Electronics Engineering Lab

Project

<u>Home Automation System</u>

4th SEMESTER

Submitted to: Engr. Ali Hassan

Session: 2021 **Section: C** **Group: 7**

SUBMITTED BY

Name	CMS ID
Muhammad Shaheer Ul Haq	373516
Muizuddin Muhammad Yusuf	376432
Muhammad Talha Irfan	368635
Muhammad Abdullah Malik	373953

School of Mechanical and Manufacturing Engineering

Introduction

Home Automation System are an important constituent of the Internet of Things ("IoT"). A home automation system monitors and/or control home attributes such as lighting, climate, and appliances. It may also include home security such as access control and alarm systems.

A home automation system typically connects controlled devices to a central smart home hub (sometimes called a "gateway"). The user interface for control of the system uses either wall-mounted terminals, tablet or desktop computers, a mobile phone application, or a Web interface that may also be accessible off-site through the Internet. [1]

The purpose of this project is to design and implement a Home Automation System using a microcontroller and a sensor. The system should be able to control different home appliances. The main aim is to create a cost-effective and efficient home automation system that can be controlled through a smartphone application.

The Home Automation System in this project will involve Temperature and Humidity sensing to switch on or off, with automation, a DC fan and an external circuit using a relay.

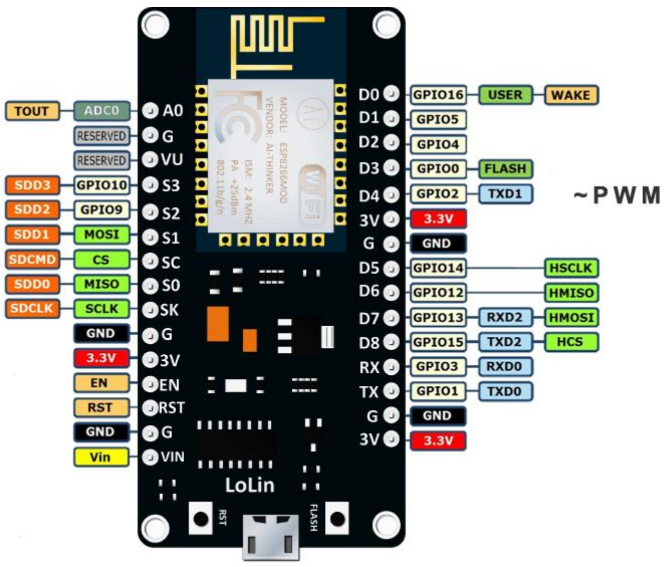
Hardware

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which are based on the ESP-12 and ESP-32 modules.

A "core" is the collection of software components required by the Board Manager and the Arduino IDE. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE", to compile an Arduino C/C++ source file for the target MCU's machine language. This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs to be widely used in IoT applications. [2]

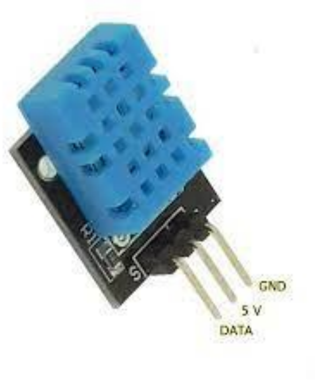
NodeMCU is powered by 5V by either Vin pin, or by micro-USB port. It only has 3.3V output signals instead of max 5V on Arduino.

The diagram and the pinouts of the NodeMCU are shown:



DHT11 Temperature and Humidity Sensor is made of two parts, a capacitive humidity sensor and a thermistor. There is also a very basic chip inside that does some analog to digital conversion and spits out a digital signal with the temperature and humidity. The digital signal is easy to read using any microcontroller. [3]
The characteristics and pinout of the DHT11 sensor are shown:

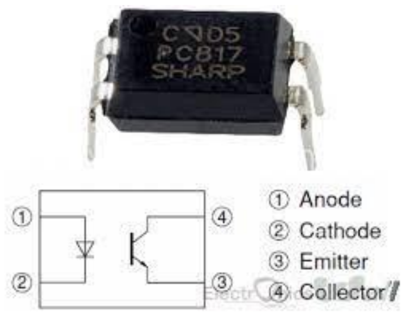
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- 3 pins with 0.1" spacing



A **2n222 NPN transistor** is used for switching on the 5V DC Fan.

