

Use of IoT Sensor Networks in the Home

Abir Faisal, Paris Isley, Tutku Gizem Guder

Group: **UG18-4164**

CNT4164 - Intro to IOT and Sensor Networks - Final Project

Prof. Mohammad Ilyas

Project Repository: https://github.com/AbirFaisal/CNT4164_GroupProject

Abstract—A brief summary of the entire paper, including the research question, methodology, key findings, and conclusions.

This paper explores the integration of IoT sensor networks in smart home environments, focusing on climate control as a key application. We present a system utilizing an RP2040W IoT device and ESP01 nodes equipped with AHT20 temperature and humidity sensors. The system is designed to create a smart thermostat that can monitor and control the home's climate efficiently. The paper details the setup, functionality, and potential extensions of this sensor network, including its ability to provide granular environmental data and automate climate control. We also discuss the broader implications of IoT devices in various home applications, from security to energy management, highlighting the benefits and challenges of implementing such technologies.

Index Terms—Sensor Networks, IoT, Internet of Things, Smart Systems, Smart Homes

I. INTRODUCTION

THE idea of the smart home has been around for a long time. IoT devices enable the smart devices to be networked. The use of these devices can provide users with several advantages, such as increased comfort, increased safety, improved home management and maintenance, energy savings, automation and convenience in general.

Furthermore, sensor networks augment the use of IoT devices by collecting data around the home. Unlike centralized sensing, a network of sensors can provide a more granular understanding of the environment. These sensors can be placed throughout the home, and connect using existing network infrastructure.

In this report we show the use of a sensor network that could be used for climate control.

II. PROJECT DESCRIPTION

We are using an RP2040W as an IoT device, this will contain a web interface and it will query nodes on the sensor network. We did not have adequate time to implement this in hardware, it is being mentioned for the purpose of explaining the concept. Instead we have created a web interface that queries the sensor for the purpose of demonstration. This can be seen in [Figure 3]

We're using an ESP01 devices as a nodes for the sensor network. It will be paired with an AHT20 temperature and humidity sensor, which will be connected to the ESP01 over i2c.

The combination of the IoT device and the sensor network will create a smart system, specifically a smart thermostat. In this implementation, the IoT device also serves as the smart system because it would have direct control of the air conditioner.

A. Sensor

An image of the sensor node can be seen in [Figure 1]. The buttons are only for debugging purposes such as resetting the controller and putting it into programming mode to flash the firmware. They are not used for any other function.

Setting up the device on the network is simple. The device presents a WiFi access point that you can connect to using your phone or computer. After you connect, a captive portal is presented where you can enter the necessary credentials such as the WiFi SSID and the password to connect the sensor to the network. After this the device connects to the specified network, and presents a server that returns a JSON that contains the readings for the onboard sensors. If the network configuration is changed, or the device is unable to find or connect to the configured network, the device presents an access point again and the user can repeat this process.

III. RESULTS

The sensor responds with a JSON formatted string when sent an HTTP request to its IP Address.

Here is an sample, a demonstration can be seen in [Figure 2]:

```
{"id": 1, "temperature": 25.347, "humidity": 35.634}
```

The temperature is reported in Celsius, and the humidity is reported in percent relative humidity.

IV. IoT DEVICES IN HOME SENSOR NETWORKS

There are many areas where sensors can be used in the home. Here is a non-comprehensive list of examples and areas where IoT devices and sensor networks can be used in the home:

