# **SERVERLESS IOT PROCESSING**

PHASE 3 : **DEVELOPMENT PART-1**

**INTRODUCTION:**

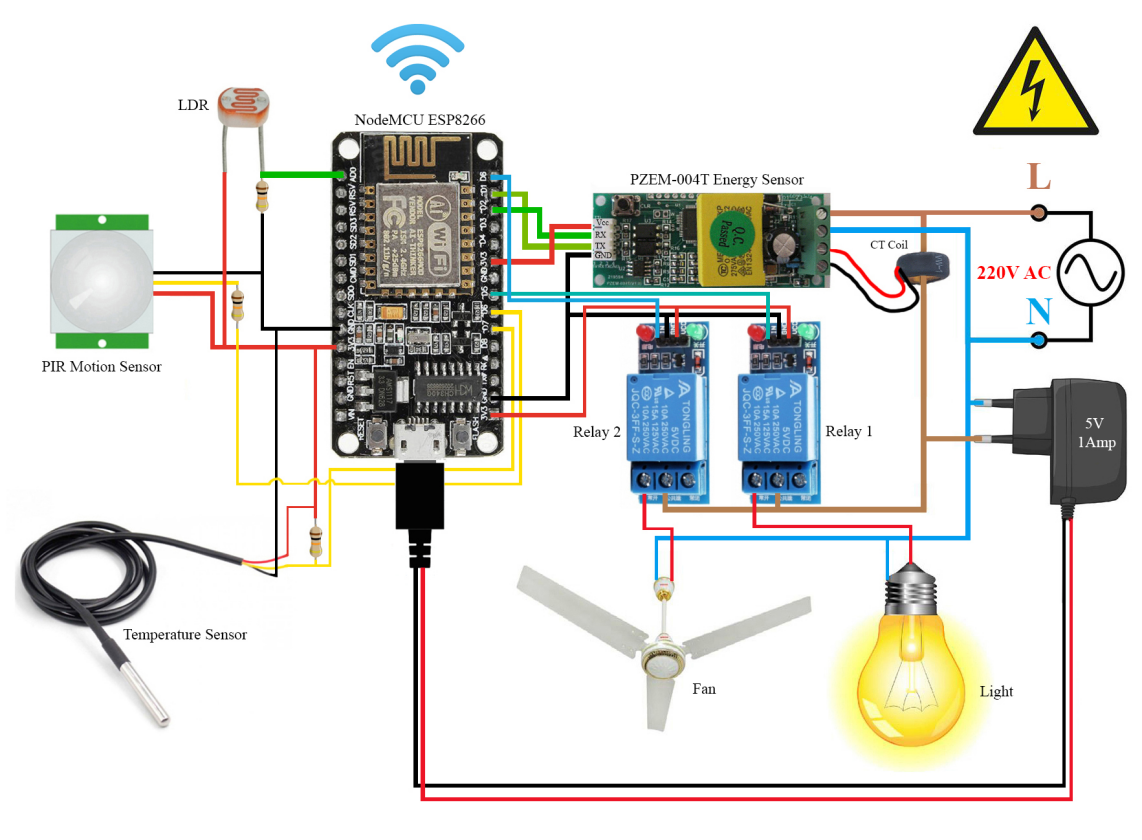
The Internet of Things (IoT) has emerged as a transformative technology, connecting everyday objects and devices to the internet, enabling them to collect and exchange data. This interconnected network of devices has found applications in various industries, from smart homes and agriculture to healthcare and industrial automation. In this document, we will explore the concept of creating a Serverless IoT Solution using NodeMCU, addressing the need for innovative problem-solving in this domain. The rapid proliferation of IoT devices presents both opportunities and challenges, with traditional IoT architectures often relying on centralised servers or gateways that can become bottlenecks as the number of devices grows. To tackle these challenges and improve scalability and cost-efficiency, serverless IoT solutions leveraging NodeMCU and cloud computing resources have gained prominence, offering a lightweight and cost-effective alternative. This document aims to outline the design and implementation of such a solution, demonstrating how NodeMCU can seamlessly integrate with serverless cloud platforms to create innovative and scalable IoT devices, contributing to the ongoing evolution of IoT technology.



