DHT11 HUMIDITY SENSOR WITH NODEMCU AND NOTIFICATION SERVER (IFTTT)

Description:

This is a simple practical explaining how to receive Notification when temperatures or Humidity goes High or Low remotely. This practical consists of a DHT11 sensor which senses Humidity and Temperature attached to NodeMCU which sends the sensed Temperature and Humidity values and triggers notification to the user via Email or a simple Message using IFTTT Notification Service.

The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a
capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital
signal on the data pin (no analog input pins needed). It's fairly simple to use, but requires careful timing to grab data.

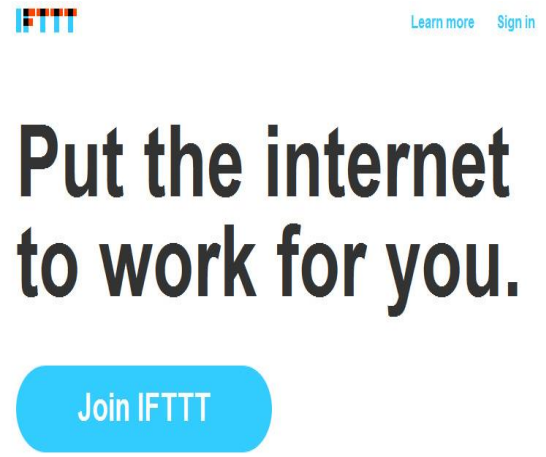
IFTTT is a Notification server that will enable you to connect 2 channels so that, when something happens with one service, a trigger goes off and an action takes place automatically on the other.

A "Channel" is nothing more than a fancy word for a service. The brilliant thing about IFTTT is that its variety of channels allows it to offer something to everybody. As of this writing, there are nearly 70 different channels that you can use for creating recipes and the list is always increasing. Some of the most famous and used channels include: Facebook, Twitter, Tumblr, Instagram, Youtube, SoundCloud, Dropbox, Evernote, Pocket & many, many more!

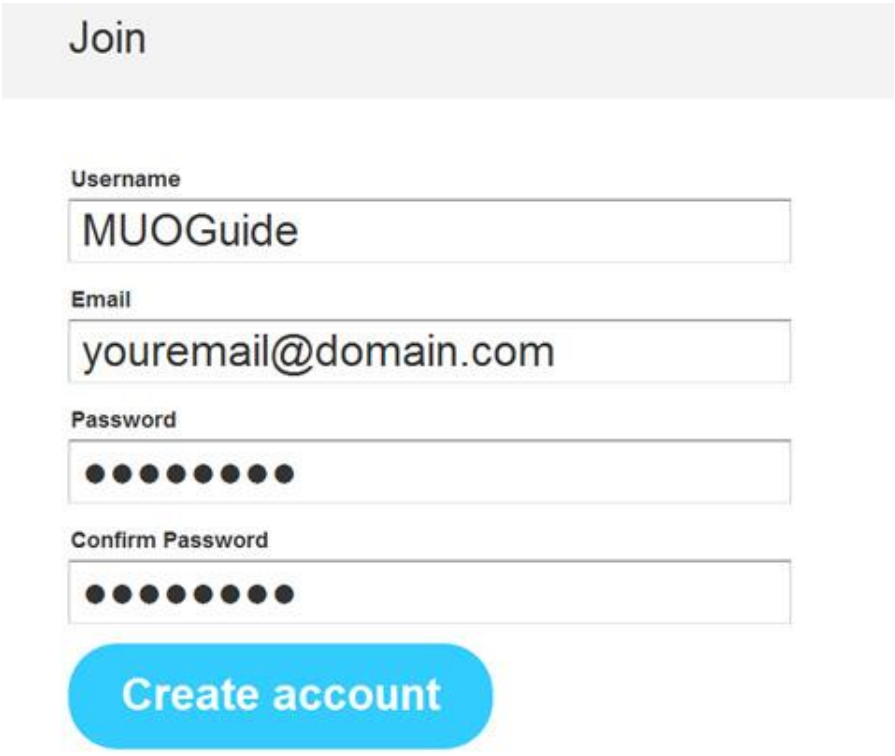
Recipes are what make IFTTT worth your time. Basically they are the combination of 2 channels, using a "Trigger" and an "Action". When something happens on one channel, it triggers an action.

How Do I Make a Recipe?