- Security and Surveillance

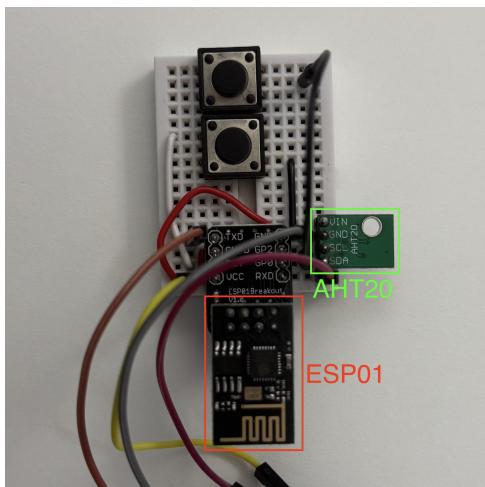


Fig. 1: ESP01 + AHT20 Sensor

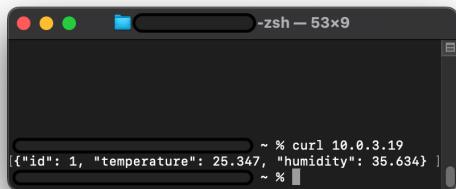


Fig. 2: Response from sensor using curl command

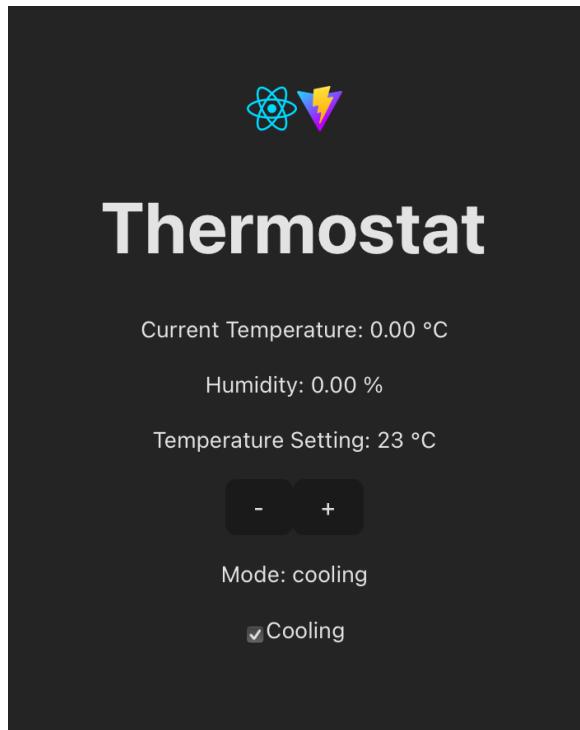


Fig. 3: Demo Web Interface using ReactJS

- Smart cameras
- Motion detectors
- Window/door sensors
- Security alarm systems
- Personnel Detection
- Intrusion Detection
- **Climate Control**
 - Smart thermostats
 - Air quality sensors (e.g., particulate matter, CO2)
 - Humidity sensors
 - Temperature sensors
 - Smart air purifiers
- **Lighting**
 - Smart bulbs
 - Smart switches and outlets
 - Motion-activated lighting
 - Color-changing LED strips
 - Outdoor lights with timers/sensors
- **Energy and Resource Monitoring**
 - Smart plugs
 - Energy monitoring systems
 - Smart meters for electricity, gas, or water
 - Power strips with energy tracking
 - Grid Power
 - Solar Panels
- **Audio and Video**
 - Smart speakers (e.g., Amazon Echo, Google Nest)
 - Soundbars with smart features
 - Wi-Fi enabled headphones
 - Streaming devices (e.g., Roku, Apple TV)
 - IP intercom systems
- **Cooking and Appliance Control**
 - Smart ovens/ranges
 - Coffee makers
 - Connected refrigerators with inventory management
 - Dishwashers with smart scheduling
 - Microwave ovens
- **Entertainment and Gaming**
 - Smart TVs with built-in apps
 - Video game consoles
 - VR headsets with home tracking sensors
 - Home theater systems with smart features
- **Health and Wellness**
 - Smartwatches and fitness bands
 - Blood pressure monitors
 - Weight scales with data tracking
 - Sleep monitoring devices
 - Smart water bottles with usage tracking
- **Home Automation**
 - Smart hubs (e.g., Amazon Alexa, Google Home)
 - Zigbee or Z-Wave compatible devices
 - Automated curtains and blinds

