClient.getBufferedMode, FishinoClient.setBufferedMode

setClientsForceCloseTime

```
bool setClientsForceCloseTime(uint32_t tim);
```

Set server clients timeout.

Parameters :

- **tim** the requested timeout value, or 0 to disable timeout

Return value :

A boolean value, **true** on success, **false** otherwise

Notes :

Timeout on clients is useful to avoid to "forget" connected clients which consume resources. You should be careful on the value of timeout, because clients with low activity could be closed automatically even if still needed.

Example :

```
server.setClientsForceCloseTime(3000);
```

Sets 3 seconds clients timeout. Each client will be automatically closed after 3 seconds of inactivity.

See also :

getClientsForceCloseTime

setMaxClients

```
bool setMaxClients(uint8_t n);
```

Set max number of server clients.

Parameters :

- **n** the number of allowed incoming connections at once

Return value :

A boolean value, **true** on success, **false** otherwise

Notes :

Because of limited resources, avoid to set a big number of allowed incoming connections. The number is anyways limited by firmware design to a value around 5.
Connections exceeding the allowed ones are simply ignored.

Example :

```
server.setMaxClients(3);
```

The example limits the number of incoming connections to 3.

See also :

getMaxClients

setNoDelay

```
virtual void setNoDelay(bool n);
```

Enable/disable nagle for incoming tcp clients.

Parameters :

- **n** boolean, true to disable Nagle algorithm, false to enable it

Return value :

A boolean value, **true** on success, **false** otherwise

Notes :

Nagle algorithm avoids flooding the network connection delaying the acknowledge answer on incoming packets; when NoDelay is true, the algorithm is disabled. This is usually the expected behaviour on embedded systems with low activity which shall get fast answers from hosts.

Example :

```
server.setNoDelay(true);
```

The example disables Nagle algorithm for incoming connections.

See also :

getNoDelay

stop

```
virtual void stop(void);
```

Stops listening for incoming connections.

Parameters :

none

Return value :

none

Notes :

Example :

```
server.stop();
```

Stop listening for incoming connections

See also :

begin, close

write

```
virtual size_t write(uint8_t b);
virtual size_t write(const uint8_t *buf, size_t size);
```

Send a byte or a buffer of bytes to ALL connected clients at once.

Parameters :

- **b** a single byte to send
- **buf** a buffer of data to send
- **size** the size of data to send

Return value :

The number of bytes sent.

Notes :

Example :

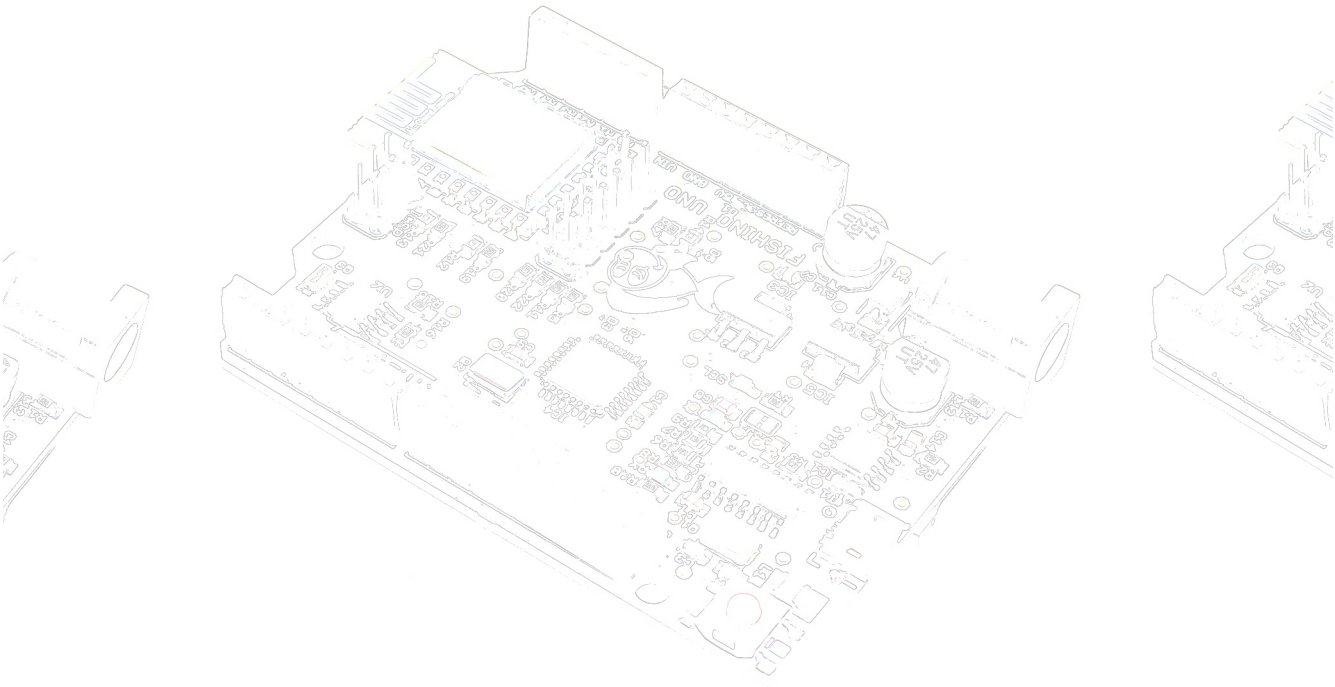
```
server.write("hello", strlen(hello));
```

