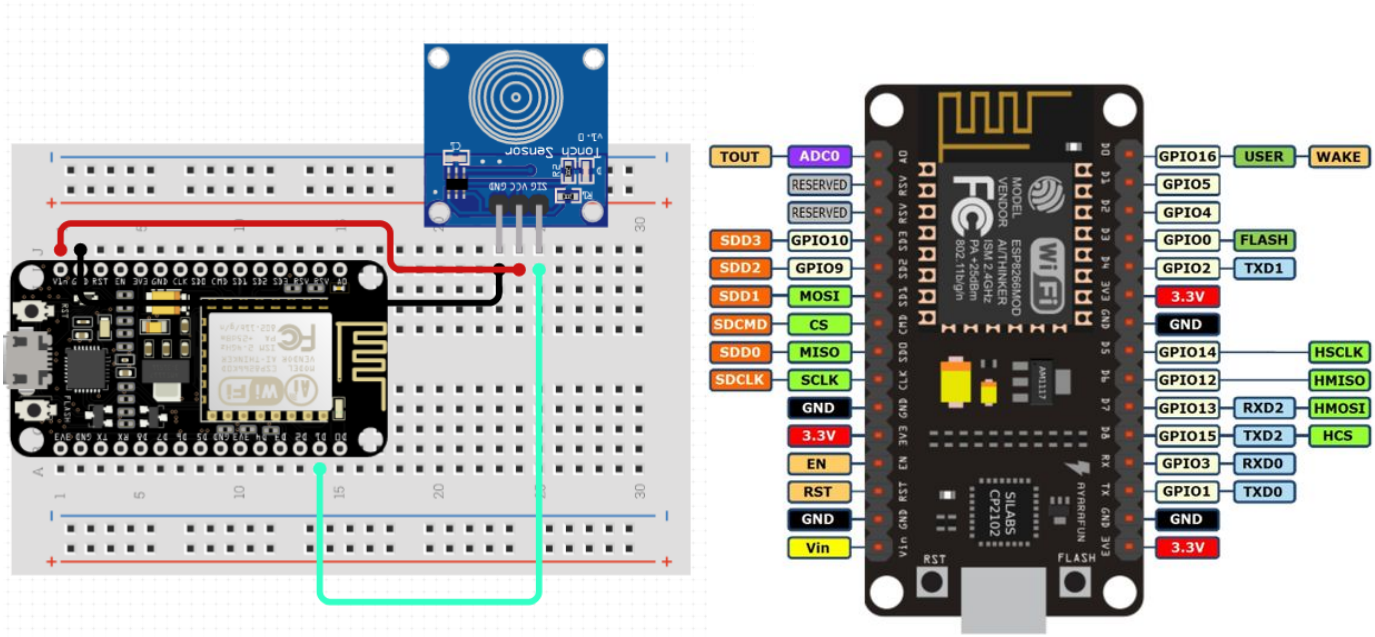
# IoT Using ESP8266 (Real-Time Monitoring Sensors Data)

**Sensors in IoT:** Many IoT based applications require some sensors to read the external data for monitoring them and sometimes for smart automation of services based on sensors. There are bunch of sensors in market with different purposes. Most commonly being used sensors include, IR sensor to detect obstacle, ultrasonic sensor to measure distance, touch sensor to detect proximity and so on. Sensors in market are of different scales, some are for industries and some for hobbyist to develop some interesting projects. Hobbyist sensors can also used to make some practical and useful applications and they work pretty good enough.


**Touch Sensor:** We will be using TTP223B capacitive touch sensor module which is cost-effective, simple to use and offers good enough accuracy. It has 3 pins; VCC (voltage), SIG (signal pin) and GND (ground) as shown in below figure. It gives 1 when touched else 0 at SIG pin.

**Connecting to ESP8266:** Connections are quite simple as we need to connect only 3 pins as below given in steps and figure (made on circuit.io) from ESP8266 to touch sensor.

- VU to VCC.
- GND to GND.
- SIG to D1 (GPIO5).



