TTP223 Touch Sensor Module

A capacitive touch sensor used for detecting finger touches as digital signals.

Pinout of TTP223 Module

Pin	Description
VCC	Power Supply (2V–5.5V, works well with 3.3V or 5V)
GND	Ground
OUT	Digital Output (connect to ESP32 digital input pin, e.g., GPIO 4)

Working Principle

- The TTP223 uses capacitive sensing to detect human touch.
- When you **touch the sensor pad**, the output pin **goes HIGH** (3.3V/5V).
- When there is **no touch**, the output pin **remains LOW** (0V).
- Internally, it works by detecting a change in capacitance caused by a finger approaching the pad.

How It Works:

- No touch → **OUT = LOW**
- Touch detected → **OUT = HIGH**
- Output is binary (digital), ideal for switch-type controls.
- Can replace traditional mechanical buttons.

Points to Remember:

- Use digitalRead() on ESP32 to read HIGH/LOW state.
- Suitable for 3.3V or 5V logic (ESP32 safe).
- Can be extended with a conductive foil/pad to increase touch area.

Example Use Cases:

- Touch-based **LED switches**
- Hidden **power buttons** for appliances or gadgets
- Capacitive **door locks** or keypads
- **Touch-to-wake** screens or displays

Code:

```
#define Touch_SW 4
#define LED 2
void setup() {
Serial.begin(115200);
pinMode(Touch_SW, INPUT);
pinMode(LED, OUTPUT);
}
void loop() {
int Touch = digitalRead(Touch_SW);
if(Touch == HIGH){
  digitalWrite(LED, HIGH);
  Serial.println("Touched");
} else{
  digitalWrite(LED, LOW);
  Serial.println("Untouched");
}
delay(500);
}
```

DHT11 Sensor Module (Digital Temperature & Humidity Sensor)

A sensor that provides calibrated digital output for **temperature** and **humidity** via a single-wire interface.

Pinout of DHT11 Module

Pin	Description
VCC	Power Supply (3.3V–5.5V)
GND	Ground
DATA	Digital Data Output (connect to any ESP32 digital GPIO, e.g., GPIO 23)

Working Principle:

- Inside the sensor:
  - **Humidity** is sensed using a resistive-type humidity sensing element.
  - **Temperature** is measured using an NTC thermistor.
- Data is converted to digital using an internal ADC and sent to the microcontroller.

Points to Remember:

- Use a **DHT library** (e.g., DHT.h) with DHT dht(GPIO_PIN, DHT11);
- Call dht.begin(); in setup() and use dht.readTemperature() / dht.readHumidity() in loop().
- Only read data every **1–2 seconds** (sampling rate limit).
- DHT11 is less accurate and slower than DHT22 but suitable for basic applications.

Typical Accuracy:

Parameter	Range	Accuracy
Temperature	0°C to 50°C	±2°C
Humidity	20%–90% RH	±5% RH

Example Use Cases:

- **Weather stations** for indoor/outdoor monitoring
- **Home automation** (AC/fan control based on temp/humidity)
- **Greenhouse monitoring**
- **IoT dashboards** (send data to cloud/Firebase)
- **Smart appliances** (trigger fan/buzzer when temp exceeds limit)

Code:

```
#include<DHT.h>
#define DHT_Pin 4
#define DHT_Type DHT11
DHT dht(DHT_Pin, DHT_Type);
void setup() {
  Serial.begin(115200);
  dht.begin();
}
void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  if(isnan(humidity) || isnan(temperature))
{
    Serial.print("DHT11 Failed to read value, check connections!");
  }
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println("%");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println("C");
  delay(2000);
}
```