A **1n4007 Silicon diode** is used for protection of DC device.

NodeMCU is only capable of 3.3V output voltages, which are not enough to switch on the transistor or relay directly. Thus, a **PC817 optocoupler** [4] consisting of an LED and a phototransistor is used, as it requires very low input voltage and can be used to switch the DC circuit by the NodeMCU data pins:



A **5V DC fan** is controlled by the home automation system, for demonstration purposes:

A **5V 1-channel relay** switches on an external mains-powered circuit, according to signals from the NodeMCU.

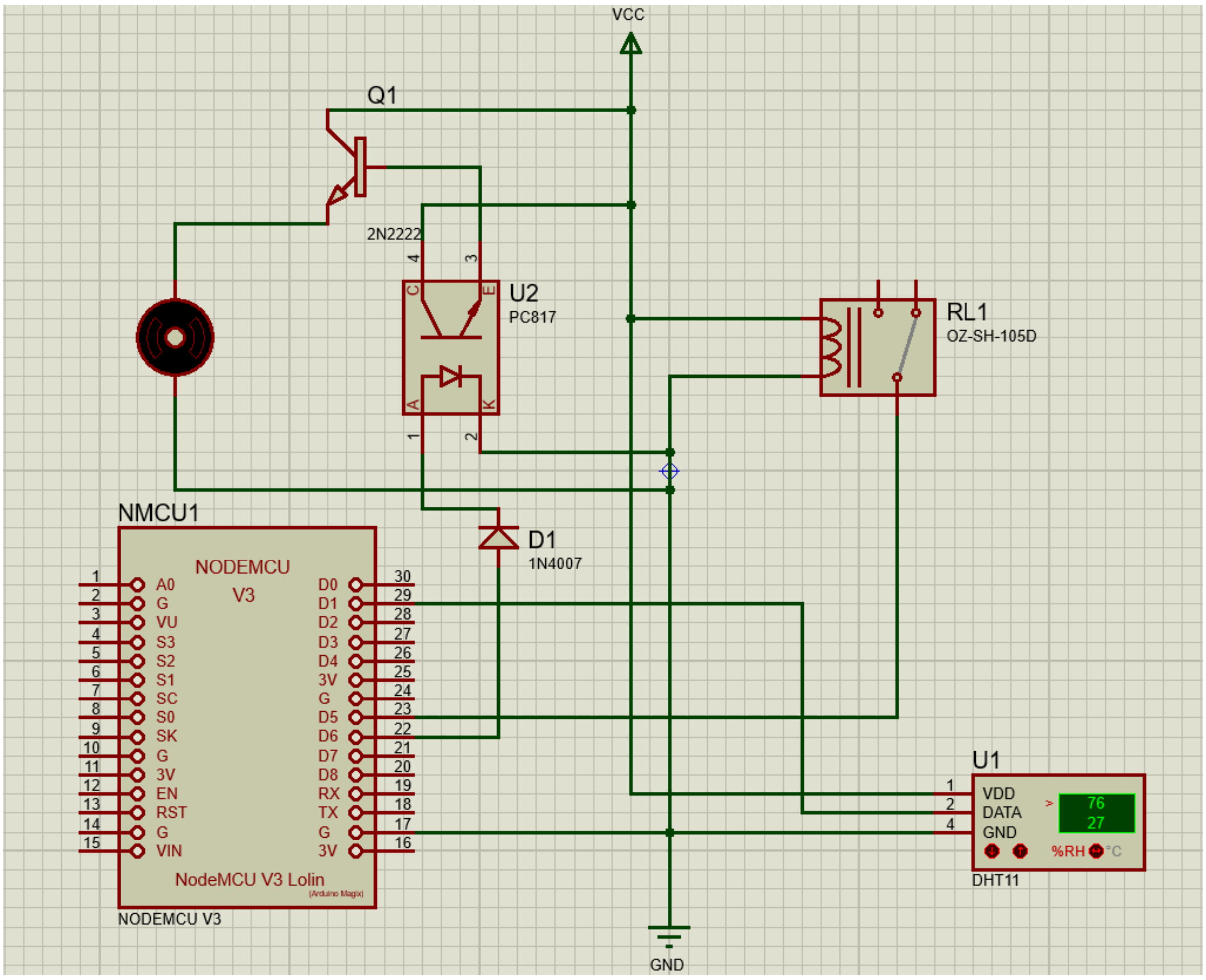
MB102 Power Supply Module takes in a 7 to 12V DC input (e.g. 9V battery) and outputs 5V or 3.3V as required. The breadboard power rails are powered and grounded by it for all essential components (transistor, relay, DHT11, DC fan etc) to provide a voltage supply to.