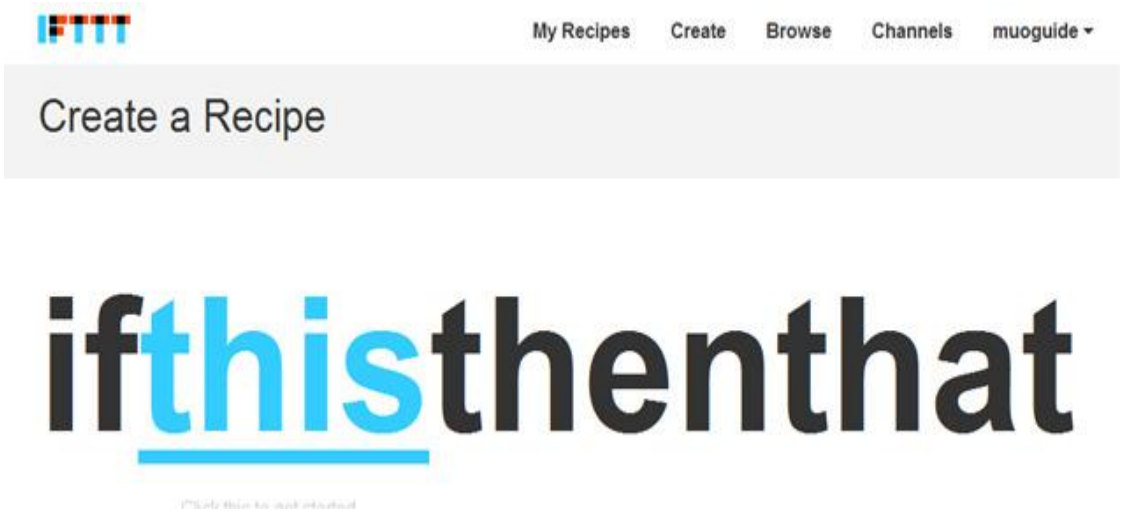
Glad you asked! Making recipes is a snap with IFTTT. First things first, you need to create an account with IFTTT. Head over to IFTTT and you will be greeted with the following screen:



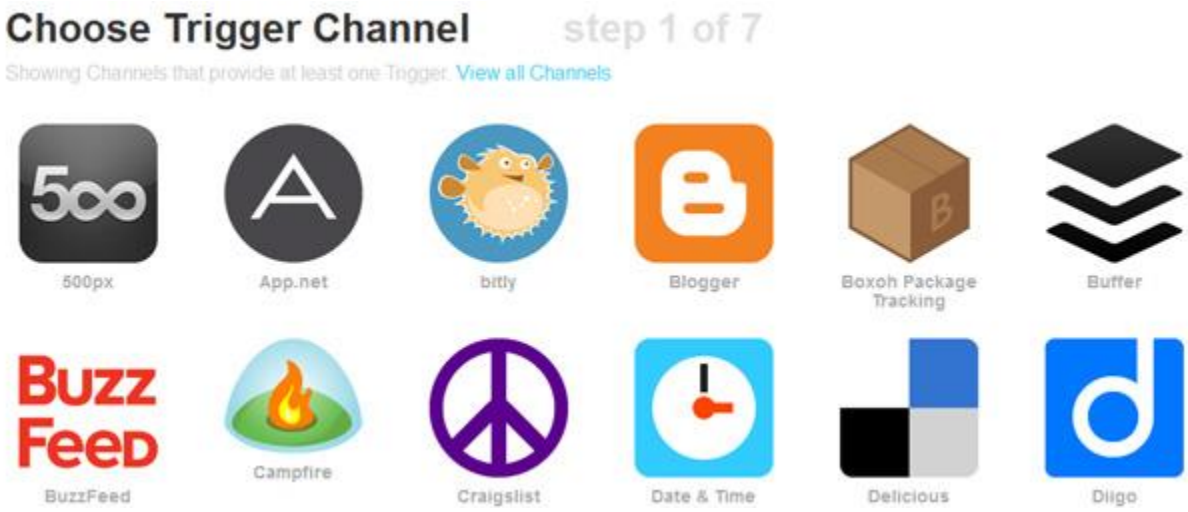
Then, when you click on Join IFTTT, you'll need to fill a very short form:



