**Idea Behind this Project:**

<https://forum.arduino.cc/index.php?topic=74260.0>

**WL\_IDLE\_STATUS:**

It is a temporary status assigned when WiFi.begin() is called and remains active until the number of attempts expires (resulting in WL\_CONNECT\_FAILED) or a connection is established (resulting in WL\_CONNECTED);

<https://www.arduino.cc/en/Reference/WiFiStatus>

**Difference Between Client and Server:**

In simplest form, a server is a connection point for several clients , that will handle their requests. A client is software that (usually) connects to the server to perform actions. The client provide a user interface that allows users to carry out actions.

<http://www.webopedia.com/TERM/S/SSID.html>

**client.available: Description**

Returns the number of bytes available for reading (that is, the amount of data that has been written to the client by the server it is connected to).available() inherits from the Stream utility class.

Syntax: client.available()

<https://www.arduino.cc/en/Reference/WiFiServerAvailable>

**client.read:read()**

Read the next byte received from the server the client is connected to (after the last call to read()).

read() inherits from the Stream utility class.

Syntax

client.read()

Returns

The next byte (or character), or -1 if none is available.

<https://www.arduino.cc/en/Reference/WiFiClientRead>

**client.connected():**

Description

Whether or not the client is connected. Note that a client is considered connected if the connection has been closed but there is still unread data.

Syntax

client.connected()

Returns

Returns true if the client is connected, false if not.

<https://www.arduino.cc/en/Reference/WiFiClientConnected>

**Client.print:**

Description

Print data to the server that a client is connected to. Prints numbers as a sequence of digits, each an ASCII character (e.g. the number 123 is sent as the three characters '1', '2', '3').

Syntax

client.print(data)

client.print(data, BASE)

Parameters

data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers:, DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Returns

byte : returns the number of bytes written, though reading that number is optional

<https://www.arduino.cc/en/Reference/WiFiClientPrint>

**millis():**

Description

Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Returns

Number of milliseconds since the program started (unsigned long)

<https://www.arduino.cc/en/reference/millis>

**client.connect: Description**

Connect to the IP address and port specified in the constructor. The return value indicates success or failure. connect() also supports DNS lookups when using a domain name (ex:google.com).

Syntax

client.connect(ip, port)

client.connect(URL, port)

Parameters

ip: the IP address that the client will connect to (array of 4 bytes)

URL: the domain name the client will connect to (string, ex.:"arduino.cc")

port: the port that the client will connect to (int)

Returns

Returns true if the connection succeeds, false if not.

<https://www.arduino.cc/en/Reference/WiFiClientConnect>

**pulseIn() :**Description

Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or 0 if no complete pulse was received within the timeout.

The timing of this function has been determined empirically and will probably show errors in shorter pulses. Works on pulses from 10 microseconds to 3 minutes in length. Please also note that if the pin is already high when the function is called, it will wait for the pin to go LOW and then HIGH before it starts counting. This routine can be used only if interrupts are activated. Furthermore the highest resolution is obtained with short intervals.

**Syntax**

pulseIn(pin, value)

pulseIn(pin, value, timeout)

**Parameters**

pin: the number of the pin on which you want to read the pulse. (int)

value: type of pulse to read: either HIGH or LOW. (int)

timeout (optional): the number of microseconds to wait for the pulse to be completed: the function returns 0 if no complete pulse was received within the timeout. Default is one second (unsigned long).

**Returns**

the length of the pulse (in microseconds) or 0 if no pulse is completed before the timeout (unsigned long)

<https://www.arduino.cc/en/Reference/pulseIn>

Important Notes:

<https://www.arduino.cc/en/Reference/WiFiBSSID>

<https://create.arduino.cc/projecthub/hgpt/wireless-weather-station-arduiono-esp8266-thingspeak-2b6d4a>

Thingspeak Data Uploader: Best Explaination

<https://www.arduino.cc/en/Tutorial/WiFi101ThingSpeakDataUploader>

uploading Sensor Data to Thingspeak: TO print Sinusoidally <https://www.arduino.cc/en/Tutorial/WiFi101ThingSpeakDataUploader>

Thingspeak Community: Examples to read the values from Thingspeak

<http://community.thingspeak.com/tutorials/arduino/send-data-to-thingspeak-with-arduino/>

advantages of using Galelio Gen 2 over Arduinos

<https://www.arduino.cc/en/ArduinoCertified/IntelGalileoGen2#toc2>

Galelio Data Sheet:

<https://www.arduino.cc/en/ArduinoCertified/IntelGalileoGen2>

WaterLevel using Transistor circuit Project:

<http://www.electroschematics.com/9964/arduino-water-level-indicator-controller/>

Water Level Controller Using Ultransonic Sensor:

<https://circuitdigest.com/microcontroller-projects/water-level-indicator-project-using-arduino>

Water Level Controller Using Ultransonic Sensor: **With Code**

<http://www.labviewarduino.in/2014/12/automatic-water-level-controller-using.html>

Putty connection

<http://www.instructables.com/file/FDFNUESIB21TF9A/>