The list of major hardware components is as follows:

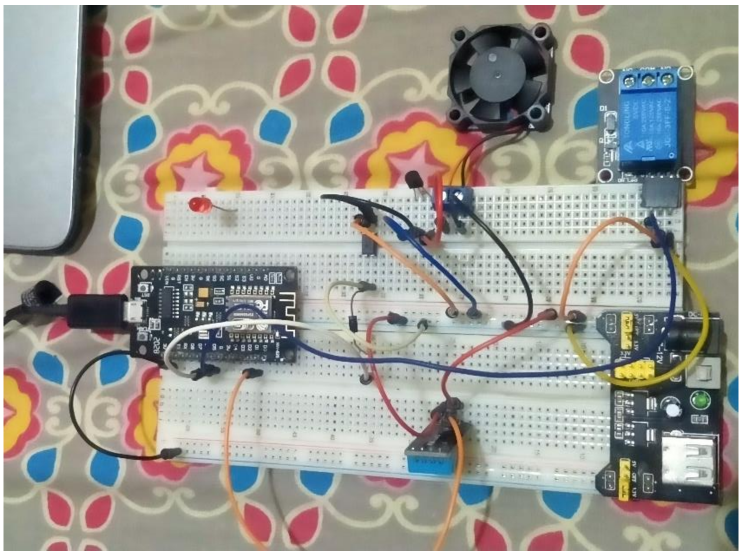
- NodeMCU ESP8266 microcontroller
- DHT11 Temperature Humidity Sensor
- 2n222 NPN transistor
- 5V 1-channel relay
- MB102 Power Supply Module
- 9V battery
- PC817 optocoupler
- 5V DC fan
- 1n4007 Silicon diode
- Breadboard(s)
- Jumper Wires
- Extension Wires for external relay circuit

Circuit Diagram

A Schematic Diagram of the whole system, in Proteus, is as follows:



The schematic is used as a guide to set up the wiring of the hardware. The complete assembled hardware is pictured:



DHT11 is connected to GPIO pin 5 of the physical NodeMCU, DC fan with pin 12 and Relay with pin 14.

Code

The code is written in C++ and uses the **Arduino.h** library (and **Hash.h**) for microcontroller operations (setting pins, switching pin states to HIGH or LOW and more). The DHT11 sensor uses the **DHT.h** library [5] and its dependency **Adafruit_Sensor.h**. ESP8266 WiFi module starts a local wireless network by **ESP8266WiFi.h** [6] (part of **Arduino.h)** to host the web app on. The web app is written within the sketch itself, using the **ESPAsyncWebServer** [7] and its dependency library **ESPAsyncTCP.h**, and programmed in HTML, CSS and Javascript languages (a resource to learn web development is [8]).

main.cpp

```
// Import required libraries
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_Sensor.h>
#include <Hash.h>
#include <DHT.h>

const char* ssid     = "Home-Automation-AP";
const char* password = "12345678";

const char* thresholdInput = "threshold";
```