- Smart locks with keyless entry
- Connected garage door openers
- **Water Management**
 - Water leak detectors
 - Smart sprinklers with soil moisture sensors
 - Toilet overflow sensors
 - Smart water heaters
- **Miscellaneous Smart Devices**
 - Smart mirrors with calendar and weather displays
 - Connected waste bins with fill level monitoring
 - Smart key fobs for location tracking
 - Plant monitors for soil moisture and light levels
 - Smart pet feeders
- **Other**
 - Smart Cars / Parking Lots / Garages
 - Healthcare at Home

V. INTERNET OF THINGS, SENSOR NETWORKS, AND SMART SYSTEMS

The IoT device in this use case serves two main responsibilities querying the sensors on the network, and providing a web interface.

The web interface should present a dashboard to the user where they can monitor the temperature and set preferences.

In addition there should be physical buttons to raise and lower the temperature if needed.

A. Smart System

A smart system is the combination of a sensor network, internet connectivity, and control.

A smart thermostat would take measurements from the sensor network to make decisions. Because of this control loop, this would be considered a smart system.

How should the system decide what to do? Using the average of the sensors, biased by the highest or lowest reading, the thermostat would know how to maintain the temperature of the room.

Assume S is a set of readings from the sensor network:

$$S = \{sensor_1, sensor_2, sensor_3, \dots, sensor_n\}$$

If the thermostat is set to maintain coolness, then the highest reading would be used as the bias:

$$T_c = \frac{1}{n+1} \left[\max_{s \in S} + \sum_{s \in S} s \right]$$

If the thermostat is set to heat, then the lowest temperature reading would be used as the bias:

$$T_c = \frac{1}{n+1} \left[\min_{s \in S} + \sum_{s \in S} s \right]$$

Where T_c is the computed temperature.

It is possible to create a smart system that automatically heats and cools as needed, but this can result in a control loop that oscillates if not done correctly. This is why thermostats generally do not have this functionality. A control loop such as PID can manage a system like this, however the engineering expense for such a system would far exceed the benefits outside of a controlled environment such as a laboratory. A common change such as new furniture or home improvement can require calibration of the PID parameters.

A simple solution for this can be the use of threshold values. This assumes that the average daily temperature fluctuation is known. If the temperature is within the specified range, nothing is done,

For example a target temperature of 75 is set, with a range of 68-78. If the temperature rises above 78, the system cools to 75, if the temperature drops below 68, the system heats to 75.

$$T_c = \frac{1}{n} \sum_{s \in S} s$$

$$heat = T_c \leq T_l$$

$$cool = T_c \geq T_u$$

$$off = T_l < T_c < T_u$$

$$action = argmax(heat, cool, off)$$

Where T_l is the lower bound, and T_u is the upper bound.

As long as the average daily temperature fluctuation is within the bounds, this system should not oscillate.

Furthermore such a system would be simple enough for an end user to configure themselves, and the system can be extended to use external APIs to retrieve weather data and adjust this range automatically.

VI. NETWORKING AND TOPOLOGY

A. Networking

If a home has internet connectivity, it means that there is a existing network infrastructure. Also the majority of ISP gateways already contain wireless networking capabilities.

As of 2021, 90% percent of U.S households have a broadband internet subscription [2].

Given this, it makes sense that the optimal way (in terms of cost and configuration) for sensor networks to communicate, through this wireless network.

B. Topology

Given the selected networking method, the best topology for the sensor network is going to be a star network. This topology allows each node in the sensor network to notify a central IoT device (the RP2040W) of its readings. Alternatively the IoT device can scan for and query each sensor. Also a house would have at most a few dozen sensors, so there is no risk of congestion on modern home networking hardware. Even if there were 1000 sensors, this would not be

a problem for modern hardware because the amount of data being transmitted is less than 1KB, and each device is queried sequentially.