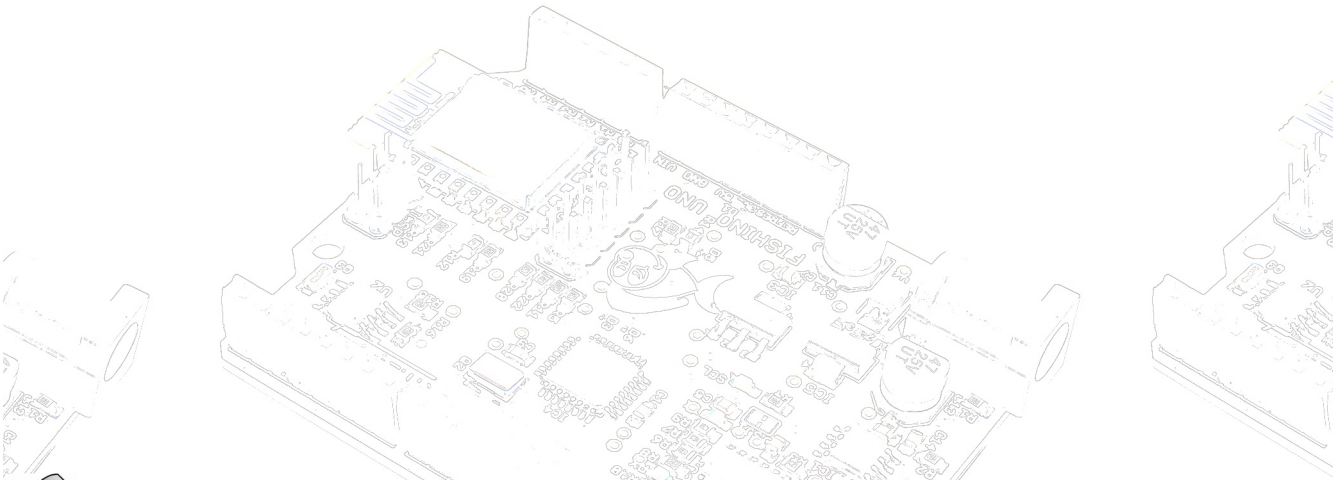
The example sends the string "hello" (without null terminator) to all connected clients.

See also :

Class FishinoSerial

FishinoSerial class manages the serial port on ESPCONN connector on 8 bit Fishino boards





available

```
virtual int available(void);
```

Get number of available bytes for reading.

Parameters :

none

Return value :

The number of available bytes on serial read buffer.

Notes :

Example :

```
while(FishinoSerial.available())
      Serial.print(FishinoSerial.read());
```

The example print incoming data on WiFi module's serial port to standard serial port.

See also :

peek, read, write

begin

```
void begin(uint32_t baud);
```

Open serial port.

Parameters :

- **baud** the requested serial port speed

Return value :

none

Notes :

Example :

```
FishinoSerial.begin(115200);
```

The example starts the serial port with a speed of 115200 bauds.

See also :

end

end

```
void end(void);
```

Close serial port.

Parameters :

none

Return value :

none

Notes :

Example :

```
FishinoSerial.end();
```

The example closes WiFi module's serial port, freeing the used I/O pins.

See also :

begin

flush

```
virtual void flush(void);
```

Flush serial port.

Parameters :

none

Return value :

none

Notes :

Example :

```
FishinoSerial.flush();
```

The example flushes serial port's write buffer, sending data immediately on serial lines.

See also :

write

peek

```
virtual int peek(void);
```

Peek a byte from serial port without actually removing from input buffer.

Parameters :

none

Return value :

The peeked byte, or a negative value if no bytes were available.

Notes :

Example :

```
int i = FishinoSerial.peek();
if(i >= 0)
{
      Serial.print("Got byte ");
      Serial.print(i, HEX);
      Serial.println(" from WiFi serial port");
}
else
      Serial.println("No bytes available from WiFi serial port");
```

The example checks if there is an incoming byte on WiFi module's serial port and print it.

See also :

available, read

read

```
virtual int read(void);
```

Read a byte from WiFi module's serial port.

Parameters :

none

Return value :

A byte of data read from serial port, or a negative value if none.

Notes :

Example :

```
int i = FishinoSerial.read();
if(i >= 0)
{
      Serial.print("Got byte ");
      Serial.print(i, HEX);
      Serial.println(" from WiFi serial port");
}
else
      Serial.println("No bytes available from WiFi serial port");
```

The example reads an incoming byte on WiFi module's serial port and print it.

See also :

available, peek

write

```
virtual size_t write(uint8_t b);
inline size_t write(unsigned long n);
inline size_t write(long n);
inline size_t write(unsigned int n);
inline size_t write(int n);
```

Write data on WiFi module's serial port.

Parameters :

- **b** a byte of data to be written
- n other kind of data to be written

Return value :

The number of bytes actually written on serial port.

Notes :

Returned value is the number of bytes written, not elements. For example, writing a 32 bit integer returns a value of 4.

Example :

```
FishinoSerial.write((uint8_t)0x05);
```

The example write one byte of data to WiFi module's serial port.

See also :

available, read, peek, flush

Index