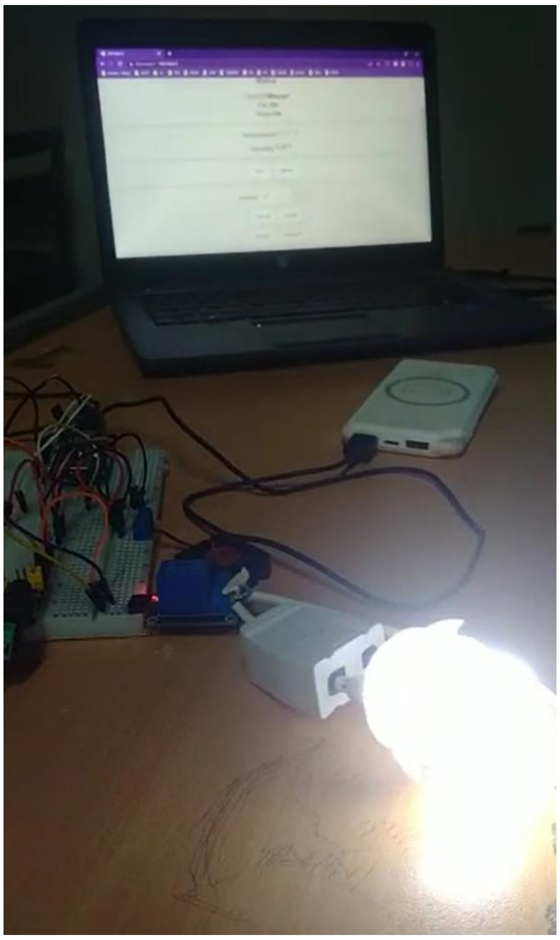
```
#define DHTPIN 5     // Digital pin connected to the DHT sensor
#define FANPIN 12
#define RELAYPIN 14

#define DHTTYPE    DHT11     // DHT 11

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 0.0;
bool autostate = 0;
float threshold = 30.0;
bool f = 0;
bool r = 0;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;    // will store last time DHT was updated

// Updates DHT readings every second
const long interval = 1000;

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    * {
    margin-left: 0;
    margin-right: 0;
    padding-left: 0;
    padding-right: 0; }
    html {
     font-family: Arial;
     display: inline-block;
     margin: 0px auto;
     text-align: center;
     background-color: #e3e3e3;
    }
    h2 { font-size: 1.8rem; color: black; background-color: #909090;}
    h3 { font-size: 1.6rem; }
    ul { list-style: none;}
    ul li { padding: 0.2rem; padding-top: 0rem;}
    span { font-size: 1.5rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      vertical-align:middle;
      padding-bottom: 10px;
    }
    .status-labels { text-align: center; }
    .toggle-buttons { font-size: 1rem; padding: 1rem 1.5rem; }
    .auto-buttons { font-size: 1rem; padding: 1rem 1rem; }
    .manual-buttons { font-size: 1rem; padding: 1rem 1.5rem; }
    div { border: 1px solid #303030; border-radius: 10px; margin: 5px ;}
  </style>
</head>
<body>
  <h2>NodeMCU DHT11 <d style="color:rgba(234, 18, 18, 0.957);">Temperature</d>
and <d style="color: blue;">Humidity</d> Sensor</h2>
  <div>
    <h3>Status</h3>
  <ul>
    <li><span class="status-labels">Control</span>
    <b><span id="s">%STATUS%</span></b></li>

    <li> <span class="status-labels">Fan</span>
    <b><span id="f">%FANSTATUS%</span></b></li>

    <li> <span class="status-labels">Relay</span>
    <b><span id="r">%RELAYSTATUS%</span></b></li>
  </div>
  <div>
  <p>
    <span class="dht-labels">Temperature</span>
    <b><span style="color: #f39c12;" id="temperature">%TEMPERATURE%</span></b>
    <sup class="units">&deg;C</sup>
  </p>
  <p>
    <span class="dht-labels">Humidity</span>
    <b><span style="color: #3498db;" id="humidity">%HUMIDITY%</span></b>
    <sup class="units">%</sup>
  </p>
  </div>
  <div>
  <p>
  <button class="toggle-buttons"
onclick="a()">&nbsp;&nbsp;Auto&nbsp;&nbsp;</button>
  <button class="toggle-buttons" onclick="m()">Manual</button>
  </p>
  </div>
  <div>
  <p>
  <label for="threshold" class="dht-labels">Threshold:</label>
  <input id="threshold" class="auto-buttons" type="number" min="-25" max="50"
value="30" step="0.5" disabled/> <sup class="units">&deg;C</sup>
  <button class="auto-buttons" onclick="threshold()" disabled>Set</button>
  </p>
  <p>
  <button class="manual-buttons" onclick="fon()">Fan On</button>
  <button class="manual-buttons" onclick="foff()">Fan Off</button>
  </p>
  <p>
  <button class="manual-buttons" onclick="ron()">Relay On</button>
  <button class="manual-buttons" onclick="roff()">Relay Off</button>
  </p>
  </div>
</body>
<script>

function a() {
  let autobutton = document.querySelectorAll(".auto-buttons");
  for (var i = 0; i < autobutton.length; i++) {
    autobutton.item(i).disabled = false;
  }
  let button = document.querySelectorAll(".manual-buttons");
  for (var i = 0; i < button.length; i++) {
    button.item(i).disabled = true;
  }
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("s").innerHTML = "Auto";
    }
  };
  xhttp.open("GET", "/a", true);
  xhttp.send();
}

function m() {
  let autobutton = document.querySelectorAll(".auto-buttons");
  for (var i = 0; i < autobutton.length; i++) {
    autobutton.item(i).disabled = true;
  }
  let manualbutton = document.querySelectorAll(".manual-buttons");
  for (var i = 0; i < manualbutton.length; i++) {
    manualbutton.item(i).disabled = false;
  }
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("s").innerHTML = "Manual";
    }
  };
  xhttp.open("GET", "/m", true);
  xhttp.send();
}
function threshold() {
  var threshold = document.getElementById("threshold").value;
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "/threshold?threshold=" + threshold, true);
  xhttp.send();
}
function fon() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("f").innerHTML = "On";
    }
  };
  xhttp.open("GET", "/fon", true);
  xhttp.send();
}
function foff() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("f").innerHTML = "Off";
    }
  };
  xhttp.open("GET", "/foff", true);
  xhttp.send();
}
function ron() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("r").innerHTML = "On";
    }
  };
  xhttp.open("GET", "/ron", true);
  xhttp.send();
}
function roff() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("r").innerHTML = "Off";
    }
  };
  xhttp.open("GET", "/roff", true);
  xhttp.send();
}
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 1000 ) ;
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 1000 ) ;
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("s").innerHTML = (this.responseText == "1")?
"Auto" : "Manual";
    }
  };
  xhttp.open("GET", "/s", true);