**PROBLEM DEFINITION:**

The project aims to transform a home into a smart living space using IBM Cloud Functions for IoT data processing. The goal is to collect data from various smart devices, process it in real-time, and automate routines for energy efficiency and home security. This involves designing the smart home setup, implementing data collection and processing, and leveraging IBM Cloud for storage and analysis.

**CIRCUIT DIAGRAM AND COMPONENTS:**



**STEPS TO INTEGRATE CLOUD AND VARIOUS SENSORS:**

1. **HARDWARE SETUP:**

- You'll have a NodeMCU (ESP8266) as your microcontroller board.

- Various sensors like a DHT22 (Temperature and Humidity Sensor), PIR Motion Sensor, and a Light Sensor can be used.

- Connect these sensors to the NodeMCU using appropriate wiring and resistors if needed.

1. **CODING FOR SENSOR AND DATA READING:**

- Write a program in Arduino IDE for NodeMCU to read data from the connected sensors.

- Use libraries like Adafruit DHT for the DHT22 sensor, and appropriate libraries for other sensors.

- Collect data from these sensors, and package it into JSON format.

1. **CONNECTING TO IBM CLOUD:**

- Sign in to your IBM Cloud account and create an instance of the Watson IoT Platform service.

1. **DEVICE CONFIGURATION:**

- In the Watson IoT Platform, create a device type for your NodeMCU and add a new device under that type.

- Generate API keys and authentication tokens for this device.

1. **NODE MCU WITH CLOUD INTEGRATION:**

- Use the IBM Watson IoT library for Arduino to enable your NodeMCU to connect to the Watson IoT Platform.

- Configure the library with the device type, device ID, API keys, and authentication tokens.

1. **DATA TRANSMISSION TO CLOUD:**

- Modify your NodeMCU code to send sensor data to the Watson IoT Platform using MQTT or HTTP.

- Send the data in JSON format to the cloud at regular intervals (e.g., every few seconds).

1. **CLOUD PROCESSING:**

- In the Watson IoT Platform, set up event handling and data processing rules to analyse incoming sensor data.

- You can create rules that trigger actions based on the data received, like turning on or off other components.

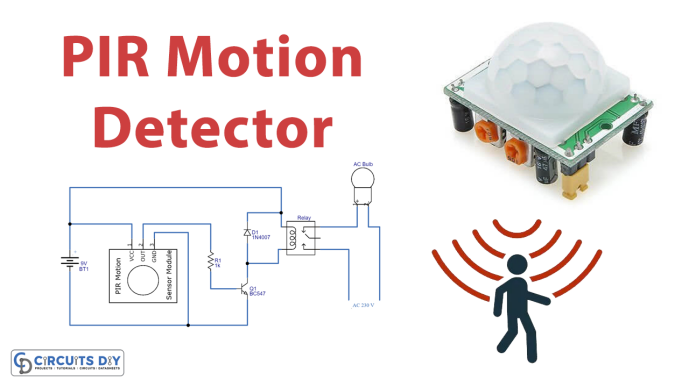
1. **CONTROLLING OTHER COMPONENTS:**

- Use cloud-based triggers to control other devices or sensors. For instance, if the motion sensor detects movement, you can create a rule to turn on lights or an alarm.