The first step is to click on Create,

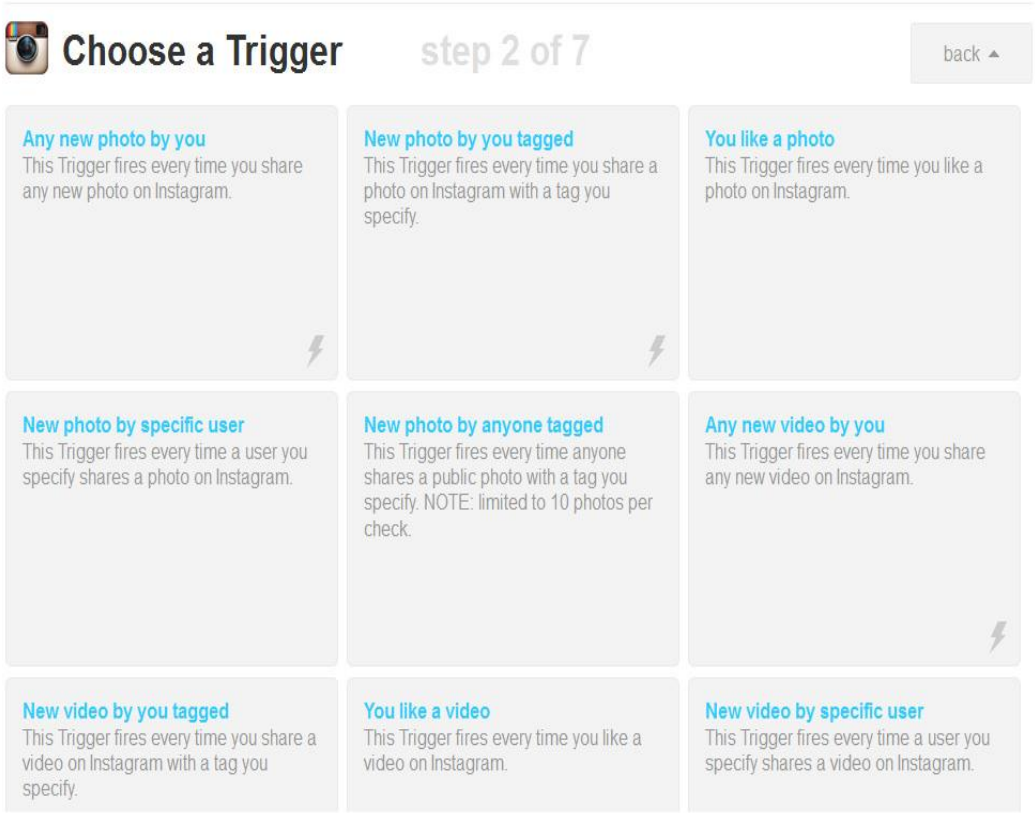


click on THIS. After doing so, we will be taken to the next step:

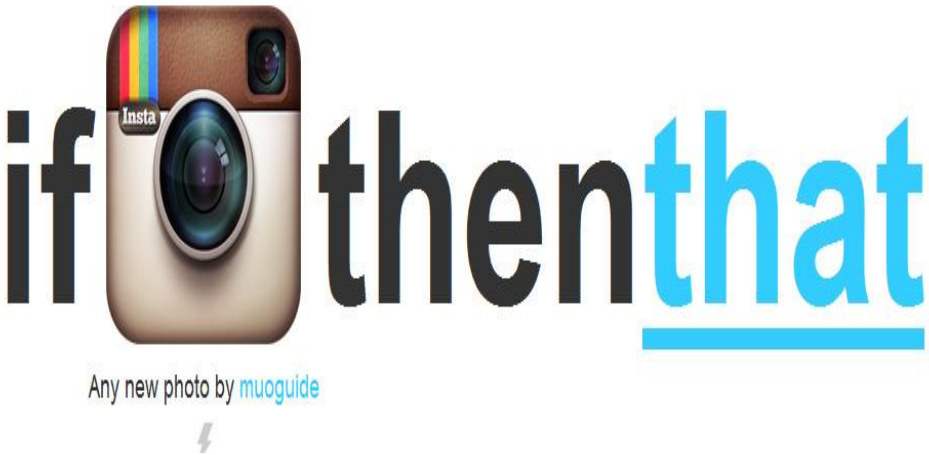


select the Maker trigger, which will then ask us to connect Maker just this once.