**Displaying real-time data over HTML:** Now we are interested in monitoring this real-time data on HTML with the help of ESP8266 client. We will first read the data from sensor and then by connecting ESP8266 to the WiFi we will display data on HTML page accessible with an IP provided by network.

## Code:

```
#include <ESP8266WiFi.h>
String ssid = "CRAIB-LAB";
String password = "********";
int TSP = 5;
WiFiServer server(80);
void setup() {
Serial.begin(115200);
pinMode(TSP, INPUT);
// Connect to Wi-Fi network
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to WiFi...");
}
// Start the server
server.begin();
Serial.println("Server started");
Serial.print("IP address:\t");
Serial.println(WiFi.localIP());
}
void loop() {
// Wait for a client to connect
WiFiClient client = server.available();
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("");
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<h3> Real-Time Touch Sensor data </h3>");
client.println("<h3> with ESP8266 over HTML </h3>");
client.println("Touch Sensor Reading :");
client.print(digitalRead(TSP));
client.print("<script> function autoRefresh() {window.location = window.location.href; } setInterval('autoRefresh()',
1500); </script>");
client.println("</html>");
}
```

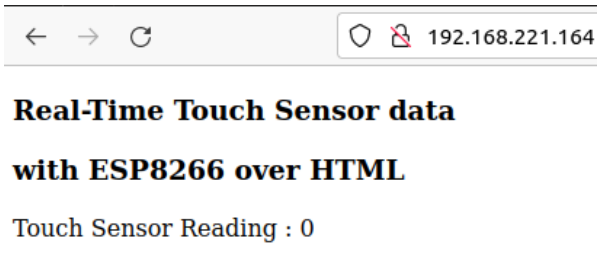
## Code Explanation:

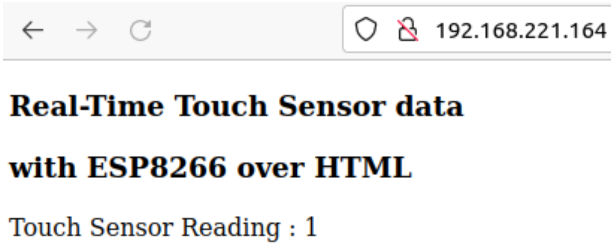
- Include ESP8266WiFi library, provide credentials for WiFi and define touch sensor pin.
- Connect with WiFi, start the server and print IP address on serial monitor to connect.
- Check for client availability, define HTML page, read the data from touch sensor pin with **digitalRead(pin)** function and auto-refresh the page to update the status of data.

**Output:** Below we can see after being connected with WiFi ESP8266 started its server and gave us an IP to access to the web page.

```
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Server started
IP address:     192.168.221.164
```

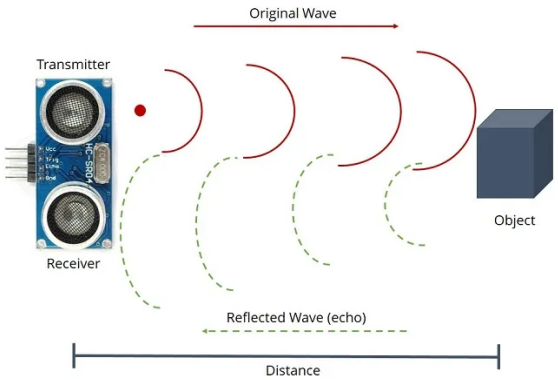
**HTML Web page:** We can see in the following figures that HTML page is created and giving us the real-time touch sensor data.