The size of an standard network packet is 1518 bytes, and the sensor returns approximately 114 bytes(chars) of data, therefore the response data should fit into a single network transmission.

For the user it's simple and easy to set up, anyone that can connect a device to their WiFi can connect a sensor to the network. The sensor presents a wifi AP, the user connects to it, it makes a setup portal appear on their device, and they simply enter their wifi credentials.

For sensors that may be out of range, off the shelf WiFi repeaters may be used.

VII. POWER

A. IoT Power

Thermostats contain existing wiring for power, usually 24v AC. For this project we are using a 24v AC to 12v DC converter, which will then be stepped down to 5v to power the IoT device.

Home HVAC systems are controlled using relays, so the 12v rail can be used to trigger the relay.

This controller is programmed over the USB port, so we are using that to power the device.

B. Sensor network power

According to the data sheet for the ESP8266, it should consume approximately 200mA under load. According the data sheet for the AHT20, the sensor typically consumes about 3mA. Both the sensor and controller will accept 3.3v.

Therefore a 3.3v power supply of 250mA (approximately 0.825W) should be sufficient.

For areas with sufficient lighting combination of solar and battery could be an option. However it makes more sense to power the nodes using a USB adapter.

For the purpose of this project, we are simply powering it on the breadboard using the power from programmer.

VIII. SECURITY

Because the devices connect to a WiFi network, some of the security for the sensor network is already provided by the encryption mechanisms of the wireless network. The IoT device however must be hardened against attacks because this is directly exposed to the internet. In addition if the sensors are on a shared network, there should be security mechanisms to prevent spoofing the IoT device with false readings. The sensor in our project is read-only and has no way being manipulated as far as we know. This should be sufficient security for most users.

A. Software Updates

All though we did not implement over the air updates. Updating the sensor network is possible using the ArduinoOTA.h library.

It is also possible to update the IoT device, however that can present security risks because the IoT device is exposed to the internet.

One way to mitigate this is to run the update endpoint on a specific port, block the port in the firewall, and require the user to update the device from their local network.

Alternatively the update mechanism can be left exposed to the internet and more sophisticated methods can be used such as code signing and requiring authentication, so that the device will only accept firmware that has been signed with the appropriate keys, and only after being properly authenticated. Furthermore the update process would require end to end encryption to prevent MITM attacks.

IX. CONCLUSIONS AND FURTHER RESEARCH

IoT has revolutionized the way we interact with our homes by allowing us to have intelligent living spaces.

The use of sensor networks in the home, like motion detectors, temperature sensors, and air quality sensors continuously collect and analyze data, enhancing security, saving energy, and automating everyday tasks.

According to security.org [8, Vigderman], 78% of prospective home buyers are willing to pay more for smart home features, if you're wondering why people pick them. Furthermore, according to 82% of tenants, their residences should include at least one smart gadget [4, Howell].

Based on data, you can presume that consumers recognize the value and appreciate convenience, security, and cost savings with cool features like voice-activated smart speakers, remotely controlled smart lighting, or multi functional smart appliances. 79% of users say smart home gadgets have enhanced their life, increasing productivity by 44%, security by 45%, and overall happiness by 57% [3, Howarth].

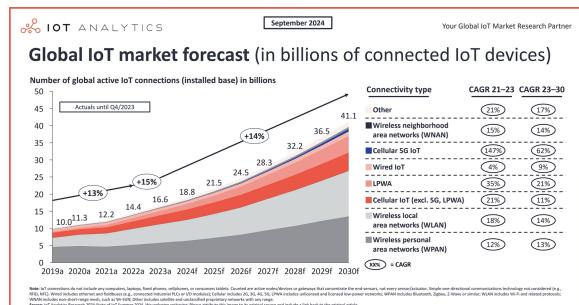


Fig. 4: Market Forecast - [7, Sinha]

The function of smart devices intended for security is to provide homeowners with peace of mind. The purpose of motion floodlights, motion sound alarms, smart doorbells, and smart cameras is to deter crime.