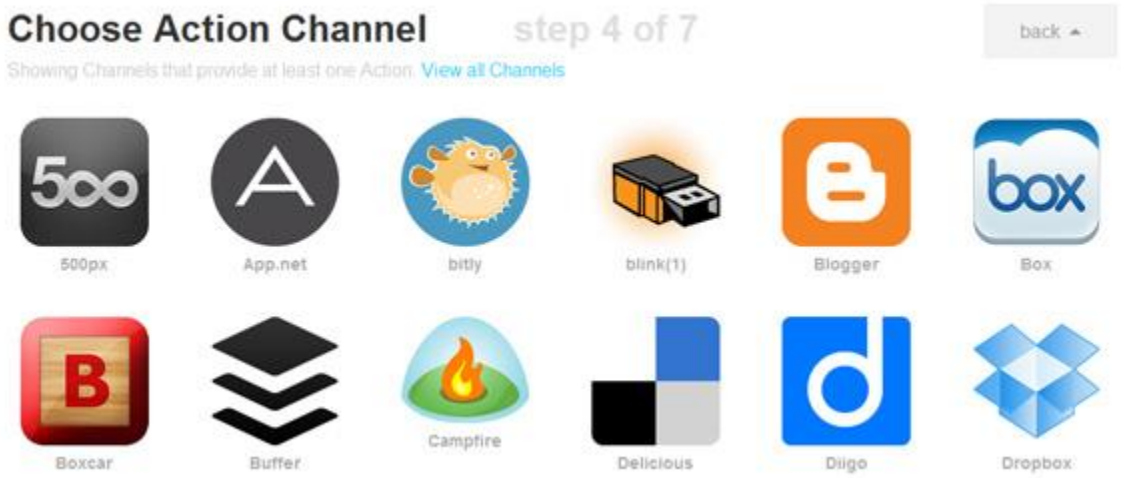
Having done that, we'll choose a trigger action:



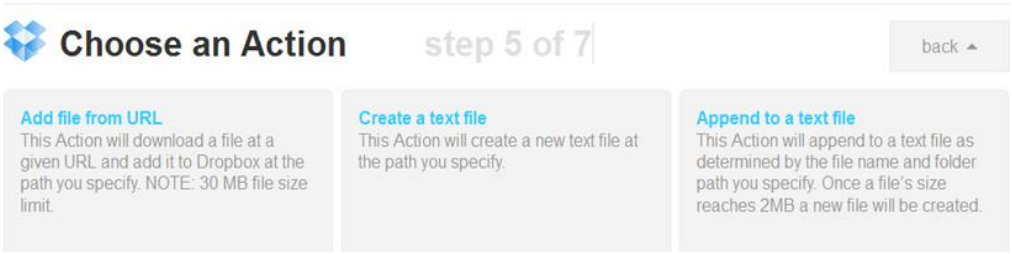
I'm going to select the first one and move on to the next step. We're going to click on THAT on the following screen:



Naturally, we're going to select Dropbox and activate it just the one time that it's necessary. After doing this, we will be greeted by the second batch of trigger actions:

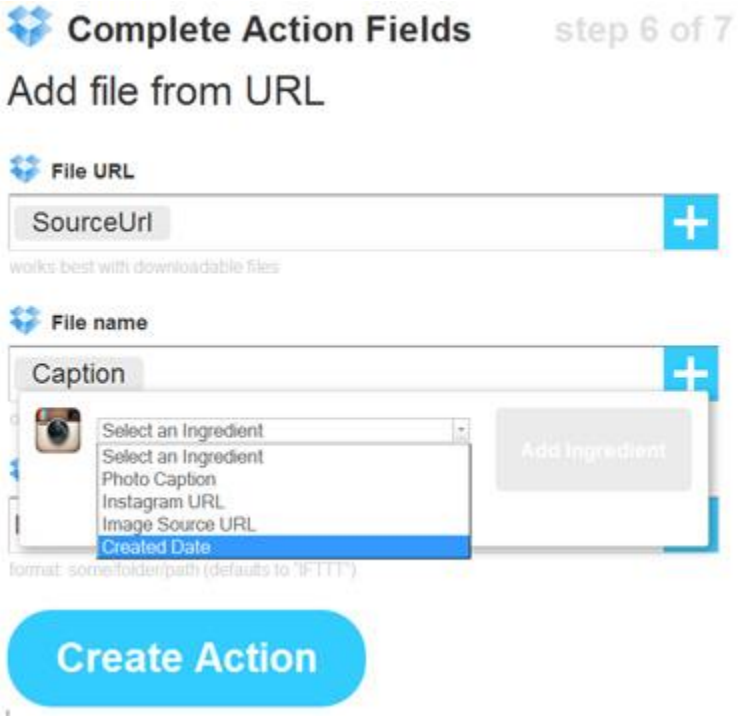


We'll select the first trigger and be taken to this:



In this case, it's asking us where to grab the photos, how to name them and where it should put them. I decided to change the format of the name so that they're named according to the date in which I post them. All you have to do is click on the "ingredient" and you'll get a dropdown as shown above. We take care

of that and we get this:



Create and activate

step 7 of 7

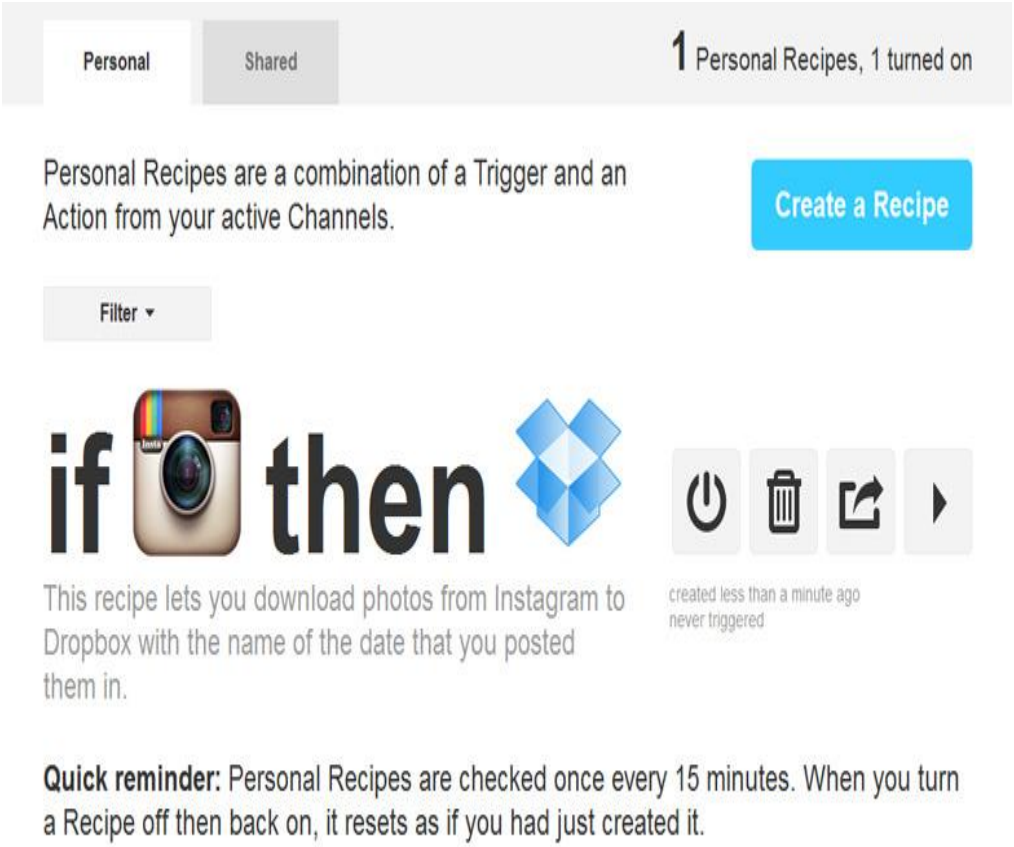
Any new photo by muoguide

Add file from URL to Al M's Dropbox

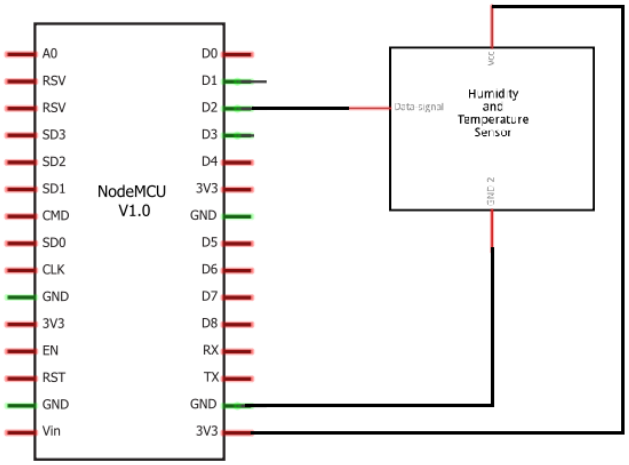
Description

use '#' to add tags

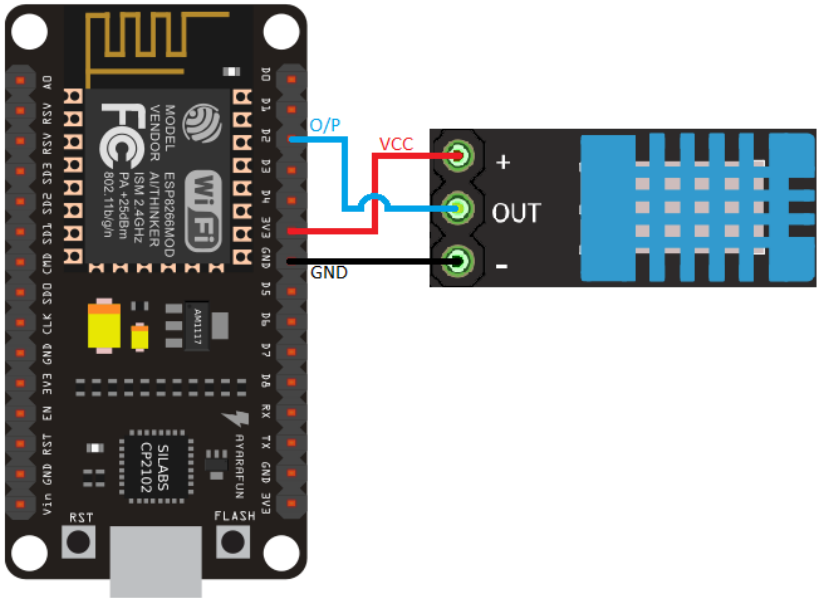
Create Recipe



Circuit Diagram:



Connection Diagram:



Components Required:

- ✓ NodeMCU
- ✓ DHT11

Library Required:

- ✓ ESP8266WiFi Library
- ✓ DHT Library

(for more details refer: https://github.com/adafruit/DHT-sensor-library)

Specifications(Nodemcu):

- ✓ Memory - 20kB
- ✓ Storage - 4MB
- ✓ Operating Voltage - 3v

Specifications(Dht11):

- ✓ Low cost
- ✓ 3 to 5V power and I/O
- ✓ 2.5mA max current use during conversion (while requesting data)

- ✓ Good for 20-80% humidity readings with 5% accuracy
- ✓ Good for 0-50°C temperature readings ±2°C accuracy
- ✓ No more than 1 Hz sampling rate (once every second)
- ✓ Body size 15.5mm x 12mm x 5.5mm

CODE:

```
#include <ESP8266WiFi.h>
#include "DHT.h"
#define DHTPIN D2     // what pin we're connected to