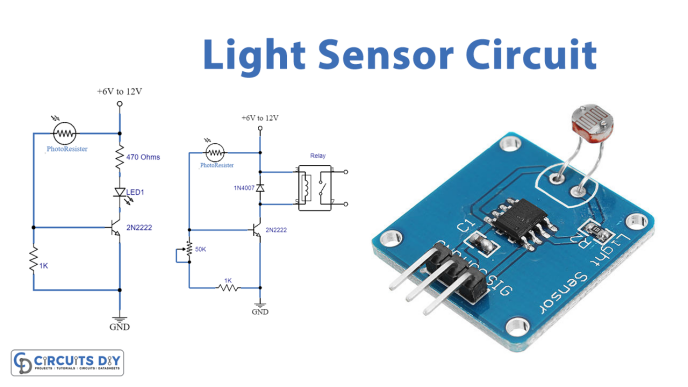
1. **SENSOR DESCRIPTION:**

- *DHT22 (Temperature and Humidity Sensor)*: This sensor provides temperature and humidity data. It has a digital output and is commonly used for climate monitoring in IoT projects.

- *PIR Motion Sensor:* Passive Infrared (PIR) sensors detect motion by measuring changes in infrared radiation. They are commonly used for occupancy detection in smart home applications.



*- Light Sensor:* Light sensors, like LDRs (Light Dependent Resistors) or photodiodes, measure ambient light levels. They are used to detect the amount of light in the environment, allowing you to automate lighting systems, for example.



By connecting these sensors to your NodeMCU and integrating with the IBM Cloud, you can create a smart home system that collects data from the sensors, processes it in the cloud, and triggers actions based on the data received. This allows you to control other components or sensors in your smart home based on real-time sensor readings.

**PROCESS OF IMPLEMENTING: (cpp)**

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include <DHT\_U.h>

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#define DHTPIN 2

#define DHTTYPE DHT22

const int motionSensorPin = 5;

const int lightSensorPin = A0;

const char\* ssid = "your\_SSID";

const char\* password = "your\_PASSWORD";

const char\* mqtt\_server = "your\_MQTT\_SERVER";

const char\* mqtt\_user = "your\_MQTT\_USERNAME";

const char\* mqtt\_password = "your\_MQTT\_PASSWORD";

DHT\_Unified dht(DHTPIN, DHTTYPE);

WiFiClient espClient;

PubSubClient client(espClient);

void setup() {

Serial.begin(115200);

delay(10);

dht.begin();

pinMode(motionSensorPin, INPUT);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

// Connect to MQTT Broker

client.setServer(mqtt\_server, 1883);

client.setCallback(callback);

}

void loop() {

// Read temperature and humidity

sensors\_event\_t event;

dht.temperature().getEvent(&event);

float temperature = event.temperature;

dht.humidity().getEvent(&event);

float humidity = event.relative\_humidity;

// Read motion sensor

int motionState = digitalRead(motionSensorPin);

// Read light sensor

int lightValue = analogRead(lightSensorPin);

int lightState = (lightValue > 500) ? HIGH : LOW; // Adjust the threshold as needed

String data = "{\"temperature\":" + String(temperature) +

",\"humidity\":" + String(humidity) +

",\"motion\":\"" + String((motionState == HIGH ? "detected" : "none")) + "\"" +

",\"light\":\"" + String((lightState == HIGH ? "on" : "off")) + "\"}";

if (client.connect("NodeMCU", mqtt\_user, mqtt\_password)) {

client.publish("sensor/data", data.c\_str());

client.loop();