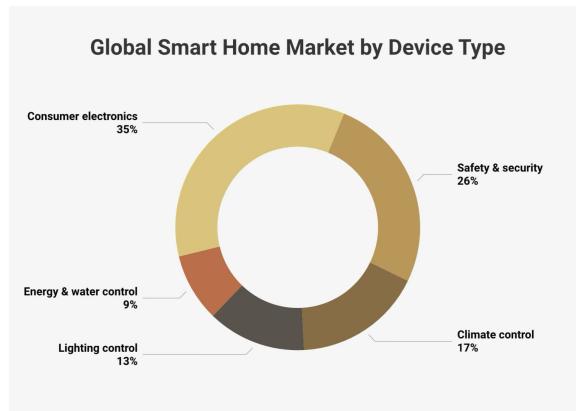


Fig. 5: Market Share By Type - [5, Mariotti]

According to the data in [Figure 5], 26% of all smart home devices fall under the Safety & Security category.

Devices for lighting and climate control are also included in the graph above. Climate control equipment includes space heaters, fans, humidifiers, and smart thermostats, among others. Users can control the temperature of their homes thanks to all of these IoT-connected devices.

In order to prevent users from returning home to a dark house, it would be possible to remotely control the lighting and climate such that the power would be turned off while no one was home. When the user returns, the sensors can update the IoT device to turn them back on.

The Energy and Water Control devices make up the graph's smallest category. This includes smart sprinkler systems, smart plugs, smart sensors, and smart appliances. Users might put a smart sensor beneath sinks in this application, which would notify a smartphone in the event of a leak or when it senses moisture. When a user has small appliances that aren't automatically smart, smart plugs can be helpful. Users could remotely manage the appliance's power operations by using a smart plug. This is more efficient and conserves energy.

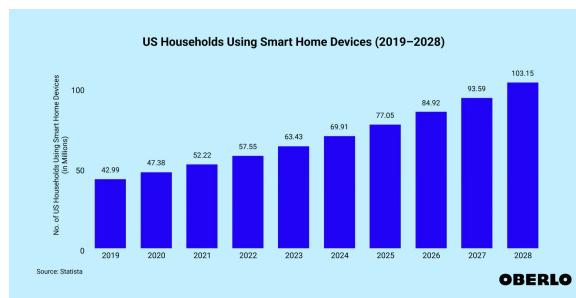


Fig. 6: Households using smart devices [6, oberlo]

A. Additional Thoughts - Energy Savings

The key to energy saving lies in better management of smart home devices. Optimizing our home's energy

consumption systems will reduce our energy consumption. Smart devices and IoT Sensors in homes mean higher overall efficiency, through better remote control.

B. Additional Thoughts - Home Automation

IoT automation connects physical objects to the internet, allowing for seamless data exchange and improved operational efficiency. This integration of IoT tech is transforming industries by enabling real-time data processing and communication between machines. Facilitating automation and remote monitoring, IoT drives the digital transformation of manufacturing and other industries, leading to smarter and more efficient operations.

C. Additional Thoughts - Growth of IoT adoption in Homes

IoT adoption and integration have increased over the past five years due to trends. According to Parks Associates [1, Parks], the average US household owns on average 16 connected devices. Which is an increase from 2021 from 13 connected devices. Through home automation and security applications, energy management and energy efficiency services, new video distribution options through various connected devices, and remote control and monitoring capabilities, smart home, connected health, and connected entertainment products offer consumers a higher level of convenience.