#define DHTTYPE DHT11   // define type of sensor DHT 11
DHT dht (DHTPIN, DHTTYPE);

const char* ssid     = "____";
const char* password = "____";
const char* host = "maker.ifttt.com";//dont change
const String IFTTT_Event = "192.168.0.1"; //Write your ifttt event name
here
const int puertoHost = 80;
const String Maker_Key = "ccgrLuCecu6kfkUpS7ncvp";

String conexionIF = "POST
/trigger/"+IFTTT_Event+"/with/key/"+Maker_Key +" HTTP/1.1\r\n" +
                "Host: " + host + "\r\n" +
                "Content-Type: application/x-www-form-urlencoded\r\n\r\n";

 void setup() {
 Serial.begin(115200);
 delay(10);
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);
 dht.begin();
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());

 while (WiFi.status() != WL_CONNECTED) {
```

```
  delay(500);
  Serial.print(".");
  }
}

void loop() {

    float h = dht.readHumidity();
  float t = dht.readTemperature();
  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
 if (t > 30){
  // This will send the request to the server
  Serial.println("E-mail sending");
  client.print(conexionIF);
  delay(10);
 }
 else {
  // This will send the request to the server
  Serial.println("ifttt disconnected");
  delay(10);
 }
  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  Serial.println();
 Serial.println("closing connection");
 delay(10000);
}
```