  xhttp.send();
}, 1000 ) ;
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("f").innerHTML = (this.responseText == "1")?
"On" : "Off";
    }
  };
  xhttp.open("GET", "/f", true);
  xhttp.send();
```

```
}, 1000 ) ;
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("r").innerHTML = (this.responseText == "1")?
"On" : "Off";
    }
  };
  xhttp.open("GET", "/r", true);
  xhttp.send();
}, 1000 ) ;
</script>
</html>)rawliteral";

// Replaces placeholder with DHT and Status values
String processor(const String& var){
  //Serial.println(var);
  if(var == "TEMPERATURE"){
    return String(t);
  }
  else if(var == "HUMIDITY"){
    return String(h);
  }
  else if(var == "STATUS"){
    return String(autostate);
  }
  else if(var == "FANSTATUS"){
    return String(f);
  }
  else if(var == "RELAYSTATUS"){
    return String(r);
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
  dht.begin();
  pinMode(FANPIN, OUTPUT);
  pinMode(RELAYPIN, OUTPUT);
  digitalWrite(FANPIN, LOW);
  digitalWrite(RELAYPIN, LOW);
  Serial.print("Setting AP (Access Point)…");
  // Remove the password parameter, if you want the AP (Access Point) to be
open
  WiFi.softAP(ssid); // password
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  // Print ESP8266 Local IP Address
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(t).c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(h).c_str());
  });
  server.on("/s", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(autostate).c_str());
  });
  server.on("/a", HTTP_GET, [](AsyncWebServerRequest *request){
    autostate=1;
    request->send_P(200, "text/plain", String(autostate).c_str());
  });
  server.on("/m", HTTP_GET, [](AsyncWebServerRequest *request){
    autostate=0;
    request->send_P(200, "text/plain", String(autostate).c_str());
  });
  server.on("/threshold", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    String inputParam;
    // GET threshold value on <ESP_IP>/threshold?threshold=<inputMessage>
    if (request->hasParam(thresholdInput)) {
      inputMessage = request->getParam(thresholdInput)->value();
      inputParam = thresholdInput;
    }
    else {
      inputMessage = threshold;
      inputParam = "none";
    }
    // Change Threshold value locally, with conditions
    inputMessage = inputMessage.toFloat();
    if ((inputMessage.toFloat() != threshold) && (inputMessage.toFloat() >= -
25) && (inputMessage.toFloat() <= 50)) {
      threshold = inputMessage.toFloat();
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", inputMessage);
  });
  server.on("/f", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(digitalRead(FANPIN)).c_str());
  });
  server.on("/fon", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(FANPIN, HIGH);
    f = 1;
    request->send_P(200, "text/plain", String(f).c_str());
  });
  server.on("/foff", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(FANPIN, LOW);
    f = 0;
    request->send_P(200, "text/plain", String(f).c_str());
  });
  server.on("/r", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(digitalRead(RELAYPIN)).c_str());
  });
  server.on("/ron", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(RELAYPIN, HIGH);
    r = 1;
    request->send_P(200, "text/plain", String(r).c_str());
  });
  server.on("/roff", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(RELAYPIN, LOW);
    r = 0;
    request->send_P(200, "text/plain", String(r).c_str());
  });
  // Start server
  server.begin();
}

void loop(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you updated the DHT values
    previousMillis = currentMillis;
    // Read temperature as Celsius (the default)
    float newT = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //float newT = dht.readTemperature(true);
    // if temperature read failed, don't change t value
    if (isnan(newT)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      t = newT;
      Serial.println(t);
    }
    // Read Humidity
    float newH = dht.readHumidity();
    // if humidity read failed, don't change h value
    if (isnan(newH)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      h = newH;
      Serial.println(h);
    }
    // Conditions for Automation
    if (autostate == 1) {
      if (t >= threshold) {
        digitalWrite(RELAYPIN, HIGH);
        server.on("/ron", HTTP_GET, [](AsyncWebServerRequest *request){
        r = 1;
        request->send_P(200, "text/plain", String(1).c_str());
      });
        digitalWrite(FANPIN, HIGH);
        server.on("/fon", HTTP_GET, [](AsyncWebServerRequest *request){
        f = 1;
        request->send_P(200, "text/plain", String(1).c_str());
      });
      }
      else {
        digitalWrite(RELAYPIN, LOW);
        server.on("/roff", HTTP_GET, [](AsyncWebServerRequest *request){
        r = 0;
        request->send_P(200, "text/plain", String(0).c_str());
        });
        digitalWrite(FANPIN, LOW);
        server.on("/foff", HTTP_GET, [](AsyncWebServerRequest *request){
        f = 0;
        request->send_P(200, "text/plain", String(0).c_str());
        });
      }
    }
  }
}
```