REFERENCES

- [1] Parks Associates. Parks: Average u.s. internet home had 17 connected devices in 2023. <https://www.parksassociates.com/blogs/in-the-news/parks-average-us-internet-home-had-17-connected-devices-in-2023>, January 2023.
- [2] US Census Bureau. Computer and internet use in the united states: 2021. <https://www.census.gov/newsroom/press-releases/2024/computer-internet-use-2021.html>, June 2024. Press Release No. CB24-TPS.61.
- [3] Josh Howarth. 50+ smart home statistics (new 2024 data). <https://explodingtopics.com/blog/smart-home-stats>, February 2024. Key Findings.
- [4] By Nicki Howell. How property managers can keep up with today's renters. <https://www.globest.com/2025/03/18/how-property-managers-can-keep-up-with-todays-renters/?slreturn=2025032850717>, March 2025.
- [5] Tony Mariotti. Smart home statistics. <https://www.rubyhome.com/blog/smart-home-stats/>, August 2023.
- [6] oberlo. Us smart home statistics (2019–2028). <https://www.oberlo.com/statistics/smart-home-statistics>.
- [7] Satyjit Sinha. State of iot 2024: Number of connected iot devices growing 13%. <https://iot-analytics.com/number-connected-iot-devices/>, September 2024.
- [8] Aliza Vigderman. Smart home consumer trends and shopping insights: 2021. <https://www.security.org/smart-home/consumer-shopping-insights/>, October 2021. Key Findings.

3RD PARTY LIBRARIES

github.com/robtiltaart/DHT20
github.com/tzapu/WiFiManager

1 Source Code - Sensor

```
/***
 * @file main.cpp
 * @brief IoT and Sensor Networks project
 * @author Abir Faisal <noreply@fau.edu>
 * @date 2025-02-06
 * @copyright Copyright (c) 2025 Florida Atlantic University/Abir Faisal
 * This software is released under the GNU GPLv3 License, see LICENSE.md
 */

/*
ESP8266 Pinout

RX/GPIO3      1 2  3.3v VCC
GPIO0/SDA     3 4  RESET
GPIO2/SCL     5 6  ENABLE
GND          7 8  GPIO1/TX/LED

*/
/*
Behaviors

Attempt to find AP
Attempt to connect

if success, respond with JSON
else
    create AP
    allow connection
    present setup page
*/

#define DEBUG 1

#define DEFAULT_SSID ''
#define DEFAULT_WPA2 ''

#define SDA_PIN 0
#define SCL_PIN 2

#include <Arduino.h>
#include "sensor.h"
#include "sensorserver.h"

class Main
{
public:
    // Services
    Sensor sensor0;
    SensorServer sensorServer0;

    // shared variables
}
```

```

float temperature = 0.0;
float humidity = 0.0;

Main()
{
}

void setup(){
    sensor0.setup();
    sensorServer0.setup();
}

void run() {
    // Serial.println("Running...");

    // run and update
    sensor0.run(&temperature, &humidity);

    sensorServer0.run(temperature, humidity);

    // print status
    // Serial.print("temperature ");
    // Serial.println(temperature);

    // Serial.print("humidity ");
    // Serial.println(humidity);

    // delay(500);
}

void blink()
{
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
};

Main prog;

void setup()
{
    Serial.begin(115200);
    delay(1000); // wait for serial connection to be established
    Serial.println("Starting...");

    // Wire.begin(SDA_PIN, SCL_PIN); // Initialize the I2C bus with defined SDA and SCL pins
    prog.setup();
}

void loop()
{
    prog.run();
}