delay(5000); // Send data every 5 seconds

}

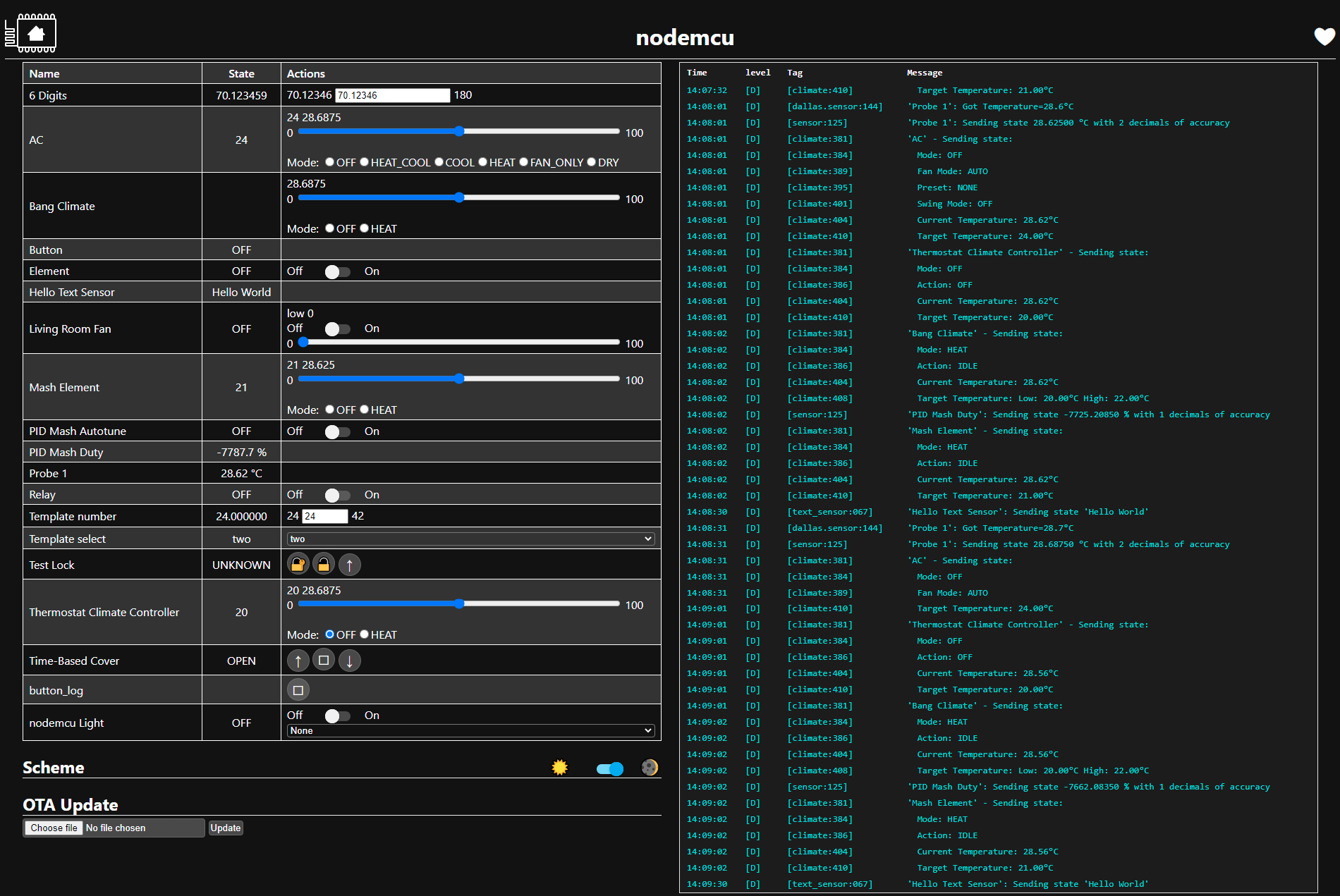
}

void callback(char\* topic, byte\* payload, unsigned int length) {

// Handle MQTT messages if needed

}

**SAMPLE OUTPUT:**



**DATA TRANSFER BETWEEN CONTROLLER AND IBM:**

{

"temperature": 23.5,

"humidity": 50.2,

"motion": "none",

"light": "on"

}

* "temperature" and "humidity" represent the readings from the DHT22 sensor.
* "motion" indicates whether the motion sensor has detected motion ("detected") or not ("none").
* "light" represents the state of the light sensor, indicating whether it's bright enough to be considered "on" or not ("off").

This data format is just an example. You can customise the JSON structure to match your requirements. Once the data is sent to the IBM Cloud, you can process it based on your rules and requirements within the IBM Cloud platform.