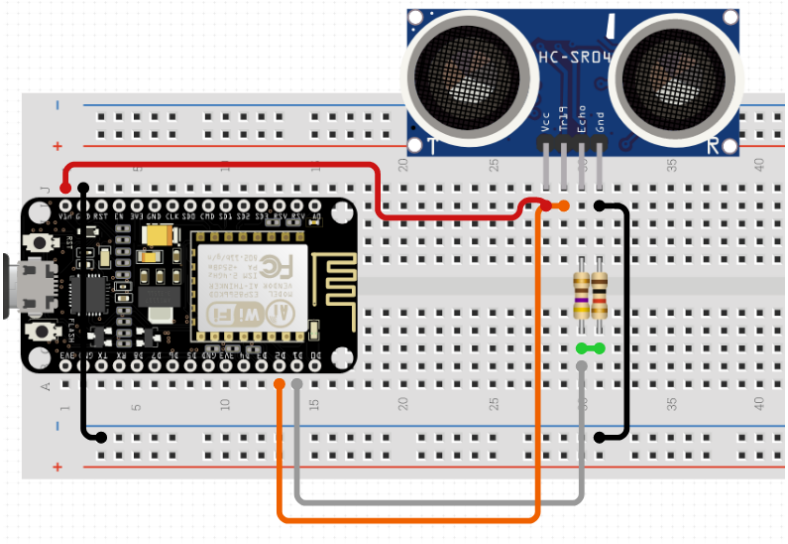

**Interfacing Ultrasonic Sensor:** Ultrasonic sensor or distance sensor is used in many applications of robotics including obstacle avoidance, object localization (complex), obstacle following and so on. HC-SR04 is one of the cheapest yet accurate enough type of distance sensor used in mostly hobbyist projects as shown in below figure (Majju PK). If we want to do more complex tasks such as easy object localization and autonomous navigation by slamming the map, we will be needing a Li-DAR which can these two talks very easily with good enough accuracy.


**Working of HC-SR04:** This sensor sends ultrasonic sound waves once triggered and waves get back if they hit an object as illustrated in below figure (Seeed Studio). Depending on the time in which it came back and the speed of the sound we can calculate the distance. We will be taking the half duration because full duration will give us double distance of actual distance of going and coming back.



**Connecting with ESP8266:** Look at the below circuit for connectivity and ignore the resistors and their green wire connection as we will not use them. Follow below steps and figure for connectivity from ESP8266 to HC-SR04.

- VU to VCC.
- GND to GND.
- D1 (GPIO5) to Echo.
- D2 (GPIO4) to Trig.




## Code:

```
// Define pin connections for the ultrasonic sensor
#include<ESP8266WiFi.h>
#define TRIGGER_PIN 4
#define ECHO_PIN 5
// Define variables for ultrasonic sensor
long duration;
int distance;
String ssid = "CRAIB-LAB";
String pass = "********";
WiFiServer server(80); // Initilize Server

void setup() {
Serial.begin(9600);
// Set pin modes for ultrasonic sensor
pinMode(TRIGGER_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
WiFi.begin(ssid,pass); // Connect with WiFi
while (WiFi.status()!=WL_CONNECTED){
delay(1000);
Serial.println("Connecting to WiFi");
}
server.begin();// Start Server
Serial.println("Server Started");
Serial.println("IP Address : ");
```

```
Serial.print(WiFi.localIP()) ;// Print IP on serial monitor
}
void loop() {
// Trigger the ultrasonic sensor to send a signal
digitalWrite(TRIGGER_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIGGER_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER_PIN, LOW);
// Measure the duration of the signal
duration = pulseIn(ECHO_PIN, HIGH);
// Calculate the distance using the duration
distance = duration * 0.034 / 2;
delay(500);
WiFiClient client = server.available();
client.println("<html>");
client.println("<h3> Real-Time Ultrasonic senosr </h3>");
client.println("<h3> data with ESP8266 over HTML </h3>");
client.println("Ultransonic Sensor Reading : ");
client.print(distance);
client.print(" cm");
client.println("<script> function autoRefresh() {window.location = window.location.href; } setInterval('autoRefresh()',
1500); </script>");
client.println("</html>");
}
```

**Code Explanation:**

- Include ESP8266WiFi library, provide credentials for WiFi and define echo and trigger pins.
- Connect with WiFi, start the server and print IP address on serial monitor to connect.
- Trigger ultrasonic sensor to throw sound waves for 10 microseconds.
- Calculate the duration at echo pin when waves are back after hitting the object with **pulsein(ECHO_PIN,HIGH)** function.
- Calculate the distance and divide it by 2 and put a delay of half second.
- Check for client availability, define HTML page, send the data to HTML page and auto-refresh the page to update the status of data.

**Output:** Below you can see the snippets of HTML web page showing read-time distance coming from HC-SR04 sensor.