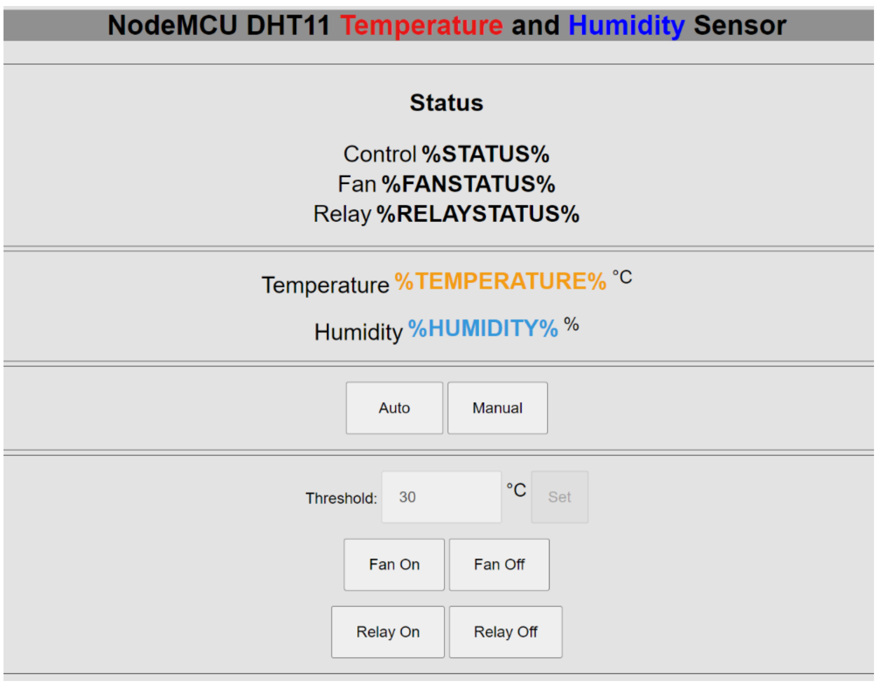
The code is hosted at [9].