```

```
}
```

```
/**  
 * @file sensor.cpp  
 * @brief Updates information for a connected sensor.  
 * @author Abir Faisal <noreply@fau.edu>  
 * @date 2025-02-06  
 * @copyright Copyright (c) 2025 Florida Atlantic University/Abir Faisal  
 * This software is released under the GNU GPLv3 License, see LICENSE.md  
 */  
  
#include "sensor.h"  
  
Sensor::Sensor() {  
}  
  
Sensor::~Sensor() {  
}  
  
void Sensor::setup()  
{  
    // Wire.begin(SCA, SDL)  
    Wire.begin(SDA_PIN, SCL_PIN);  
  
    bool is_connected = AHT.begin();  
  
    Serial.print("is_connected_");  
    Serial.println(is_connected);  
  
    // if (Serial.available())  
    // {  
    //     Serial.print("Init_Sensor");  
    //     Serial.print(DHT20_LIB_VERSION);  
    // }  
}  
  
void Sensor::run(float* temperature, float* humidity)  
{  
    #ifdef DEBUG  
    Serial.print("Sensor_Run_State:");  
    Serial.println(state);  
    #endif  
  
    switch (state)  
    {  
        case 0:  
            // Request  
            AHT.requestData();  
            state++;  
            break;  
        case 1:  
            // Read  
            AHT.readData();  
    }
```

```

        state++;
        break;
    case 2:
        // Convert
        AHT.convert();

        // Update
        *temperature = getTemperature();
        *humidity = getHumidity();

        state++;
        break;
    default:
        state = 0;
        break;
    }
}

float Sensor::getTemperature()
{
    return AHT.getTemperature();
}
float Sensor::getHumidity()
{
    return AHT.getHumidity();
}



---


</*
 * @file sensor.h
 * @brief Updates information for a connected sensor.
 * @author Abir Faisal <noreply@fau.edu>
 * @date 2025-02-06
 * @copyright Copyright (c) 2025 Florida Atlantic University/Abir Faisal
 * This software is released under the GNU GPLv3 License, see LICENSE.md
 */

#define SDA_PIN 0
#define SCL_PIN 2

#include "DHT20.h"

class Sensor {

    public:

        int read_rate = 5000; // ms

        DHT20 AHT;

        uint8_t state = 0;

        Sensor();
        ~Sensor();

    void setup();
}

```

```

        void run(float* t, float* h);
        float getTemperature();
        float getHumidity();

    };



---


/*
 * @file webserver.cpp
 * @brief Webserver that allows configuration or returns data from sensor.
 * @author Abir Faisal <noreply@fau.edu>
 * @date 2025-03-04
 * @copyright Copyright (c) 2025 Florida Atlantic University/Abir Faisal
 * This software is released under the GNU GPLv3 License, see LICENSE.md
 */

#include "sensorserver.h"

SensorServer::SensorServer() : server(80)
{
}

SensorServer::~SensorServer()
{
}

void SensorServer::setup()
{
    // Initialize WiFiManager
    wm.WiFiManagerInit();

    // Set a static IP configuration for the Access Point (AP) mode
    wm.setAPStaticIPConfig(IPAddress(192, 168, 0, 1), IPAddress(192, 168, 0, 1), IPAddress(255, 255,
        // Clears the settings stored in the EEPROM
        // wm.resetSettings();

        // Attempt to auto connect to a previously saved WiFi network using WiFiManager
        bool res = wm.autoConnect();

        // Check if the connection attempt was successful
        if (!res)
        {
            // If not successful, print an error message to the serial monitor
            Serial.println("Failed_to_connect");
            // Optionally, restart the ESP32/ESP8266 to try again (uncomment the line below)
            // ESP.restart();
        }
        else
        {
            // If connected successfully, print a success message to the serial monitor
            Serial.println("WiFi_connecateda_sucessafully:");
            server.begin(80);
        }
    }
}

```

```

void SensorServer::run(float temperature, float humidity)
{
    client = server.available();

    char buffer[10];

    if (client)
    {
        Serial.println("Has_client");

        String request = "";

        while (client.connected())
        {
            // Serial.println("Client is connected");
            if (client.available())
            {
                char c = client.read();
                Serial.println(c);
                if (c == '\n')
                {
                    // If the current line is blank, skip it.
                    if (request.length() == 0)
                    {
                        // Response Header
                        client.println("HTTP/1.1 200 OK");
                        client.println("Content-Type: application/json; charset=utf-8");
                        client.println("Connection: close");
                        client.println();

                        // Response Body
                        client.print("{\"id\":1,\"temperature\":");
                        dtostrf(temperature, 2, 3, buffer);
                        client.print(buffer);

                        client.print(",\"humidity\":");
                        dtostrf(humidity, 2, 3, buffer);
                        client.print(buffer);

                        client.print("}");

                        // End
                        client.println();
                        break;
                    }
                }
                else
                {
                    request = "";
                }
            }
            else if (c != '\r')
            {
                // Otherwise the char is a normal char:
                request += c;
            }
        }
    }
}

```

```

        }
    }

    client.stop();
    // Serial.println("Client Disconnected");
}



---


/***
 * @file webserver.h
 * @brief Webserver that allows configuration or returns data from sensor.
 * @author Abir Faisal <noreply@fau.edu>
 * @date 2025-03-04
 * @copyright Copyright (c) 2025 Florida Atlantic University/Abir Faisal
 * This software is released under the GNU GPLv3 License, see LICENSE.md
 */

#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>

#include <WiFiManager.h>

#include <Arduino.h>

class SensorServer {

public:

    WiFiServer server;

    WiFiClient client;
    WiFiManager wm;

    int PORTAL_TIMEOUT = 15;

    SensorServer();
    ~SensorServer();

    void setup();
    void run(float temperature, float humidity);

};


```

2 Source Code – Web Interface

```
// Main.jsx
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)

/**
 * @file App.jsx
 * @brief IoT and Sensor Networks project
 * @author Abir Faisal <noreply@fau.edu>
 * @date 2025-02-06
 * @copyright Copyright (c) 2025 Florida Atlantic University/Abir Faisal
 * This software is released under the GNU GPLv3 License, see LICENSE.md
 */
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

import { useEffect } from 'react'

function App() {
  const [temperature, setTemperature] = useState(0)
  const [humidity, setHumidity] = useState(0)
  const [temperatureSetting, setTemperatureSetting] = useState(20)
  const [mode, setMode] = useState('heating')

  const fetchData = async () => {
    try {
      // process.env.NODE_TLS_REJECT_UNAUTHORIZED = '0';

      const response = await fetch('10.0.3.19', {
        referrerPolicy: "unsafe-url",
      })
      console.log('response:', response)

      if (!response.ok) {
        throw new Error(`HTTP error! Status: ${response.status}`)
      }
      const data = await response.json()
      setTemperature(data.temperature)
      setHumidity(data.humidity)
    } catch (error) {
```

```

        console.error('Error fetching data:', error)
    }
}

useEffect(() => {
    const interval = setInterval(() => {
        fetchData()
    }, 5000) // fetch data every 5 seconds

    return () => clearInterval(interval)
}, [])

const increaseTemperatureSetting = () => {
    setTemperatureSetting(prevSetting => prevSetting + 1)
}

const decreaseTemperatureSetting = () => {
    setTemperatureSetting(prevSetting => prevSetting - 1)
}

const toggleMode = () => {
    setMode(prevMode => (prevMode === 'heating' ? 'cooling' : 'heating'))
}

return (
    <>
    <div className="App">
        <header className="App-header">
            <img src={reactLogo} className="App-logo" alt="logo" />
            <img src={viteLogo} className="App-logo" alt="logo" />
            <h1>Thermostat</h1>
            <p>Current Temperature: {temperature.toFixed(2)} C</p>
            <p>Humidity: {humidity.toFixed(2)} %</p>
            <p>Temperature Setting: {temperatureSetting} C</p>
            <button onClick={decreaseTemperatureSetting}>-</button>
            <button onClick={increaseTemperatureSetting}>+</button>
            <p>Mode: {mode}</p>
            <label>
                <input
                    type="checkbox"
                    checked={mode === 'cooling'}
                    onChange={toggleMode}
                />
                Cooling
            </label>
        </header>
    </div>
)
}

export default App

```