Working

A still of the working of the Home Automation System, with the Web App on the screen:



A still of the web app hosted on the IP address 192.168.4.1 on a desktop (with placeholders) and mobile (with sensor values):



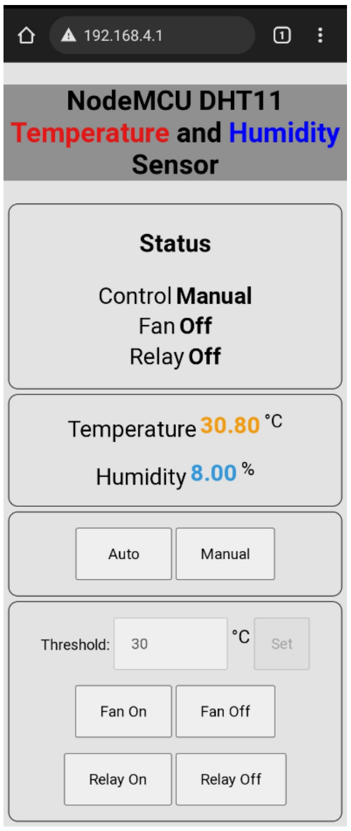
NodeMCU DHT11 Temperature and Humidity Sensor

Status

Control **%STATUS%**
Fan **%FANSTATUS%**
Relay **%RELAYSTATUS%**

Temperature **%TEMPERATURE%** °C

Humidity **%HUMIDITY%** %



192.168.4.1

NodeMCU DHT11 Temperature and Humidity Sensor

Status

Control **Manual**
Fan **Off**
Relay **Off**

Temperature **30.80** °C

Humidity **8.00** %

References

[1] Home automation – Wikipedia - https://en.wikipedia.org/wiki/Home_automation

[2] NodeMCU – Wikipedia - https://en.wikipedia.org/wiki/NodeMCU

[3] Overview | DHT11, DHT22 and AM2302 Sensors | Adafruit Learning System - https://learn.adafruit.com/dht/overview

[4] pc817x_e - https://www.farnell.com/datasheets/73758.pdf

[5] adafruit/DHT-sensor-library: Arduino library for DHT11, DHT22, etc Temperature & Humidity Sensors - https://github.com/adafruit/DHT-sensor-library

[6] Arduino/libraries/ESP8266WiFi at master · esp8266/Arduino · GitHub - https://github.com/esp8266/Arduino

[7] me-no-dev/ESPAsyncWebServer: Async Web Server for ESP8266 and ESP32 - https://github.com/me-no-dev/ESPAsyncWebServer

[8] W3Schools Online Web Tutorials - https://www.w3schools.com/

[9] mtalirfan/EE-227-Home-Automation-System - https://github.com/mtalirfan/EE-227-Home-Automation-System